User's Manual

Engineering Capture System

CAD/CAM Group, Inc.

Fourth Edition
February 1992

Table of Contents

Table of Contents iii
Preface xiii
Using This Manual xv

Section I Before You Begin

ECS Conventions I-2
ECS on Sun Workstations I-3
- Hardware Configuration I-3
- Software Configuration I-3
- Installing the ECS I-3
- Starting ECS I-4
- Using Sunview I-5
  - What is a Window? I-5
  - Using the Mouse I-6
  - Selecting I-7
    - Selecting the Active Window I-7
  - Using Menus I-7
    - Walk-Through Menus I-7
    - Pull-Down Menus I-8
  - Dialog Boxes I-8
  - Using Scroll Bars I-9
  - Arranging Windows I-10
  - Making a Window into an Icon I-11
  - Quit an Application I-11
- The ECS Adaptation of Sunview I-11
  - Mouse Buttons I-11
  - Selection I-11
  - Cut versus Delete I-12
  - Copy/Paste versus Duplicate I-12
  - Cut/Paste vs. Move I-12
  - Undo and Redo I-12
- Prompting and Error Messages I-12

ECS on IBM PC Compatible Systems I-13
- Hardware Configuration I-13
- Software Configuration I-13
- Installing Microsoft Windows I-14
- Installing the Engineering Capture System I-14
- Starting ECS I-15
- Using Microsoft Windows I-15
  - What is a Window? I-15
  - Using the Mouse I-16

Selecting....................I-17
Selecting the Active Window....................I-17
Using Menus....................I-17
Dialog Boxes....................I-17
Using Scroll Bars....................I-19
Arranging Windows....................I-19
Enlarging a Window....................I-19
Making a Window into an Icon....................I-19
Quit an Application....................I-19
Exiting Windows....................I-20
The WIN.INI Initialization File....................I-20
The ECS Adaptation of Windows....................I-21
Mouse Buttons....................I-21
Selection....................I-21
Cut versus Delete....................I-22
Copy/Paste versus Duplicate....................I-22
Cut/Paste versus Move....................I-22
Undo and Redo....................I-22
Prompting and Error Messages....................I-23

Section II System Configuration

Libraries and Directory Structures....................II-2
Program Directories....................II-2
User Directories....................II-3
Library Directories....................II-3
Project Directories....................II-4
Model Directories....................II-4
Symbol Libraries....................II-4
Library Searching....................II-6
Memory Limitations....................II-7
Memory Size Limit on IBM compatible PC....................II-7
Segmentation Limit on IBM Compatible PC....................II-7
Subdividing a Schematic....................II-8
Subdividing the Hierarchy....................II-9
Graphic Control....................II-10
Setup of ecs.ini....................II-11
Location of the ecs.ini File on Sun Workstation....................II-11
Multiple ecs.ini Files....................II-11
Editing the Master ecs.ini File....................II-13
The ecs.ini Editor....................II-14
Attributes->Net, Pin, Symbol Attributes....................II-14
Modification of Symbol Attributes....................II-16
Modification of Pin Attributes....................II-16
Modification of Net Attribute....................II-16

- Example Attributes II-17
Controls->Color Assignment II-19
Controls->Color Palette II-20
Controls->ECS Controls II-21
- ApplicationMode II-21
- ClipboardFormat II-22
- DontClipPrinter II-22
- EditSuffix II-22
- FirstCharAlpha II-22
- HierarchyChar II-22
- IgnoreTreeStop II-22
- MarkOpenEnds II-23
- NameBarChar II-23
- PinNameOffset II-23
- PlotScale II-23
- PrintFormat II-23
- PrintScale II-24
- PullDownMenus II-24
- RetainedParentheses II-24
- RotateNetNames II-24
- RotatePinNumbers II-24
- ShowBorder II-24
- ShowCrossPageConnects II-25
- ShowNetNos II-25
- ShowPinDots II-25
- ShowPinNumbers II-25
- ShowSimVals II-25
- ShowSolderDots II-25
- ShowSymbolAttributes II-25
- ShowSymbolText II-25
- SnapToPins II-26
- Symbol Type II-26

Controls->Global Signals II-26
Controls->Processes II-28
Controls->PS Printer II-29
- Orientation II-29
- TypeFace II-29
- Setup Line II-29
- PrinterName II-30
- PaperWidth II-30
- PaperHeight II-30
- SideMargin II-30
- TopMargin II-30

Controls->Sheet Layout II-30
- Horizontal Zones II-30
- Direction II-30

Vertical Zones....................II-30
Direction....................II-31
Master Symbols....................II-31
Numbers on....................II-31
Units....................II-31
Grid Size....................II-31
Smallest Text....................II-31
Controls->Sheet Sizes....................II-31
Controls->Simulator Environment....................II-32
Controls->Tools....................II-32
Controls->Waves Controls....................II-33
File->Exit....................II-33
File->Save....................II-33
Search Paths....................II-33

Section III Getting Started

Concept of Design Entry with ECS....................III-2
Schematics....................III-3
Symbols....................III-3
Graphic Entities....................III-3
Pins....................III-3
Attributes....................III-4
Iterated Instances (Arrays)....................III-4
Wires....................III-5
Busing Features....................III-5
Simple Buses....................III-5
Ordered Buses....................III-6
Bus Pins....................III-6
Attributes....................III-6
Graphics....................III-6
Hierarchy....................III-7
Symbols, Schematics, and Hierarchy in ECS....................III-8
Hierarchical Design Structure....................III-9
Hierarchical Naming....................III-9
Nets in the Hierarchy....................III-10
Design Debugging and Verification Tools....................III-10
Automatic Aliasing....................III-11
After Design Entry is Complete....................III-11
A Brief Tour....................III-12
Starting the System....................III-12
Conventions Used in This Tutorial....................III-13
On-Line Help....................III-13
Designing a Circuit....................III-14
Creating Symbols....................III-14

Drawing the AND Gate III-17
Creating the Latch Symbol III-20
Drawing the Symbol III-20
Displaying Pin Names on Schematic III-21
Creating Schematics III-22
Placing Symbols III-22
Editing the Schematic III-23
Wiring the Schematic III-24
Naming Symbol Instances III-25
Naming Nets III-26
Saving the Schematic III-27
Check for Errors III-27
Creating the Reg4 Schematic III-27
Navigating the Hierarchy III-28
Error Checking III-30
Correct by Construction III-30
Symbol and Schematic Checking III-30
Engineering Analysis Tools III-31
Simulation III-31
Creating a Netlist III-31
Printing III-32
Exiting from the System III-33
Summary III-33

Section IV Entering Your Design

The ECS Executive Program IV-2
Dialog Box Interface IV-2
Selecting a Program IV-3
Selecting Design Files IV-3
Naming Design Files IV-4
Running a Program IV-4
Utility, Print, View and Setup Menus IV-5
The ECS Symbol Editor IV-7
Symbol Components IV-7
Graphic Entities IV-7
Pins IV-7
Attributes IV-8
Symbol Types IV-8
Component and Gate Symbols IV-8
Cell Symbols IV-8
Block Symbols IV-9
Pin Symbols IV-9
Graphic Symbols IV-9
Master Symbols IV-9

Creating Symbols....IV-9
Starting the Symbol Editor....IV-10
Grids....IV-10
Drawing Graphics and Fixed Text....IV-11
Rectangles....IV-11
Circles and Arcs....IV-11
Negation Bubbles....IV-11
Saving a Symbol....IV-12
Printing the Symbol....IV-12
Editing Symbols....IV-12
Delete....IV-13
Move....IV-13
Duplicate....IV-13
Drag....IV-13
Copy, Cut, and Paste....IV-13
Undo and Redo....IV-14
Symbol Preparation for Schematics....IV-14
Pins....IV-14
Bus Pins....IV-15
Attributes....IV-15
Pin Attributes....IV-15
Symbol Attributes....IV-16
Attribute Windows....IV-16
Set Origin....IV-17
Check....IV-17
Schematic Editor....IV-19
Schematic Components....IV-19
Symbols....IV-20
Wires....IV-20
Attributes....IV-20
Graphics....IV-20
Beginning the Design....IV-20
Starting the Schematic Editor....IV-20
Working with Sheets....IV-21
Grids....IV-21
Controlling Display Options....IV-22
Connect Dots....IV-22
Border....IV-22
Symbol Pins....IV-22
Pin Numbers....IV-22
Symbol Text....IV-22
Symbol Attributes....IV-22
Open Ends....IV-22
Cross Page Connects....IV-23
Drawing Graphics....IV-23
Adding Components....IV-23

Placing Symbols.......... IV-23
Adding Instance Names.......... IV-24
Iterated Instances.......... IV-24
Attributes.......... IV-25
Editing Pin Attributes.......... IV-25
Symbol Attributes.......... IV-26
Attribute Windows.......... IV-26
Wiring the Schematic.......... IV-26
Drawing Wires.......... IV-26
Net Names.......... IV-27
Identification.......... IV-27
Interconnection.......... IV-27
Bus Entry.......... IV-27
Hierarchy Connections.......... IV-27
Ordered Bus Definition.......... IV-27
Simple and Compound Names.......... IV-28
Entering Net Names.......... IV-28
Renaming a Net.......... IV-29
Attaching the Net Name to the Schematic.......... IV-29
Buses.......... IV-30
Bus Taps.......... IV-31
Implicit Net Connections.......... IV-31
Specifying Polarity.......... IV-32
Wiring Constraints.......... IV-32
Editing the Schematic.......... IV-33
Delete.......... IV-33
Move.......... IV-33
Duplicate.......... IV-34
Drag.......... IV-34
Rotate and Mirror.......... IV-34
The Clipboard and Copy, Cut, Paste.......... IV-34
Undo and Redo.......... IV-35
Connectivity Aids.......... IV-35
Check.......... IV-36
Output the Schematic.......... IV-37
Saving the Schematic.......... IV-37
Printing the Schematic.......... IV-37
Netlisting.......... IV-37
Creating Symbols in the Editor.......... IV-38
The Update Schematic Utility.......... IV-39
Hierarchy Navigator.......... IV-40
Navigating the Design.......... IV-40
Push/Pop.......... IV-41
Save.......... IV-41
Sheet.......... IV-42
Print.......... IV-42

Control IV-42
Statistics IV-43
Attributes..IV-43
Pin Attributes..IV-44
Symbol Attributes IV-44
Attribute Windows..IV-44
Verifying Design Entry IV-45
Tracing Signals..IV-45
Check Circuit IV-47
Circuit Analysis IV-47
Simulation..IV-47
Working with Split Hierarchy under DOS IV-48
The Interactive Logic Analyzer..IV-48
ProbeItem..IV-48
FindItem..IV-49
Netlisting..IV-49
Preparing for Manufacture IV-49
Pin Numbers..IV-50
Reference Designators IV-51
Modify Attributes..IV-51
Netlisting..IV-52
Attributes..IV-53
Attribute Types IV-53
Creating Attributes..IV-54
Assigning Values To Simple Attributes..IV-56
Displaying Attribute Values On A Schematic..IV-57
Derived Attributes..IV-58
Calculated Derived Attributes..IV-61
Derived Attributes and Hierarchy IV-62
Case Study Using Derived Attributes..IV-62
Number Notation In Attributes..IV-65
Differences in IC and PCB Design Entry IV-67
Symbol Types IV-67
DeMorgan Equivalent Gates IV-68
Instance Names and Reference Designators..IV-68
Assigning Reference Designators IV-69
Pin Numbers IV-69
Gate Assignment..IV-70
Pin Swapping..IV-70
Example PCB Design..IV-71
System Configuration IV-72
Creating a Gate Symbol..IV-72
Create the Latch Schematic..IV-73

Section V Command Reference

Quick Command Reference V-3
Add->Attribute V-5
Add->Bus Tap V-7
Add->Instance Name V-8
Add->I/O Marker V-10
Add->Net Name V-11
Add->Pin V-16
Add->Pin Attribute V-17
Add->Pin Name Location V-19
Add->Pin Numbers V-21
Add->Reference Designator V-22
Add->Symbol V-23
Add->Symbol Attribute V-25
Add->Window V-27
Add->Wire V-29
Draw->Arc V-31
Draw->Bubble / Big Bubble V-32
Draw->Circle V-33
Draw->Line V-34
Draw->Rectangle V-35
Draw->Text V-36
Edit->Copy V-37
Edit->Cut V-38
Edit->Delete V-39
Edit->Drag V-41
Edit->Duplicate V-43
Edit->Mirror V-45
Edit->Move V-47
Edit->Paste V-48
Edit->Rotate V-49
Edit->Schematic / Symbol V-51
File->Create Symbol V-52
File->Exit V-54
File->Expand V-55
File->Help V-56
File->New V-57
File->Open V-58
File->Print V-59
File->Print Window V-60
File->Save V-61
File->Save As V-62
File->Sheet V-63
Misc->Attribute V-65
Misc->Attribute Window V-66
Misc->Check V-67

Misc->Control V-69
Misc->Graphics Control V-70
Misc->HiLite V-71
Misc->Mark V-72
Misc->Pin Numbers V-73
Misc->Query V-74
Misc->Redo V-76
Misc->Reference Designator V-77
Misc->Set Origin V-78
Misc->Statistics V-79
Misc->Undo V-81
Tools->FindItem V-84
Tools->ProbeItem V-85
View->Full Fit V-86
View->Pan/Win V-87
View->ZoomIn V-88
View->ZoomOut V-89

Section VI Interfaces

ASCII Interface ASCII-1
    ASCII Symbol Interface ASCII-1
        Example ASCII Symbol File ASCII-3
    ASCII Schematic Interface ASCII-4

Back Annotation Interface Annotate-1
    Attribute Update File Annotate-1
        Example Attribute Update File Annotate-2

The EDIF Interface EDIF-1
    Translating ECS Symbols To EDIF Files EDIF-1
    Translating EDIF Files to ECS Symbol Files EDIF-1
    EDIF File Format EDIF-2
    Key Points For Creating EDIF Files EDIF-3

Generic Netlists Generic-1
    Processes->Net List By Net (netorder) Generic-1
    Processes->Net List By Pin (pinorder) Generic-2

The PADS Interface PADS-1
    Setting up the ecs.ini File PADS-1
        Adding a Menu Entry to the Hierarchy Navigator PADS-1
        Necessary Attributes PADS-1
    Preparation of Schematics PADS-2
    Error Messages for Pads PADS-3

Hardcopy Output Printing-1
    ECS Configuration Printing Controls Printing-1
    Menu Printing Controls Printing-1
        From the ECS Executive Printing-1
        From the Symbol Editor, Schematic Editor, and Hierarchy Navigator .. Printing-2
    Printing on Sun Workstation Printing-2
        Interface to Other Applications on the Sun Printing-3
    Printing Under DOS Printing-3
        Selecting the Output Device Printing-3
        Scaling the Output Printing-3
        Print Status Messages Printing-4
            Batch Printing Printing-5
        Interface to Other Applications in DOS Printing-5

The Redac Interface RINF-1
    Setting up the ecs.ini File RINF-1
        Adding a Menu Entry to the Hierarchy Navigator RINF-1
        Necessary Attributes RINF-1
    Preparation of Schematics RINF-2
        Setting Up The Power Pins RINF-3
    Error Messages RINF-3

The SILOS Interface SILOS-1
    Installing SILOS SILOS-1
    Setting up the ecs.ini File SILOS-1
        Adding a Menu Entry to the Hierarchy Navigator SILOS-1
        Necessary Attributes SILOS-2
    Creating the Pattern File SILOS-3
    Controlling The Netlist Extraction SILOS-4
    SILOS Format Conversions SILOS-5
    Building SILOS Symbol Libraries SILOS-5
    Examples SILOS-7
        Negative Edge-Triggered J-K Flip Flop SILOS-7
        Four-Bit Inverter SILOS-8
    Error Messages SILOS-8

The SPICE Interface SPICE-1
    Setting up the ecs.ini File SPICE-1
        Adding a Menu Entry to the Hierarchy Navigator SPICE-1
        Necessary Attributes SPICE-2
    Controlling The Netlist Extraction SPICE-4
    SPICE Format Conversions SPICE-5
        Working with the Hierarchical Netlist SPICE-6
        Working with the Flat Netlist SPICE-6
        Preprocessing a Control File SPICE-7
    Attributes Required for SPICE Elements SPICE-7

Examples SPICE-10
Independent Voltage Source SPICE-10
MOSFET..SPICE-10
Error Messages SPICE-11

The Timemill Interface Timemill-1
Installation of Timemill and Pathmill Timemill-1
Setting up the ecs.ini File Timemill-1
Adding a Menu Entry to the Hierarchy Navigator................................Timemill-2
Necessary Attributes...Timemill-2
Preparing Additional Files...Timemill-4
Sample Timemill Stimulus File...Timemill-4
Sample Pathmill Static Setup File...Timemill-4
Controlling The Netlist Extraction...Timemill-5
Invoking the Timemill Simulator (PC only) Timemill-5
Invoking the Pathmill Critical Path Analyzer (PC only)..........................Timemill-7
Timemill Format Conversions..Timemill-8
Building Timemill Symbol Libraries...Timemill-8
Examples Timemill-9
Functional Model for J-K Flip Flop: Timemill-9
Dummy Symbol Using Bus Timemill-9
Error Messages Timemill-10

The Verilog Interface..Verilog-1
Setting up the ecs.ini File Verilog-1
Controls->Verilog Controls Verilog-1
Adding a Menu Entry to the Hierarchy Navigator................................Verilog-2
Necessary Attributes...Verilog-3
Controlling The Netlist Extraction...Verilog-4
Verilog Format Conversions Verilog-4
Building Verilog Symbol Libraries..Verilog-5
Example...Verilog-6
Error Messages Verilog-6

Preface

Schematic drafting systems are used extensively in the electronics industry. They are one of the primary tools used for making drawings that represent the circuits built on printed circuit boards (PCBs) and integrated circuit (IC) chips.

These systems provide two essential services. They provide a working environment where the design engineer can perform and document the creative tasks. The systems can also extract a data description of a drawing and interface to verification tools that are used in the development process.

A variety of schematic systems are available. At one end of the spectrum are the workstation-based products that offer the greatest capability at the highest price. At the other extreme are the low-cost, personal-computer-based systems that offer a few features but are unable to handle the complex design tasks that face the modern design engineer.

The Engineering Capture System (ECS) offers the best of both extremes. The tool suite runs on many platforms, UNIX workstations, IBM compatible PCs, and Macintosh computers. Data can be exchanged among the different platforms. This allows you to perform schematic entry on a low-cost PC and then, if necessary, migrate the design to a workstation that has the capacity to handle the largest IC designs.

Some of the highlights of the ECS are:

- The system maintains connectivity as you draw a schematic. This prevents many common errors because you are not relying on post-process checking programs to find the errors after the design is almost complete.
- Multi-sheet schematics are supported throughout the system. Connectivity is instantly established between sheets. There is no need to perform a special linking process to connect signals that span multiple sheets.
- Hierarchy is an integral part of the system. Top-down and/or Bottom-up methodologies are fully supported. The complete design can be viewed in its hierarchical context with hierarchical names on the nets, pins, and symbol instances.
- Specific instances of circuit elements in the hierarchy can be identified and annotated with attributes that represent the physical implementation of the circuit. These attributes can be viewed and passed on to other design processes.
- The user interface includes an on-line help facility that provides needed information at the touch of a key.
- Operator errors are easily corrected with the use of the **Undo** and **Redo** commands. These commands can be used repetitively to backtrack in the work session.

- Symbols are easily created without stopping the schematic editing session. The symbols are immediately available for placement into the schematic.

The ECS uses the window environments of Sunview, Motif, Microsoft Windows, and MacOS as a foundation. These environments provide a consistent and friendly user interface. Multi-window and multi-task operations allow you to work concurrently on several portions of design. Many of the features common to workstations are supported on MacOS and MS_DOS.

Using This Manual

This manual is divided into ten sections.

Section 1, Before You Begin
Covers material that you need to know before using the ECS. This includes conventions used in the ECS and throughout this manual. Necessary hardware and software configurations are also described. The different window environments used on the Sun workstation, IBM compatible PC, and Macintosh are explained. Installation for each platform is also described.

You should look at the conventions used throughout this manual. If you are familiar with the graphics environment described for your platform, you can skip the graphics environment discussion in this section.

Section 2, System Configuration
Explains how design directories are organized and libraries are accessed. The Preferences Editor is explained in detail. This editor is responsible for setting colors, creating attributes, setting library search paths, setting default display parameters, and much more.

You should read this before attempting a serious design with the ECS. The information about libraries and search paths can have a large impact on the way you structure your design.

Section 3, Getting Started
Provides an overview of the way ECS treats schematics and hierarchy. Understanding the philosophy of the system enables you to use it more effectively.

A brief tutorial guides you through the fundamental operation of the ECS while building a small hierarchical design.

First-time ECS users should read through this section because the fundamental organization of data in the ECS is different from other design systems.

Section 4, Entering Your Design
Gives you a detailed explanation of how to enter your design. It covers the Symbol Editor, Schematic Editor, and Hierarchy Navigator. The differences between entering an IC design and a PCB design are explained in this section. The use of attributes in the ECS is also explained.

Section 5, Command Reference
Contains a detailed alphabetical listing of commands used in the Symbol Editor, Schematic Editor, and the Hierarchy Navigator.

Section 6, Interfaces
Explains several interfaces to the ECS. These include simulation interfaces for Verilog, SILOS, Timemill, SPICE, PADS, and Racal-Redac. Data transfer interfaces are supported for EDIF and ASCII. More interfaces are being added. Explains how to interface any digital simulator to the Hierarchy Navigator and the Waveform Tool. Refer to the *Interfaces* section for the latest list of supported interfaces. Printing is also explained in this section.

Section 7   Design Analysis Tools
Describes the tools that help identify errors in designs. Also shows how to calculate an attributes value based on other attributes. Lists and describes the necessary attributes required to perform the checks. Explains the operation of the Electrical Rules Checker and the Critical Path Viewer. The PCB Packager and derived attributes are described.

Section 8   Waveform Tool
Describes the operation of the Waveform Tool. Shows how to manipulate waveforms and extract information for analysis. Indicates how to use this tool in conjunction with the Hierarchy Navigator and as a standalone tool.

Application Notes
Application notes describing various short cuts, advanced techniques, or other useful information are added to this section.

Appendix
This section will be used for future enhancements, additions or revisions to the manual.

1

Before You Begin

This section contains information describing the different hardware and software environments that support the Engineering Capture System (ECS).

The major topics covered in this section are:

- Conventions used throughout this manual
- Hardware and software configurations of Macintosh computers
- Installation procedures

ECS Conventions

The following conventions are used throughout the ECS system on all platforms and in all the documentation. Platform-dependent information is described in the relevant section for your platform.

- ECS commands always appear in **boldface type**.
- ECS commands appear with the name of the menu where the command is found.

  Draw->Line refers to the **Line** command found under the **Draw** menu
- User-supplied data and file names are usually shown in *italics*.
- File names are shown in lower case.
- Colons indicate a level of hierarchy in the file structure. For example, the file *reg.sch* is located in folder *demo* which is in turn located in the top-level folder *ECS*. This file is referred to as ECS:demo:reg.sch in the Macintosh file structure.
- In order to maintain compatibility among Sun, PC, and Macintosh, file names should be restricted to eight characters plus a three-character file extension. No spaces should be used inside file names or folder names. These restrictions allow files to be passed among the Sun, IBM compatible PC, and Macintosh.

Legal Characters in Net Names

The following characters can be used in net names:

all alphanumeric characters	
single quote, `	
tilda, ~	
exclamation mark, !	
at sign, @	
pound sign, #	
per cent sign, %	
circumflex, ^	
ampersand sign, &	
asterisk, *	
underscore, _	The underscore cannot be used as the second character when following a B as in B_ . The underscore cannot be used in a net name having the following construction, N_nn, where nn is any integer number. These are reserved for use in ECS.
plus sign, +	
double quote, “	
apostrophe, ‘	creates overbar in net name

question mark, ?
forward slash, /

The following characters are not legal characters in net names:

dollar sign, $
minus sign, -
back slash, \
colon, :
semicolon, ;
period, .

Legal Characters in Instance Names

The following characters are legal for use in instance names:

all alphanumeric characters
underscores, _ — The underscore cannot be used for the first or last character of an instance name.

The following characters are not legal for use in instance names:

single quote, `
tilda, ~
exclamation mark, !
at sign, @
pound sign, #
per cent sign, %
circumflex, ^
ampersand sign, &
asterisk, *
underscore, _
plus sign, +
double quote, “
apostrophe, ‘ — creates overbar in instance name
question mark, ?
forward slash, /
dollar sign, $
minus sign, -
back slash, \
colon, :
semicolon, ;
period, .

ECS on Macintosh

This section explains how to install and set up ECS on a Macintosh computer. The Macintosh graphical window environment is also described.

Hardware Configuration

ECS is designed to run under MacOS on Macintosh computers. A minimum configuration is suggested to ensure reasonable performance and compatibility. This configuration includes:

- Any Mac SE or Mac II
- A minimum of 2MB of memory
- A hard disk with at least 10MB available for ECS programs, symbol libraries, and data files
- A color or black-and-white monitor
- A single-button mouse
- A 3-1/2-inch floppy disk drive
- A PostScript printer such as the Apple LaserWriter II if you wish to print

Software Configuration

ECS is designed to run under MacOS. The following software configuration is required.

- System 6.0.1 or later (ECS is system 7 compatible)
- MultiFinder 4.3 or later
- Finder 6.1 or later

Installing ECS

ECS can be installed anywhere in the MacOS file system. All of the files on the distribution media (including the example schematics) take up 10MB. You need to ensure that the target directory has access to at least this much storage space.

Insert the first CAD/CAM floppy disk and follow the instructions in the readme file. The readme file shipped with the software always provides the most up to date installation information. You can install the PCB symbol libraries as described on the first library disk.

Listing of Installed Files and Folders

The folder created during installation contains a number of files and folders. They are grouped below according to their functional use in the ECS.

Core Application Files

ECS Preferences	Configures the ECS design environment
Navigator	Loads complete design hierarchy and facilitates various interfaces
Schematic	Creates schematic files
Symbol	Creates symbol files
Waves	Digital waveform viewing tool (WT module)

Utilities

ASCII_to_SCH	Translates ASCII files into schematic files
ASCII_to_SYM	Translates ASCII files into symbol files
ECS.INI_Maker	Compiles configuration information into binary file
EDIF_to_Symbol	Translates EDIF files to symbol files
SCH_to_ASCII	Translates schematic files to ASCII files
SYM_to_ASCII	Translates symbol files to ASCII files
Symbol_to_EDIF	Translates symbol files to EDIF files
Update_Schem	Updates the time stamps of symbol references in schematic files

Netlist Interfaces and Tools

CheckCkt	Checks electrical loading characteristics (part of DAT module)
CheckPCB	Checks reference designators and symbol types (part of DAT module)
EdifNet	Generates EDIF netlist
FutureNt	Generates Futurenet netlist
HspiceNt	Generates hierarchical spice netlist
Lister	Creates generic parts lists
PCBBack	Performs back annotation for PCB interfaces
PCBNet	Generates Printed Circuit Board netlists
SilosNet	Generates SILOS netlist
SpiceNet	Generates flat spice netlist
TimeNet	Generates Timemill netlist
vhdl	Generates VHDL netlist
vericode	Generates Verilog netlist

Folders

data	Contains ECS help files and simulator configuration files for ECS
examples	Contains ECS demo examples

mod_libs	Contains ECS model libraries
sym_libs	Contains ECS symbol libraries

Miscellaneous Files

ecs.cfg	ECS security file
ecs.ini	ECS configuration setup file

Starting ECS

When the installation is complete, you must run the ECS Preferences Editor to configure the system. A set of system default values is supplied in the file **ecs.ini**. To run the ECS Preferences Editor, use the following procedure.

1. Open the folder containing the ECS executable files.
2. Double-click on the ECS Preferences Editor. An About box is displayed. Clear the About box by clicking on it.
3. Open the file called **ecs.ini**. You can now change the system default values to reflect your particular design environment. If you are loading ECS for the first time, you can proceed without making any changes to the configuration.
4. Saving from the ECS Preferences Editor compiles the configuration information into a form accessible by the ECS application programs.

You can modify the configuration defaults as described later in this section.

The individual programs — such as the Hierarchy Navigator, Schematic Editor, and ASCII to Symbol — can be started by using one of the following methods.

- Double-click on the desired ECS application.
- Schematic, Symbol, and Navigator files can be opened directly by double-clicking on the design file.

Using MacOS

ECS runs under the MacOS window environment. The mouse is used extensively to input information. This makes the user interface easy to learn.

MacOS also allows you to have multiple tasks operational and visible on the screen. These two features make MacOS an ideal environment for use with ECS.

You need to know a few basics about MacOS before you can run ECS. Some basic operating techniques used in MacOS are explained here. Refer to the *Macintosh System Software User's Guide* and other Apple documentation for more details.

What Is a Window?

Each ECS program that you run in the MacOS environment is called an application. When an application is open, it has a window that occupies a rectangular area on the screen. The window can be visible and occupy all or a significant portion of the screen. It can also be hidden by other windows on top of it.

The main areas of the window are shown in Figure 1-1.

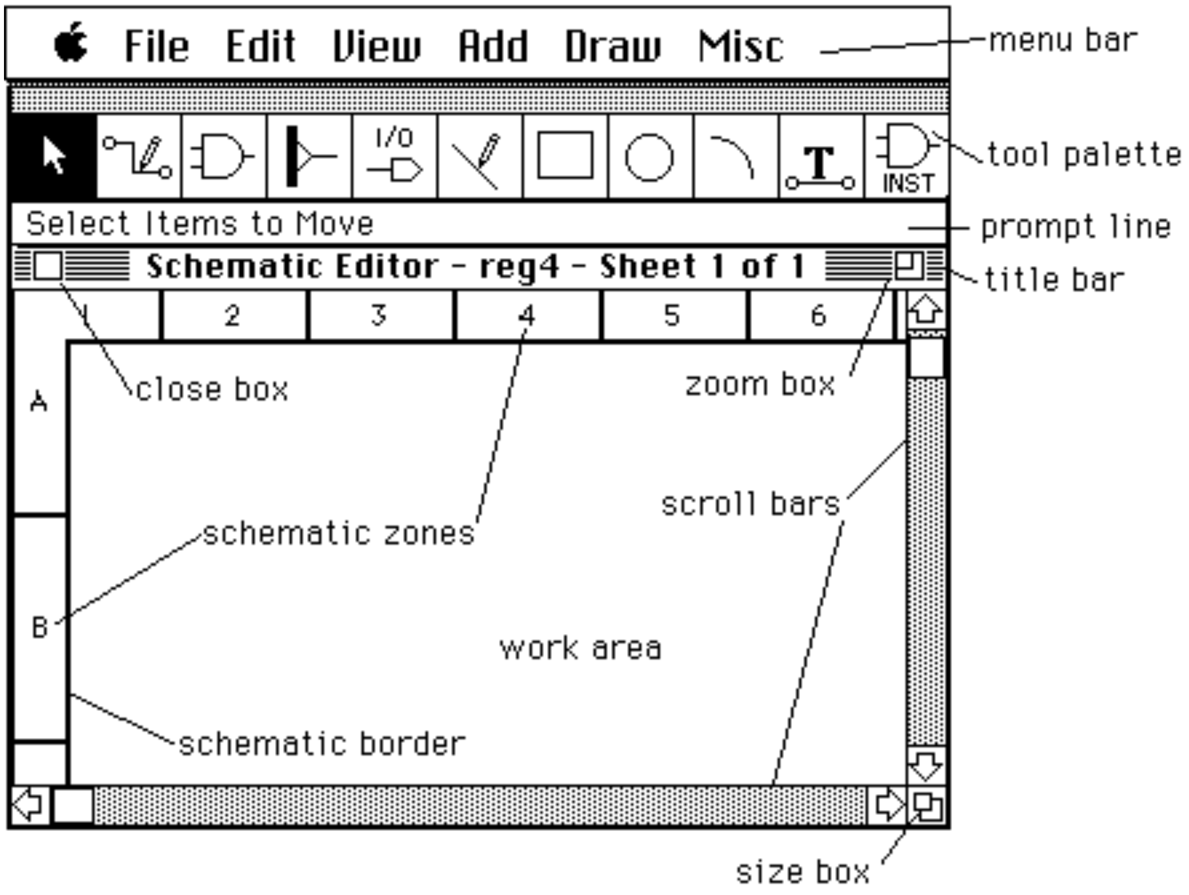


Figure 1-1 Schematic Editor Window

Work Area	Used by the application to display its data.
Tool Palette	Most common functions used during schematic creation are accessed through the tool palette. Clicking on an icon in the palette activates that particular tool.
Title Bar	Displays the name of the application and the file being accessed by the application program. In Figure 1-1, the Schematic Editor is being used to edit the schematic *reg4*. You can position the window on the screen by dragging its Title Bar.
Close Box	Clicking in this box closes the window and design. The application is still active.
Zoom Box	Clicking in the zoom box alternates between a full-screen display and the window size in effect before you clicked on the zoom box.

Menu Bar	Displays a list of the application menu names. Each menu name is a pull-down menu.
Scroll Bars	Move the application's picture within the window to make different portions visible.
Prompt Line	Any prompt or brief instruction for a given command is displayed here.
Schematic Border	The outside edge of the drawing area used by the Schematic Editor.
Schematic Zones	Used to locate various features on the schematic drawing. The location on the schematic is referenced as a grid zone specified by a letter and number combination.
Size Box	Dragging from the size box changes the size of the displayed window.

Using the Mouse

ECS is a graphically oriented system. Both ECS and MacOS are intended to be used with a mouse.

The mouse has one button. The terms applied to using the mouse are:

Point	Move the mouse so the cursor is placed on an object.
Click	Press and release the mouse button without moving the mouse and while pointing to an object.
Double-click	Clicking twice in rapid succession without moving the mouse.
Drag	Press the mouse button and move the mouse to a new location before releasing it. Some form of visual feedback is usually presented.

Selecting

ECS on the Macintosh follows the Macintosh convention of first selecting an object and then operating on the selected object. For example, you select a window and then work on it; you select a command and then execute it. The way you select and the way the selected item is identified vary with what you are doing and the context in which you are working. One of the following methods can be used to select objects:

- Select a single object by pointing at it and clicking the mouse.
- Select an area and all the objects contained in the area by dragging a box around the area to be selected.
- Select multiple objects and areas by holding down the *Shift* key while making your selections.

Selecting the Active Window

When you are running more than one application it is necessary to select one of the applications as the active one. Applications are made active by clicking in the window in which they are resident. This brings the selected window to the front so that everything in the window is visible.

Using Menus

The native menus in MacOS are called pull-down menus. All ECS and MacOS features are accessed using pull-down menus.

Immediately below the title bar is the main menu bar. Each name on the menu bar is a pull-down menu. Point the mouse at a menu name and press the mouse. The menu appears below the name. As you drag the mouse down the menu, the lines of the menu are highlighted to indicate the line at which the mouse is pointing. When you release the button, that command is selected and the menu disappears.

Many of the most common commands can also be accessed from an iconic tool palette below the main menu. This tool palette can be moved to any convenient location on your screen. For small screen displays, it is often convenient to move the tool palette partially off the bottom of the screen so it doesn't take up too much of the available screen.

Many of the commands can also be selected by pressing keys on the keyboard. If a command has a key associated with it, the key name is shown to the right of the command in the menu. The *Command* key is held down while the second key is pressed.

Dialog Boxes

Dialog boxes are used by MacOS and the applications to display messages and to prompt you to enter information. A sample dialog box is shown in Figure 1-2. The following types of controls are used in dialog boxes.

Text Fields	Used by the programs to display messages or status information. You cannot modify these fields.
Edit Boxes	Used to input information into the program. Edit fields are shown as a rectangle with a caption. The insertion point is shown as a blinking vertical line. When you type characters, they appear in the edit box.

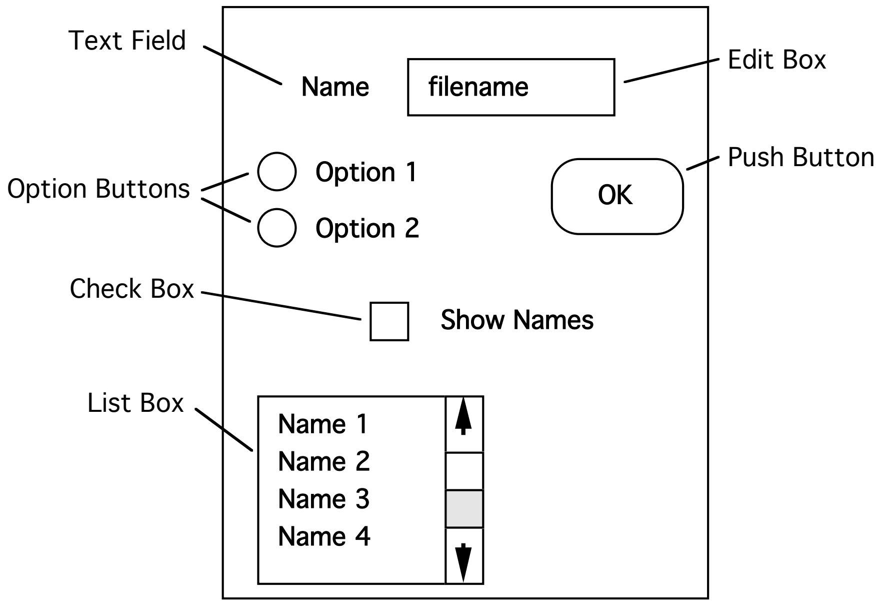


Figure 1-2 Sample Dialog Box

Push Buttons	Signal the program to take an action. Typically, a button marked OK or DONE is provided to instruct the program to terminate and accept the dialog status. When this button is pressed, the program records the state of each of the other controls.
Check Boxes	Indicate and allow changes to status flags. Clicking the mouse on a check box causes it to toggle between on and off.
Option Buttons	Select one of a group of conditions. When the mouse is clicked on one button in the group, that button is turned on and the remaining buttons are turned off.
List Boxes	Display a list of items. An item is selected by clicking the mouse on the line containing the name of the item. A double-click on a list box item is usually interpreted by the program as a selection followed by pressing the OK button. A scroll bar is usually provided so that long lists can be read.
Grayed Controls	Dialog box controls that are inactive. They cannot be selected while they are gray.

Using Scroll Bars

Scroll bars are shown on windows that have data that can extend beyond the window. Figure 1-3 shows a typical scroll bar.

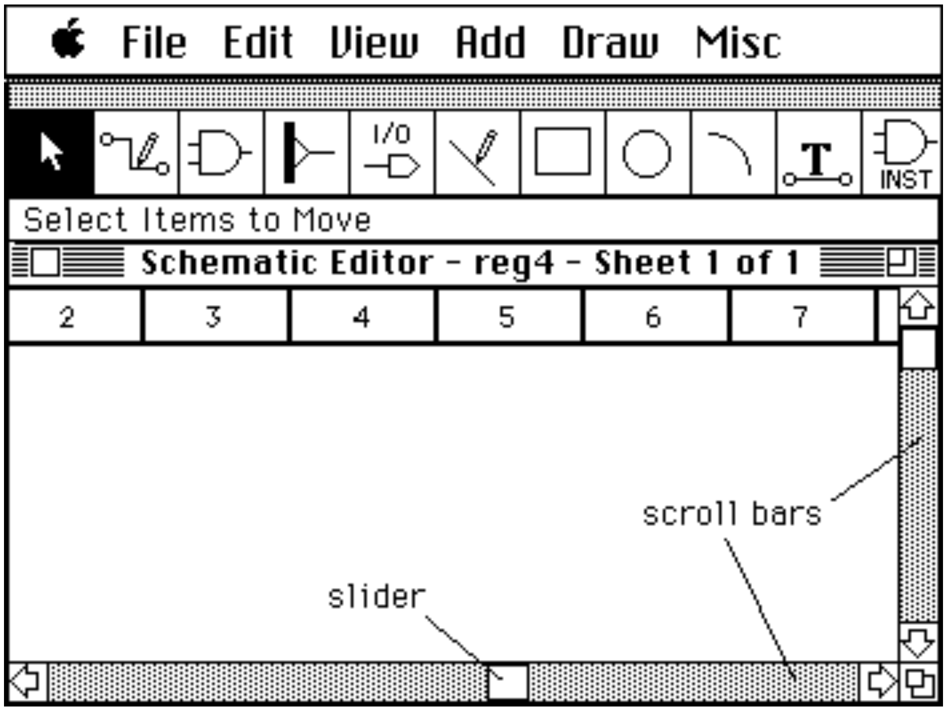

File Edit View Add Draw Misc
I/O
INST
Select Items to Move
Schematic Editor - reg4 - Sheet 1 of 1
2 3 4 5 6 7
scroll bars
slider

Figure 1-3 Sample Scroll Bar

The scroll bar has an arrow at each end and a scroll box located in the bar. You can move the window over the data with the mouse with the following operations:

- Click the mouse on the arrow to move one line at a time
- Click the mouse between the arrow and the scroll box to move a distance equal to the size of the window
- Drag the scroll box to move to the approximate location in the picture.

Arranging Windows

To move windows to different locations on the screen:

1. Point to the title bar and press the mouse.
2. Drag the title bar to the desired location and release the button. The window then moves to the selected location.

To change the window's size:

1. Point to the size box and press the mouse.
2. Drag the mouse; the corner of the window will follow. Release the mouse when the window is the required size.

Quitting an Application

Quitting an application releases the screen and any memory used by the application. To quit an application, select **Quit** from the **File** menu.

When you quit, the application checks for any open files. If you have changed but not saved the work file, the application prompts you to save the file before quitting.

The ECS Adaptation of MacOS

ECS uses most of the conventions that are common to applications running under the MacOS environment. There are a few deviations and extensions to the normal protocol that are made to provide an easier user interface. These are described below.

The Mouse and Escape Key

The mouse is used by ECS for selecting commands, manipulating dialog boxes, and selecting locations within the window. ECS also uses the *Escape* key to:

- Change to the alternate mode of a command if one is defined
- Restart the command

Selection

Normal Macintosh protocol for executing commands is to select the object to be operated on and then select the required command. For example, the sequence for deleting a line is:

1. Select the line.
2. Execute the **Delete** command.

ECS also provides several ways to select objects in the **Edit** commands.

- Select a single object by pointing at it and clicking the mouse.
- Select an area and all the objects contained in the area by dragging a box around the area to be selected.
- Select multiple objects and areas by holding down the *Shift* key while making your selections.

Undo and Redo

ECS provides a continuous **Undo** capability that allows the reversal of all the commands that were executed since the last time the data was saved. In addition, the **Redo** command allows you to recover from accidental Undoing.

Prompting and Error Messages

ECS prompts you for the next action. This information is displayed in a small prompt line that is initially drawn at the top of the screen as part of the tool palette. You can move the tool palette anywhere on the screen.

The prompt line of the tool palette is also used for input of textual information, such as the names of symbols and signals in the Schematic Editor.

The prompt line of the tool palette is used to report minor errors that prevent a command from completing its function. Major errors are reported in pop-up message boxes.

2

System Configuration

This section explains how to configure the Engineering Capture System (ECS) for your design environment. The major topics covered in this section are:

- Hierarchical file structures
- Libraries
- Graphic display control
- ECS Preferences Editor
- Setting the environment
- Creating attributes
- The color palette
- Adding **Tools** and **Processes** to the Hierarchy Navigator menu

Libraries and Directory Structures

Libraries are collections of symbols, models, or hierarchical design blocks that can be accessed by the schematics of many different designs. Having a central location where circuit elements are stored makes the organization of the design much simpler than would be the case if these circuit elements were scattered over many different folders. It also makes it much easier to ensure that all symbols and models are kept updated.

ECS is designed to work in conjunction with the directory structures found in UNIX, DOS and MacOS. These directory structures help to organize the ECS software into logical divisions. They also allow you to subdivide your project into pieces that are easier to manage. You should become familiar with the directory structures used on your system.

Program Directories

All of the programs and related software that ECS uses are typically located in a master folder called *ECS*. The ECS software can reside anywhere on the system. You must make sure that the various files under the *ECS* directory have the proper permissions to be accessed by everyone using the tools. Some of the sub-folders under ECS are shown in Figure 2-1.

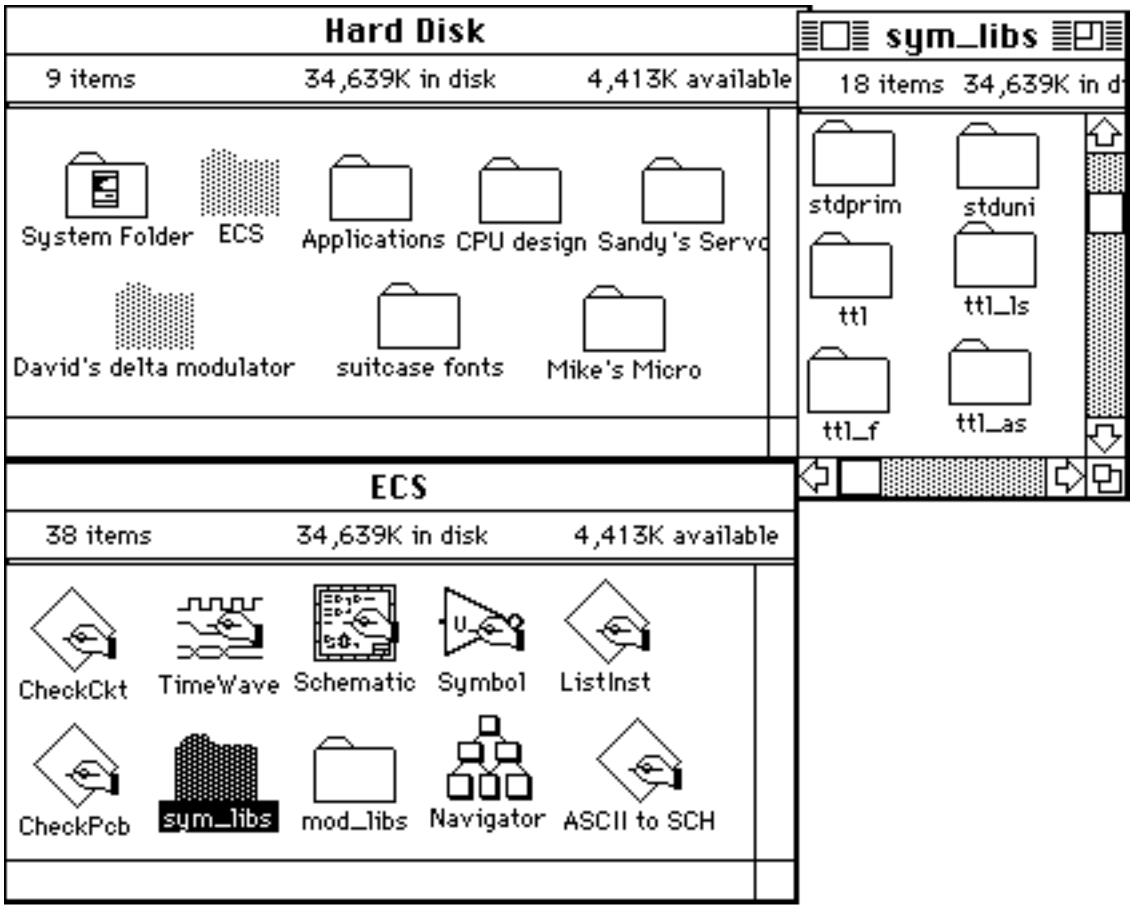


Figure 2-1 Typical ECS Program Directory Structure

Macintosh Computers

The various files and folders found in the master ecs folder on the Macintosh are:

ecs	The ecs folder contains the ECS applications. These include *Navigator*, for running the hierarchical navigator; *Schematic*, for running the Schematic Editor; and about 20 other applications. The ecs license file is located in this folder. The **ecs.ini** file is also located in this folder. It configures ECS and allows you to customize colors, default symbol types, additional simulator or netlist interfaces, and many other elements in the ECS environment.
data	Under the data folder, you find ECS help files, and the simulator configuration files.
sym_libs	Contains the default symbols supplied with the ECS software.
mod_libs	Contains the default model libraries supplied with the ECS software.
demo	Contains a number of example designs using the ECS system.

User Folders

The system expects that the data created for a design are maintained in a single folder. Several designs can be maintained on a system by creating different folders for each design. The actual names and locations of these folders are optional.

The current folder is considered the default folder for a design session. This folder is referred to as the user or local folder.

A user folder can contain project directories. This allows a design to be stored in a hierarchical fashion.

Library Directories

Libraries are used to store fundamental building blocks that are used many times in different designs. Libraries are typically organized so that a single library contains related items. For example, a symbol library can consist of symbols for the standard TTL devices. Another library can contain symbols for certain gate array primitives.

Libraries are typically maintained by a system administrator, and the individuals who use the libraries are not allowed to modify them. This enforces consistency in the design environment.

The ECS software interfaces with three different types of libraries: symbol libraries, project directories, and model libraries. These are described below.

Project Directories	Contain partial designs in a hierarchical fashion. That is, they store a high-level symbol, the underlying schematic, symbols beneath that, and potentially many more layers of hierarchy. Project directories store design blocks or sub-circuits that are used in many places. A register bank or a 32-bit counter are two examples of sub-circuits that could be stored in project directories.
Model Libraries	Contain schematics that represent higher-level primitives. By including the model library on the library search path, a lower-level or more detailed description of the circuit is obtained. For example, a schematic containing logic gates is drawn and simulated using a gate-level simulator. A model library exists containing the transistor-level schematics of all the logic gates. A netlist for a switch-level simulation is easily obtained by adding the model library to the library search path and then netlisting. You can represent multiple levels of hierarchy using model libraries.
Symbol Libraries	Contain the primitive symbols for IC and PCB designs. Examples of typical symbols are PMOS and NMOS transistors, AND gates, NOR gates, and a 74ALS193 counter

chip. Symbols are used throughout the hierarchy and at the primitive level in a design.

Project directories support top-down design because they keep the lower levels hidden from view, allowing you to think in terms of high-level blocks. The lower levels of hierarchy are available when you need to look at them.

One application of model directories is to allow you to complete a design at the logic gate level and then compare the performance of different technologies. One model library is made for each different technology. To get a circuit description containing only logic gates, leave the model libraries off the library search path. To check the performance of technology *A*, add model library *A* to the library search path. The logic gates are expanded in terms of the technology used in library *A*. Other technologies are checked by changing the model library search path.

Libraries are specified in the ECS Preferences Editor. You can specify many different libraries of each type and you can control which ones are on the library search path by using the ECS Preferences Editor. To modify the library search path, refer to the *Search Paths* section of the ECS Preferences Editor description later in this section.

Library Searching

ECS searches for symbols in a prescribed order. ECS uses the first symbol of the required name that is found. For example, a symbol *myname* can exist in a project directory and another version of *myname* can exist in a symbol library. Because project directories are searched before symbol libraries, the version of *myname* from the project directory is used.

The symbol search order is as follows.

1. Local directory.
2. The project directories specified in the ECS Preferences Editor are searched. Project directories closest to the top of the list box in the ECS Preferences Editor are searched first.
3. The symbol libraries specified in the ECS Preferences Editor are searched. Symbol libraries closest to the top of the list box in the ECS Preferences Editor are searched first.

If you try to open a schematic and the symbol search paths are not set correctly, the schematic editor may not be able to locate all of the symbols. If this happens, the Schematic Editor opens the schematic and you can see blanks on the schematic where the missing symbols should be placed. This allows you to see which symbols are missing. The schematic opens with the name *untitled* so if you accidently save, you do not destroy the references to the original symbols which are missing.

ECS also searches for schematics in a prescribed order. ECS uses the first schematic of the required name that is found. The schematic search order is as follows.

1. Local directory.

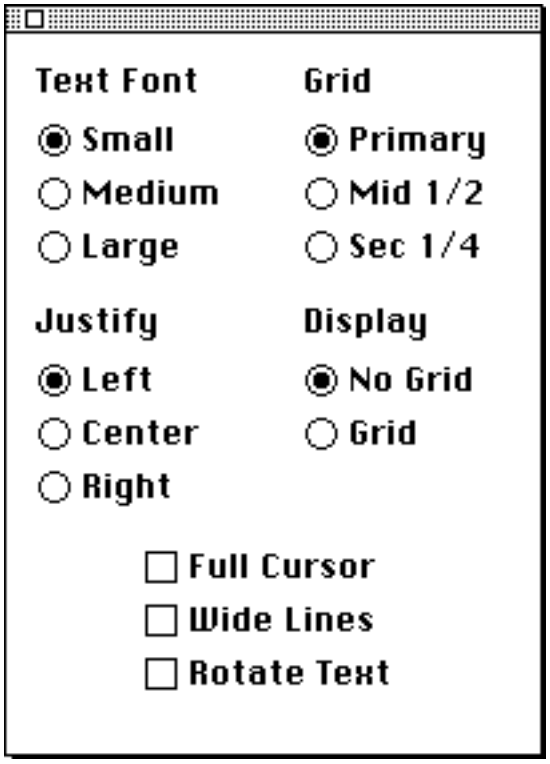
2. The project directories specified in the ECS Preferences Editor are searched. Project directories closest to the top of the list box in the ECS Preferences Editor are searched first.
3. The model libraries specified in the ECS Preferences Editor are searched. Model libraries closest to the top of the list box in the ECS Preferences Editor are searched first.

The following shows a typical setup of the symbol search path from the **Search Paths->Symbol Library** section for the Macintosh. Note the use of the leading colon in the search path. This indicates a relative path from the same folder containing the **ecs.ini** file. Note also that spaces are not allowed in any names found in search paths. This means that disk drive names, folder names and file names should not use spaces.

```
MacintoshHD:ECS:sym_libs:std
MacintoshHD:ECS:sym_libs:misc
:sym_libs:ttl
:sym_libs:ttl_ls
:sym_libs:ttl_als
:sym_libs:ttl_as
```

Graphic Options

The **Graphic Control** command sets the parameters that control the **Draw** commands in the Symbol and Schematic Editors. When you invoke the command, a dialog box is displayed that offers control over the parameters noted below in Figure 2-2.



Text Font

[✓ Small] Medium Large

Grid

[✓ Primary] Mid 1/2 Sec 1/4

Justify

[✓ Left] Center Right

Display

[✓ No Grid] Grid

☐ Full Cursor
☐ Wide Lines
☐ Rotate Text

Figure 2-2 Dialog Box for Graphic Options

Text Font	There are three text sizes that are used in the system for drawing fixed graphic text and symbol attribute windows. This control selects the size to be used. The available choices are small, medium, and large.
Justify	Text can be left-justified, center-justified, or right-justified. This parameter applies to fixed graphic text and symbol attribute windows. Text is always center-justified vertically.
Grid	All electrical elements of the schematics and symbols are drawn on the primary grid as specified in the ECS Preferences Editor. Graphic elements and text can optionally use a finer grid for their locations. This control allows the primary grid to be subdivided by 2 or 4 to provide this higher resolution.
Display	This controls the primary grid display. If you display the grid, it appears as an array of dots, with one dot at every grid intersection. Every 10th grid point is larger to aid in counting grids. As you

	zoom out in scale and the grids get closer together, some grid dots may not be displayed
Full Cursor	This control selects between the normal cursor and the full-screen cursor. The system has a choice of a small *plus* cursor (the default) and a full-screen cursor. The full-screen cursor makes it easier to align objects.
Wide Lines	This control selects between normal and heavy line weights for drawing lines and rectangles in the symbol and schematic. The heavy lines have the same weight as buses on schematics.
Vertical Text	Text and Attribute windows can be entered as horizontal or vertical text. The Vertical Text control selects the orientation.

The ECS Preferences Editor

ECS utilizes the ECS Preferences Editor to provide configuration control and to define the default values of many parameters. The following list describes many of the parameters that can be changed using the ECS Preferences Editor.

- Creating attributes
- Color assignment
- IC or PCB application mode
- Display parameters such as *Show Border* and *Show Pin Dots*
- Default symbol type
- Naming conventions
- Global nets
- **Tools** and **Processes** menu entries
- Schematic sheet sizes and layout parameters
- Setting library search paths

Using the ECS Preferences Editor

This section is used to define a variety of parameters for control of the system modules. The various parameters are explained below. They are organized alphabetically according to their menu entry. The top-level menu is used as the first key in the alphabetical organization, resulting in parameters controlling related functions being grouped together.A number of conventions are used in the following description.

- The menus in the ECS Preferences Editor are referenced in bold type with the top-level menu followed by **->** and the lower menu. For example, the **Sheet Layout** menu is located under the **Controls** menu in the ECS Preferences Editor. It is referenced as **Controls->Sheet Layout**.
- Most parameters in the ECS Preferences Editor are modified using either edit fields, check boxes, or pop-up menus.

Starting the ECS Preferences Editor

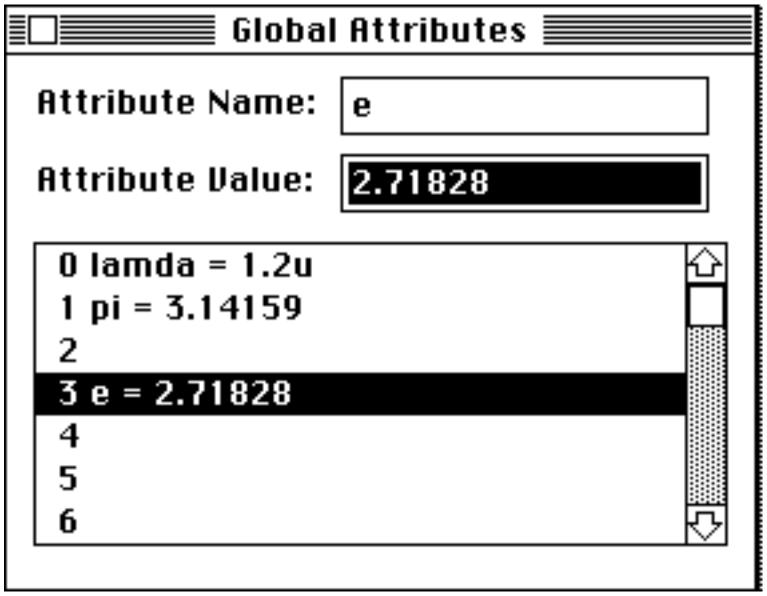
The configuration information is stored in a text file (usually called **ecs.ini**). You can open this configuration file using the ECS Preferences Editor and make any necessary changes. When the ECS Preferences Editor saves the text file, it creates a binary image file (**ECS_init**) that is stored in the system folder. All of the ECS programs access this binary file for the configuration information. Whenever you make changes to the configuration information using the ECS Preferences Editor, you must restart any ECS applications in order to use the new configuration information.

The rest of this section is organized into ECS Preferences Editor menu groups. All the parameters under one menu entry are described together.

Attributes->Global Attributes

This section of the ECS Preferences Editor defines the global attributes used in schematics. To define a global attribute, use the following procedure while referring to Figure 2-3.

1. Select the desired global attribute number from the list box.
2. Type the name of the global attribute in the Attribute Name edit field.
3. Type the value of the global attribute in the Attribute Value edit field.



Attribute Name: e

Attribute Value: 2.71828

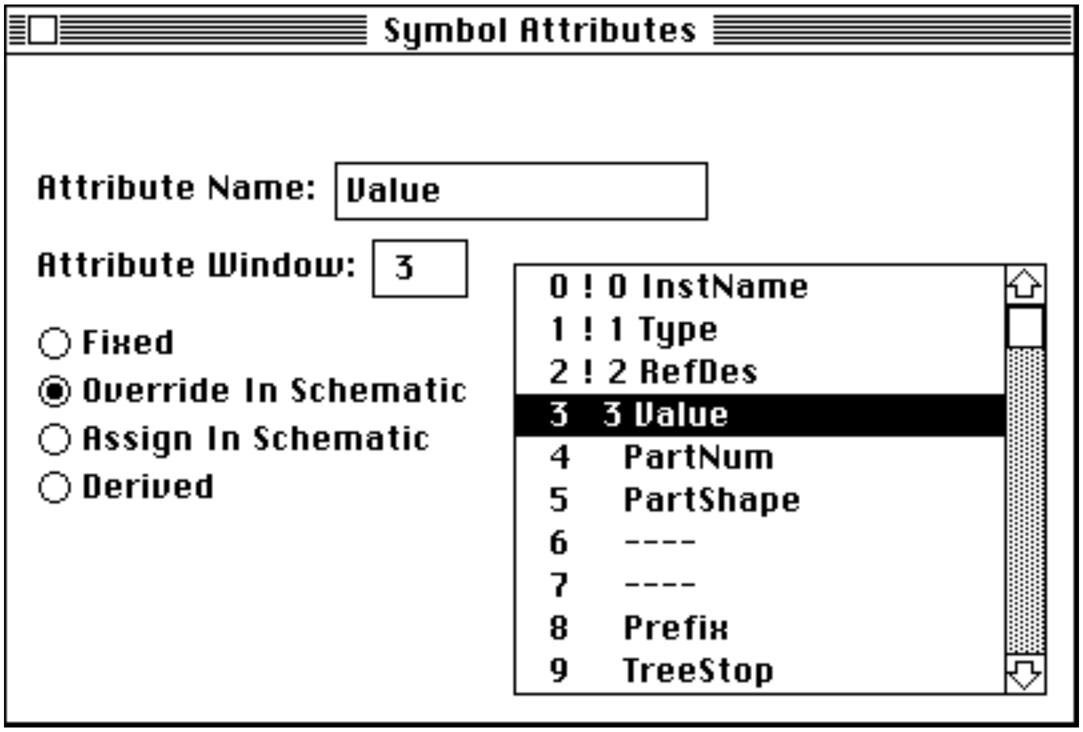
```
0 lamda = 1.2u
1 pi = 3.14159
2
3 e = 2.71828
4
5
6
```

Figure 2-3 Global Attribute Editor

Global attributes can also be edited in the Hierarchy Navigator using the **Misc->Edit Constants** command.

Attributes->Net, Pin, Symbol Attributes

This section of the ECS Preferences Editor defines the attributes used in schematics. Net, pin, and symbol attributes are all defined in a similar fashion using the Attribute Editor shown in Figure 2-4. The description that follows applies to pin, net, and symbol attributes unless otherwise noted.



Attribute Name: Value

Attribute Window: 3

( ) Fixed
(●) Override In Schematic
( ) Assign In Schematic
( ) Derived

```
0 ! 0 InstName
1 ! 1 Type
2 ! 2 RefDes
3   3 Value
4     PartNum
5     PartShape
6     ----
7     ----
8     Prefix
9     TreeStop
```

Figure 2-4 Symbol Attribute Editor

The list box on the right of Figure 2-4 contains all symbol attributes currently defined in the ECS Preferences Editor. The fields on each line of the list box are:

- The first field is the attribute number. This number is how the attribute is represented in the database and is how ECS accesses them.

  There are usually some free attribute numbers in the list box. They are shown with dashes in the name field. These can be used to create new attributes. Attributes 0 through 99 have their meanings defined by the system, while attributes 100 through 199 are user defined.
- The second field is for an optional attribute modifier. The default condition allows you to define an attribute in the Symbol Editor and override it in the Schematic Editor and the Navigator. This condition is indicated with a blank in field two. You can modify this field for symbol attributes and pin attributes, but not for net attributes.

  Additional modifiers and their meanings are:

!	Edit the attribute value only with special ECS commands such as **Add->Instance Name**. These are special attributes that ECS uses internally.
-	Edit the attribute value only in the Symbol Editor; it cannot be overridden in the Schematic Editor or Hierarchy Navigator.
+	Edit the attribute value in the Symbol Editor, Schematic Editor, or the Hierarchy Navigator. If you edit the value in more than one place, values placed in the Schematic Editor override those placed in the

Symbol Editor and values placed in the Hierarchy Navigator override all others.

* Specifies a derived attribute and allows you to edit the attribute value in the Symbol Editor and override in the Schematic Editor or the Hierarchy Navigator.

- The third field is optional and specifies the attribute window. Attribute windows are applicable to symbol attributes only. The attribute is displayed on the symbol inside this attribute window. Attribute windows are described in the *Entering Your Design* section under Attributes.

  If two attributes both have the same attribute window defined, the attribute with the lowest attribute number is displayed.
- The last field is the attribute name.

Modification of Symbol Attributes

To modify a symbol attribute, follow the procedure described.

1. Select the attribute you wish to modify from the list box. You can select any line in the list box even if the attribute number is the only field filled. An edit field is displayed for entering the attribute name.
2. Type the attribute name in the edit field. An edit field for the attribute window number is displayed and option buttons allow you to specify the attribute modifier.
3. Type the attribute window number if you want the attribute displayed on the schematic.
4. Check the required attribute modifier. *Override on Schematic* is the default modifier. If this is selected, the modifier field is left blank in the list box.

Modification of Pin Attributes

To modify a pin attribute, follow the procedure described.

1. Select the attribute you wish to modify from the list box. You can select any line in the list box even if the attribute number is the only field filled. An edit field is displayed for entering the attribute name.
2. Type the attribute name in the edit field. Radio buttons for selecting the attribute modifier are displayed.
3. Check the required attribute modifier. *Override on Schematic* is the default modifier. If this is selected, the modifier field is left blank in the list box.

Modification of Net Attribute

To modify a net attribute, follow the procedure described below.

1. Select the attribute you wish to modify from the list box. You can select any line in the list box even if the attribute number is the only field filled. An edit field is displayed for entering the attribute name.
2. Type the attribute name in the edit field.

Example Attributes

Tables 2-1, 2-2, and 2-3 show example attribute definitions for net, pin, and symbol attributes. Attribute modifiers, numbers, and windows are shown where appropriate.

Refer to the discussion of attributes in the *Entering Your Design* section for a complete description of how to use attributes.

Attr Num	Attr Mod	Attribute Name	Comments
0	!	NetName	Net name
3		Cap	
5		Length	
8		Width	
10		VeriType	
30		VHDLNetType	
31		VHDLBusType	

Table 2-1 Standard Net Attributes

Attr Num	Attr Mod	Attribute Name	Comments
0	-	PinName	Pin name
1	-	Polarity	IN, OUT, BIDIR
2		Fanin	Dimensionless number for IC loads
3		Fanout	Dimensionless number for IC drive
4	-	PinNumber	Used in PCB's for Gate or Component pin numbers, represents physical pin connection
5		WireOr	Tristate, Opencollector, or Yes
6	-	PinGroup	Used in PCB design, indicates can swap pins
7		LoadLow	Current load in low state (μA)
8		DriveLow	Current drive in low state (μA)
9		OpenOK	OK to be unconnected pin
10	-	SilosName	Identifies pins in models
11		SilosLoad	Numeric load factor for load calculation
14	-	VeriName	Alternate pin name or order for pins
15		LoadHigh	Current load in high state (μA)
16		DriveHigh	Current drive in high state (μA)
18	-	TimilName	Alternate pin name or order for pins
20		LoadCap	Capacitive load (pf)
21		DriveCap	Capacitive drive (pf)
22	+	KCL	Drive factor for simulation models
23	+	Pin2Pin	Pin to pin delay for simulation models
24	+	DelayBack	Back annotated delay for simulation models
25	+	ChkPulseW	Check for pulse width violation on this pin
26	+	ChkHold	Check for hold time violation on this pin
27	+	ChkSetup	Check for setup time violation on this pin
30		VHDLPinType	Port type if scalar port in VHDL
31		VHDLBusPinType	Port type if vector port in VHDL
32		VHDLDefValue	Default value for Port
33		VHDLNetConv	Type conversion function for port
34		VHDLBusConv	Type conversion function for port
35		VHDLPinUse	Used for port type of BUFFER in VHDL
38		SpiceOrder	Integer that forces order of subcircuit pins
40		HiLoPinName	Alternate pin name used for HiLo simulations
50		XSimPinName	Pin name used for X-Sim primitives
51		PinNegation	Apply negation to this pin
90	-	BusPin_A	First set of pins attached to bus pin
91	-	BusPin_B	Second set of pins attached to bus pin
92	-	BusPin_C	Third set of pins attached to bus pin
93	-	BusPin_D	Fourth set of pins attached to bus pin
94	-	BusPin_E	Fifth set of pins attached to bus pin
95	-	BusPin_F	Sixth set of pins attached to bus pin
96	-	BusPin_G	Seventh set of pins attached to bus pin
97	-	BusPin_H	Eighth set of pins attached to bus pin

Table 2-2 Standard Pin Attributes

A typical set of symbol attribute names, numbers, and window assignments is shown in Table 2-3 (appearing on the next two pages).

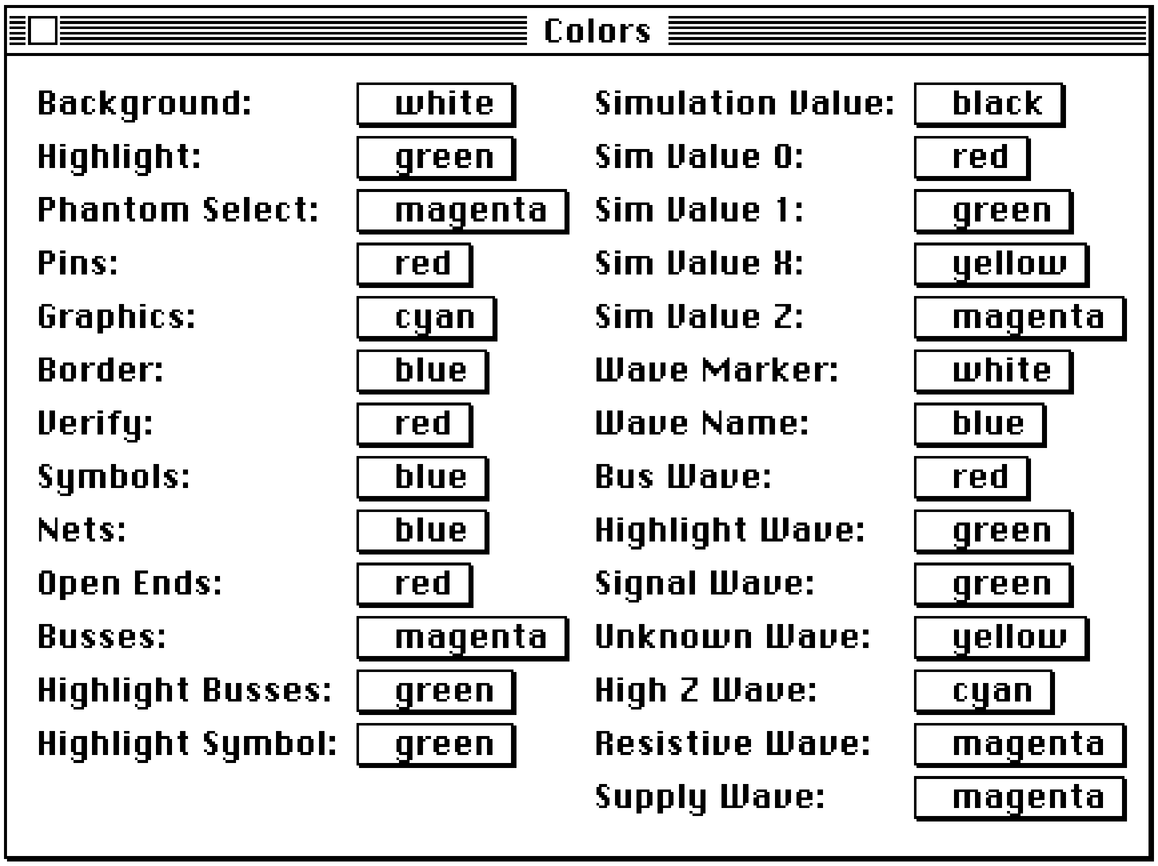
Attr Num	Attr Mod	Attr Win	Attribute Name	Comments
0	!	0	InstName	Instance Name
1	!	1	Type	Symbol Name
2	!	2	RefDes	Reference designator for PCBs
3		3	Value	General value parameter
4			PartNum	PCB part number
5			PartShape	Footprint of PCB part
8			Prefix	SPICE element prefix (Q, M, R…)
9			TreeStop	Used to split large designs
10	-		SilosModel	Primitive model name for Silos
11			SilosTimes	Delay specifier for Silos primitives
15	-		TegasModel	Tegas model parameter
17	-		XSimModel	X-Sim primitive model name
18	+		BodyDelay	Body delay parameter for simulation
20	-		VeriModel	Primitive or model name for Verilog
21			VeriTimes	Delay specification for Verilog
22			VeriStrnth	Strength specification for Verilog
25	-		TimilModel	Primitive or Model name for Timemill
26			TimilExtra	Extra parameter for Timemill
34			Impedance	SPICE parameter
35			Width	SPICE transistor width
36			Length	SPICE transistor length
37			Multi	Multiplication factor for SPICE
38			SpiceModel	Model card for SPICE
39			SpiceLine	Model parameters for SPICE
40			SpiceLine2	Additional SPICE model parameters
41	*		AreaS	Area of source for SPICE
42	*		AreaD	Area of drain for SPICE
43	*		PeriS	Perimeter of source for SPICE
44	*		PeriD	Perimeter of drain for SPICE
45			NRS	squares of source diffusion, SPICE
46			NRD	squares of drain diffusion, SPICE
47			DefSub	Substrate node
60	-		GND	Power connection attribute for PCBs, typ GND
61	-		VDD	Power connection attribute for PCBs, typ VDD
62	-		VCC	Power connection attribute for PCBs, typ VCC
63	-		PCBGlobal3	Power connection attribute for PCBs
64	-		PCBGlobal4	Power connection attribute for PCBs
65	-		PCBGlobal5	Power connection attribute for PCBs
66	-		PCBGlobal6	Power connection attribute for PCBs
67	-		PCBGlobal7	Power connection attribute for PCBs

68	-		PCBGlobal8	Power connection attribute for PCBs
69	-		PCBGlobal9	Power connection attribute for PCBs
70	-		HiloModel	Specifies model for HILO primitives
71			HiloTimes	Specifies HILO delay times
72			HiloStrength	Specifies HILO driving strength
73			HiloParam	
74			HiloParamValue	
75			HiloDelayScale	
76			HiloVisibility	
78			VHDLConfig	
79			VHDLUseLib	
80			VHDLModel	
81			VHDL1	User definable for VHDL
82			VHDL2	User definable for VHDL
83			VHDL3	User definable for VHDL
84			VHDL4	User definable for VHDL
85			VHDL5	User definable for VHDL
86			VHDL6	User definable for VHDL
87			VHDL7	User definable for VHDL
88			VHDL8	User definable for VHDL
89			VHDL9	User definable for VHDL
99			EllaType	

Table 2-3 Standard Symbol Attributes (see previous page)

Controls->Colors

The various colors available on your system are assigned to different functions within ECS using the Color Assignment portion of the ECS Preferences Editor. You choose the color to be assigned from a pop-up list box. Figure 2-5 shows the Color Assignment Editor.



Colors

Background:	white	Simulation Value:	black
Highlight:	green	Sim Value 0:	red
Phantom Select:	magenta	Sim Value 1:	green
Pins:	red	Sim Value X:	yellow
Graphics:	cyan	Sim Value Z:	magenta
Border:	blue	Wave Marker:	white
Verify:	red	Wave Name:	blue
Symbols:	blue	Bus Wave:	red
Nets:	blue	Highlight Wave:	green
Open Ends:	red	Signal Wave:	green
Busses:	magenta	Unknown Wave:	yellow
Highlight Busses:	green	High Z Wave:	cyan
Highlight Symbol:	green	Resistive Wave:	magenta
		Supply Wave:	magenta

Figure 2-5 Color Assignment Editor

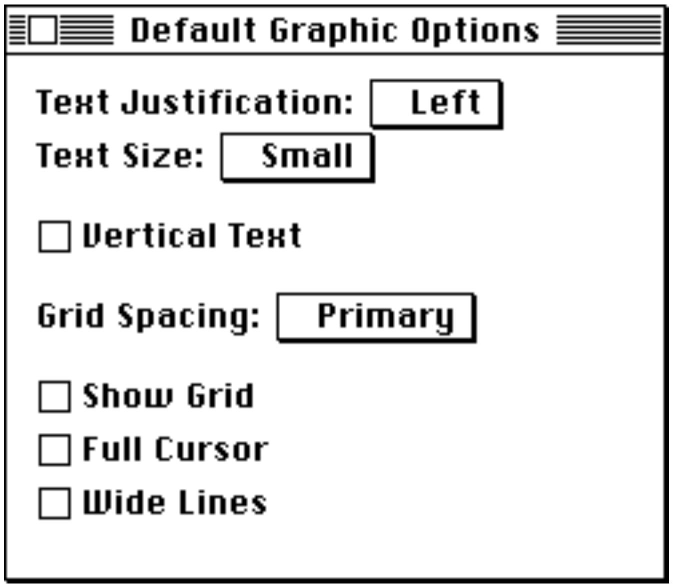
The different display functions are explained below.

Highlight	Highlighting nets
Phantom	Highlight group editing functions
Pins	Symbol pin dots
Graphics	Lines, arcs, text ...
Border	Sheet border
Verify	Indicates connection to net
Symbols	Symbol body
Nets	Wires and net names
OpenEnds	Dot on hanging wire or isolated net name
Buses	Buses and bus names
HighlightBus	Highlighting buses
HighlightSymbol	Highlighting symbols
SimVal	Textual simulation values shown on schematic
SimVal0	Colored square shown on schematic representing logic 0 values
SimVal1	Colored square shown on schematic representing logic 1 values
SimValX	Colored square shown on schematic representing unknown values

SimValZ	Colored square shown on schematic representing high impedance values
WaveMarker	Sets color of Marker in Waveform Tool
HighlightWave	Highlights the selected waveform in Waveform Tool
WaveName	Waveform name in Waveform Tool
BusWaveform	Bus waveforms displayed in Waveform Tool
SignalWave	Signal waveforms displayed in Waveform Tool
UnknownWave	Buses or signals displayed in Waveform Tool with unknown value
HighZWave	Buses or signals displayed in Waveform Tool with high impedance values
ResistiveWave	Buses or signals displayed in Waveform Tool with resistive impedance values
SupplyWave	Color of waveform with strength *Supply*.

Controls->Default Graphic Options

This section contains a variety of default graphic options. Figure 2-6 shows the Default Graphic Options dialog box.


Default Graphic Options

Text Justification: Left

Text Size: Small

☐ Vertical Text

Grid Spacing: Primary

☐ Show Grid

☐ Full Cursor

☐ Wide Lines

Figure 2-6 Default Graphic Option Dialog Box

Text Size	sets the default text size. Choices are small, medium and large.
Text Justification	sets the default text justification. Choices are left, center, and right.
DefaultTextRot	sets the default text rotation
Full Cursor	defines whether full cursor is the default
Grid Spacing	sets the default grid spacing

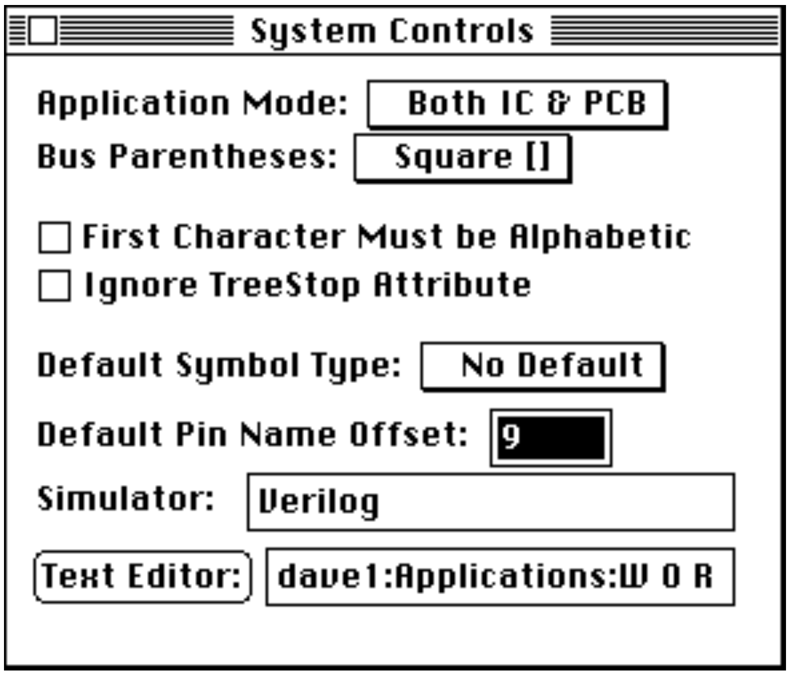
Show Grid	if checked, the grid is displayed by default on schematics
Wide Lines	if checked, the lines drawn on schematics are wide by default

Controls->System Controls

This section contains a variety of parameters used to control system function. Figure 2-7 shows the System Controls dialog box.

In addition to these parameters available through the dialog box, there are two parameters that can only be set manually in the Preferences file. These are:

NetNameCaps	if set =yes, net names are forced to uppercase. This is default if this parameter is not specified. If set = no, net names retain the case you typed them in with.
AttributeCaps	if set =yes, attribute values are coerced to be uppercase. If set =no, attribute values retain the case you type them in with.



Application Mode: Both IC & PCB
Bus Parentheses: Square []
☐ First Character Must be Alphabetic
☐ Ignore TreeStop Attribute
Default Symbol Type: No Default
Default Pin Name Offset: 9
Simulator: Verilog
Text Editor: dave1:Applications:W 0 R

Figure 2-7 System Controls Dialog Box

Application Mode
This parameter configures ECS for IC design or PCB design, or both. The choices are:

IC	Provides instance name support
PC	Provides support for pin numbers and reference designators
BOTH	Provides support for instance names, pin numbers, and reference designators

Further discussion about the differences between IC and PCB design appear in the section, *Differences Between IC and PCB Design*.

Bus Parentheses

There are several conventions that are commonly used to apply indices to ordered bits in a bus. For example, one common way to name bus elements is A[2], A[1], A[0]. The *Bus Parentheses* parameter allows you to specify one of:

- [ ]   square brackets
- ( )   round parentheses
- { }   braces

First Character Must Be Alphabetic

Many systems have the requirement that the first character of an instance or a net name must be an alphabetic character. ECS checks this constraint if this parameter is specified is checked. Numbers can be used as first characters if this is unchecked.

Ignore TreeStop Attribute

This parameter causes the TreeStop attribute to be ignored when the hierarchy is built.

When working with large designs on the PC, a design hierarchy may have to be split into sections by the insertion of TreeStop attributes. When a design is transferred to a Sun workstation, there is no longer a need to have the design split.

Default Symbol Type

This parameter sets the default symbol type used in the Symbol Editor when it is started with a new drawing. This parameter can take on a value corresponding to each of the symbol types listed below and to the value *none*. This parameter is primarily used during library creation.

Component	Primitive in PCB designs representing complete packaged device
Gate	Primitive in PCB designs representing a fraction of complete device (for example, one of four NAND gates in a LS7400)
Cell	Primitive in IC design
Block	Represents hierarchy
Pin	Represents physical pins on a PCB
Graphic	Used for non-electrical information, such as tables and notes
Master	Used for title blocks and other non-electrical information positioned with respect to a corner of a drawing

If the parameter is set to *no default*, the Symbol Editor queries you for the required symbol type when you start a new drawing.

Refer to the *Symbol Editor* section for more details on the different symbol types.

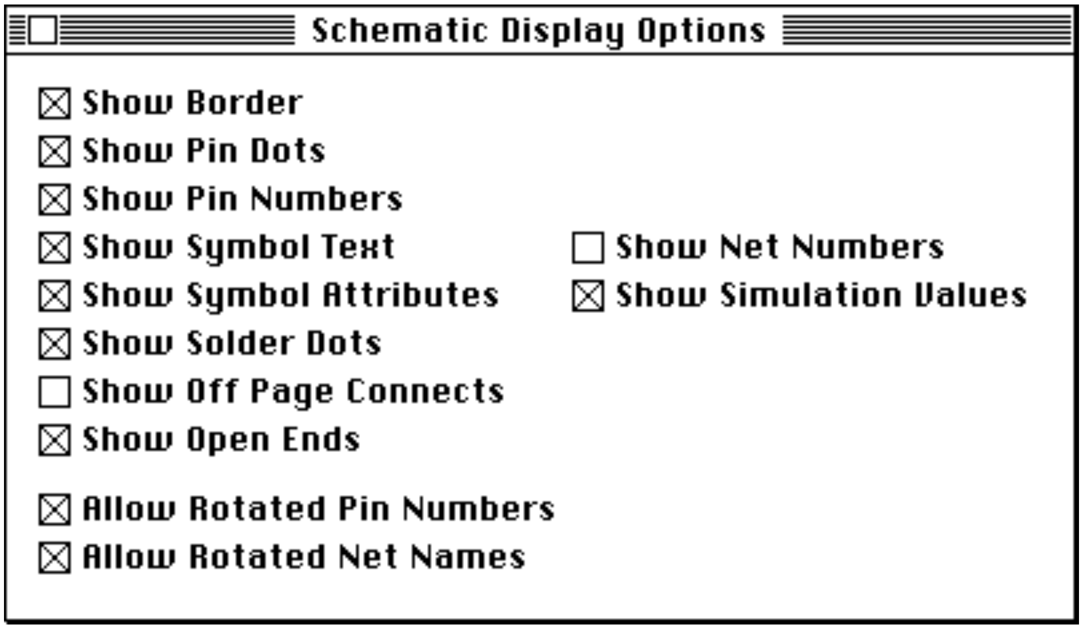
Default Pin Name Offset
This parameter sets the default pin name offset. This is the distance measured in quarter primary grids between the pin name and the actual pin. This parameter is used by the **Add->Show Pin Names** command and **File->Create Symbol** command. Values can range between 0 and 15.

Simulator
This control configures ECS for a particular simulation environment. Current environments include *Verilog*, *VHDL*, and *SILOS II*. The major effect of the simulation environment is to provide hardware description language (HDL) templates in the Symbol Editor and Hierarchy Navigator. This facilitates writing behavioral models for the simulators mentioned. It is also used in the *Dynamic Waveform Interface*.

Text Editor
This parameter allows you to choose the editor used whenever ECS needs to open a text file. You push the **Text Editor:** button to select the desired editor from a file selection dialog box.

Controls->Display Options

This section contains a variety of parameters used to control the display of schematics. Figure 2-8 shows the Display Options dialog box.


Schematic Display Options

☒ Show Border
☒ Show Pin Dots
☒ Show Pin Numbers
☒ Show Symbol Text
☒ Show Symbol Attributes
☒ Show Solder Dots
☐ Show Off Page Connects
☒ Show Open Ends
☐ Show Net Numbers
☒ Show Simulation Values

☒ Allow Rotated Pin Numbers
☒ Allow Rotated Net Names

Figure 2-8 Display Options Dialog Box

Show Border
This parameter turns on and off the screen and plotter display of the schematic and symbol borders. It can be modified during the work session using the **Misc->Control** command.

Show Pin Dots
This parameter turns on and off the screen and plotter display of pin dots. It can be modified during the work session using the **Misc->Control** command.

Show Pin Numbers
This parameter turns on and off the screen and plotter display of pin numbers. This is useful for reducing the repaint time and clutter on a crowded drawing. It can be modified during the work session using the **Misc->Control** command.

Show Symbol Text
This parameter turns on and off the screen and plotter display of text inside symbols. This is useful for reducing the repaint time and clutter on a crowded drawing. It can be modified during the work session using the **Misc->Control** command.

Show Symbol Attributes
This parameter turns on and off the screen and plotter display of symbol attributes. This is useful for reducing the repaint time and clutter on a crowded drawing. It can be modified during the work session using the **Misc->Control** command.

Show Solder Dots
This parameter turns on and off the screen and plotter display of solder dots. It can be modified during the work session using the **Misc->Control** command.

Show Off Page Connects
On multiple sheet schematics, you can show references to other sheets with nets that connect across more than one sheet. This control enables the cross-reference display on wire segments with their names at the end of the wire.

This parameter can be modified during the work session using the **Misc->Control** command.

Show Open Ends
This parameter turns on and off the screen and plotter display of dangling wires and symbol pins.

Wires not terminating on a net name flag, symbol pin, or another wire are considered errors in a schematic drawing. The highlighting of these errors on the screen and plotter is turned on and off with this parameter.

This parameter can be modified during the work session using the **Misc->Control** command.

Allow Rotated Pin Numbers
The numbers on the pins of a symbol are normally displayed as horizontal text justified away from the symbol. As an option, the numbers of pins on the top and bottom of a symbol can be displayed with vertical text instead of horizontal text.

Allow Rotated Net Names
This parameter allows net names at the end of vertical wires to be displayed either vertically or horizontally.

Show Net Numbers
This parameter turns on and off the screen and plotter display of node numbers. This is useful for reducing the clutter on a crowded drawing. This parameter only affects the display in the Hierarchy Navigator.

Every node in the circuit has a node number assigned in the ECS database. These node numbers are used internally and can also be used by simulators like SPICE, which require numbers rather names.

This parameter can be modified during the work session using the **Misc->Control** command.

Show Simulation Values
This parameter turns on and off the screen and plotter display of simulation values on the schematic. This feature is only needed when viewing the results of a simulation. It can be modified during the work session using the **Misc->Control** command.

This parameter only affects the display in the Hierarchy Navigator.

Controls->Global Nets

This section of the ECS Preferences Editor allows you to define global signals used throughout your design. The Global Signal Editor is shown in Figure 2-9.

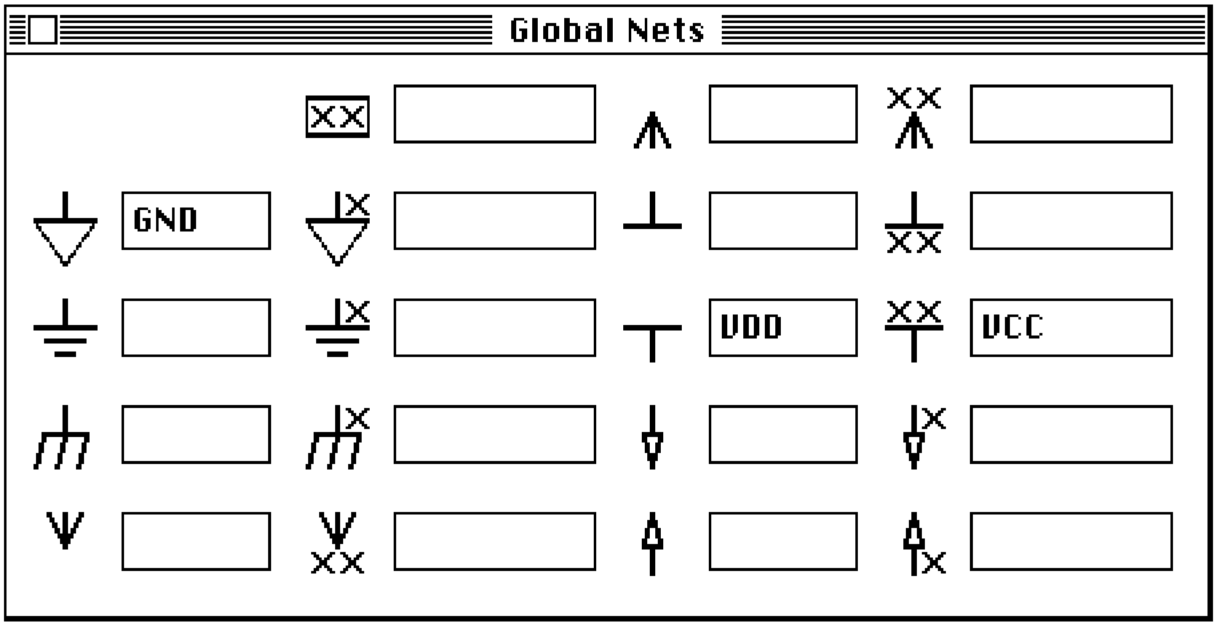


Figure 2-9 Global Signal Editor

Global signals can be accessed across all levels of hierarchy and across all sheets and schematics in a design. For this reason, any names assigned as global signals cannot be used as local signal names in your design.

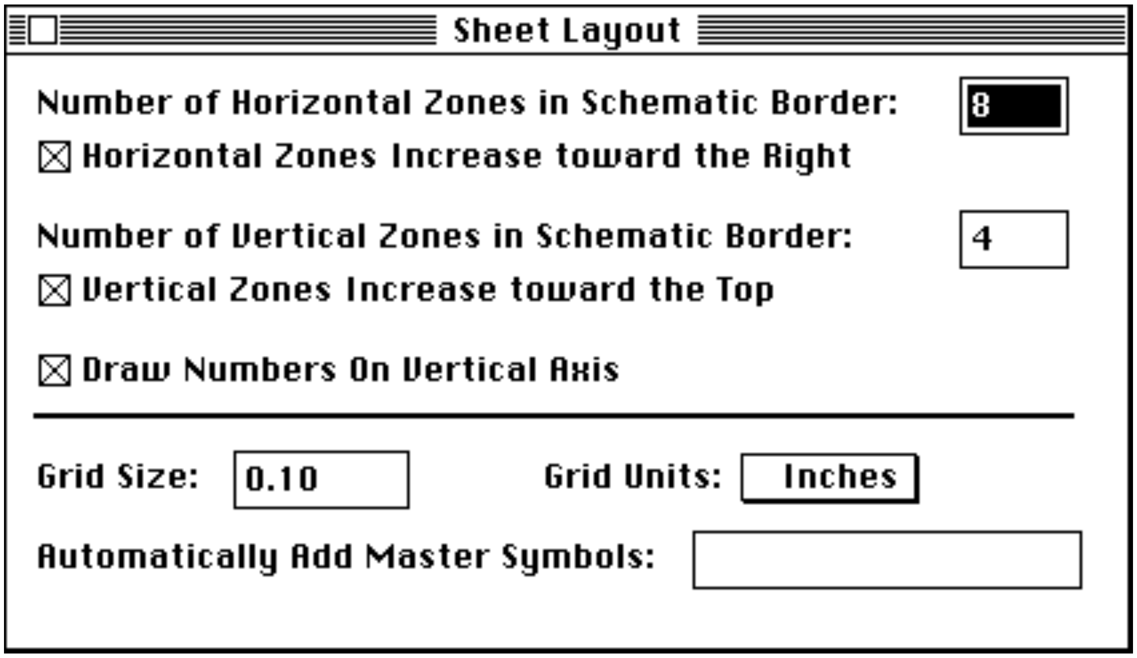
You can choose from three different types of global signals. They are:

Unnamed Symbol	You can attach a single name to each of the symbols in the first and third columns. When you specify one of those names in your schematic, the corresponding symbol is attached to the net you are naming. No name is shown. The symbol is the only external indication of the name of the net. You can **Query** the net and find the net name if needed.
Named symbol	You can attach names to the symbols in the second and fourth columns. When you name a net with one of the names associated with these symbols, the corresponding symbol is drawn attached to the net. The *X*s in the symbol are replaced with the actual net name when placed in your schematic.
No Symbol	You can attach names to the box containing two *X*s at the top of the second column. These names are global; whenever you attach one of these names to a net, it is global throughout all levels of hierarchy. The name appears attached to the net.

Type the name of the global net next to the selected symbol. *Unnamed* symbols in the first and third columns can only have a single name associated with each symbol. *Named Symbols* and the *No Symbol* icon can have multiple names associated with them, separated using spaces.

Controls->Sheet Layout

The reference location zones, grid spacing and text size are specified in this section. Figure 2-10 shows the Sheet Layout dialog box.


Sheet Layout

Number of Horizontal Zones in Schematic Border: 8

☒ Horizontal Zones Increase toward the Right

Number of Vertical Zones in Schematic Border: 4

☒ Vertical Zones Increase toward the Top

☒ Draw Numbers On Vertical Axis

Grid Size: 0.10

Grid Units: Inches

Automatically Add Master Symbols:

Figure 2-10 Sheet Layout Dialog Box

You can modify the following parameters.

Number of Horizontal Zones in Schematic Border
This parameter specifies the number of zones used on the horizontal axis. Allowed values are from 1 to 8. Every component placed on a schematic can be located using the vertical and horizontal reference zones. The **Misc->Query** command references components using this scheme.

Horizontal Zones Increase Toward the Right
Checking this box causes the schematic border zones on the top and bottom of the schematic to be numbered from left to right.

Number of Vertical Zones in Schematic Border
This parameter specifies the number of zones used on the vertical axis. Allowed values are from 1 to 8. Every component placed on a schematic can be located using the vertical and horizontal reference zones. The **Misc->Query** command references components using this scheme.

Vertical Zones Increase Toward the Top
Checking this box causes the schematic border zones on the left and right edges of the schematic to be numbered from top to bottom.

Draw Numbers on Vertical Axis
This parameter controls whether the vertical zone references are numbers and the horizontal ones are letters, or vice versa.

Grid Size
The schematics and symbols are defined on a primary grid. The spacing of adjacent grid lines for display and plotting purposes is given by the product of the *Units* parameter and the *Grid Size* parameter.

Grid Units
This parameter defines the units used for the primary grid on which schematics and symbols are drawn. The choices are:

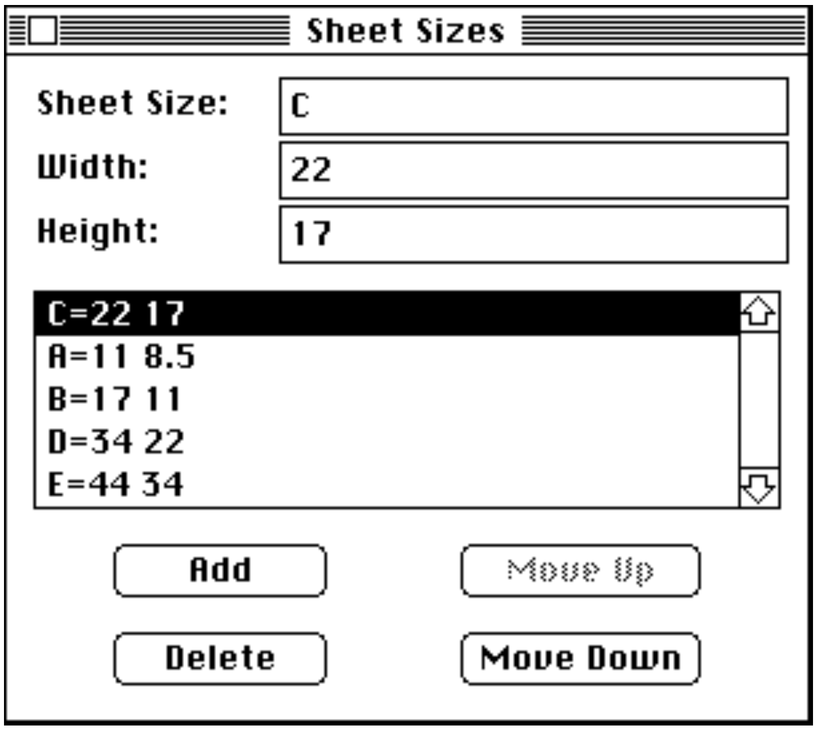
- inches
- centimeters
- millimeters

Automatically Add Master Symbols
Any master symbols listed in this edit field are automatically placed on all sheets as they are created. If the origin of a master symbol is at the bottom right of the symbol, the symbol is automatically placed at the bottom-right corner of the schematic. Similarly, if the origin of the master symbol is in one of the other three corners, the symbol is automatically placed in the corresponding corner of the schematic.

Multiple master symbols can be specified by separating them with spaces in the entry field.

Controls->Sheet Sizes

This menu sets the drawing sheet sizes that are allowed in ECS. A list box is presented with the current sizes. The controls operate in two different modes. Figure 2-11 shows the Sheet Sizes dialog box.



Sheet Size: C
Width: 22
Height: 17

C=22 17
A=11 8.5
B=17 11
D=34 22
E=44 34

Figure 2-11 Sheet Sizes Dialog Box

- If there are no sheets selected in the list box, an *Add* button allows you to create new sheet size entries.
- If there are sheets selected in the list box, the following buttons are presented.

Add	Allows you to create a new sheet size entry.
Delete	Deletes the currently selected sheet size entry from the list box.
Move up	Moves the currently selected line one position closer to the top of the list.
Move down	Moves the currently selected line one position closer to the bottom of the list.

The sheet size names are arbitrary. Because schematics are usually wider than they are tall, the width number is usually larger than the length measurement. The sizes are measured in the units specified in the **Controls->Sheet Layout** section of the ECS Preferences Editor.

The maximum dimensions supported are 8000 grids in each axis.

When the Schematic Editor is started on a new drawing, the default sheet size is the size specified at the top of the list box. The sheet size for a particular drawing can be changed using the **File->Sheet** command in the Schematic Editor.

Typical sheet size settings are listed below.

English		Metric	
C=22	17	A3=420	297
A=11	8.5	A4=297	210
B=17	11	A2=594	420
D=34	22	A1=841	594
E=44	34	A0=1189	841

Controls->Waves Controls

This section sets parameters used in the Waveform Tool. These parameters and how to modify them are described in the *Waveform Tool* section.

File->New

This command creates a new file using the system defaults. This file contains system configuration information generated by the ECS Preferences Editor.

File->Open

This command presents a standard file dialog box allowing you to open any text file containing configuration information. Usually this file is called **ecs.ini**.

File->Quit

This command quits the ECS Preferences Editor. A dialog box displays the following options.

Save	Click the mouse on this button to save your edits.
Discard	Click the mouse on this button if you want to quit the editor without saving your edits.
Cancel	Click the mouse on this button if you wish to return to your editing session.

File->Revert to Saved

This command restores the file to its original state before any edits were made. This allows you to recover from errors made while editing the configuration information.

File->Save

This saves any changes made to the configuration information using the ECS Preferences Editor. Applications requiring this configuration information must be restarted.

File->Save As

Any changes made to the configuration information using the ECS Preferences Editor can be saved to a different file name using the **Save As** command. Applications requiring this configuration information must be restarted.

Search Paths

This section of the ECS Preferences Editor sets the search paths for symbol libraries, model libraries, project libraries, and simulation models. Refer to the discussion of Libraries and Directory Structures at the beginning of this section for an explanation of the use of libraries and how to set the search paths.

1. Select the ECS Preferences Editor. The main ECS Preferences Editor window is displayed.
2. Select one of the entries under the **Search Paths** menu:

   Project Directory

   Model Library

   Symbol Library

   A list is displayed of the current libraries of the specified type.
3. Select a library from the displayed list. The operations you can perform are:

Add Path	Pops up a standard file dialog box, allowing you to add a new library to the search path.
Delete	Deletes the currently selected library from the search path.
Move Down	Interchanges the currently selected library with the library immediately below it in the list box. This is used to change the order in which libraries are searched. Libraries closer to the top of the list box are searched before libraries below them. Libraries are searched in order from the top of the list box to the bottom.
Move Up	Interchanges the currently selected library with the library immediately above it in the list box. This is used to change the order in which libraries are searched. Libraries closer to the top of the list box are searched before libraries below them. Libraries are searched in order from the top of the list box to the bottom.

In addition to these commands to manipulate library paths, you can *enable* paths by selecting a given path and then checking the *enabled* box. Library paths are disabled by leaving the *enabled* check box unchecked.

Tools->Navigator Processes

This section of the ECS Preferences Editor allows you to add menu entries to the **Processes** menu of the Hierarchy Navigator. The menu entries created in the Navigator spawn tasks for netlisting or simulation. ECS supports several netlisters that are accessed through this menu. In addition, you can write your own interface programs that can be called from the **Processes** menu.

A list box is presented with the current processes selection. The controls for modifying processes operate in two different modes.

- If there are no processes selected in the list box, an *Add* button allows you to create new process entries.
- If there is a process selected in the list box, the following buttons are presented:

Add	Pops up a file selection dialog box. You can select ECS applications to add to the **Processes** menu.
Move up	Moves the currently selected line one position closer to the top of the list.
Move down	Moves the currently selected line one position closer to the bottom of the list.
Delete	Deletes the currently selected process entry from the list box.

The entries themselves consist of three edit fields:

Menu Label	This is what appears in the **Processes** menu of the Hierarchy Navigator. It appears to the left of the equal sign in the list box.
Application	This is the application that is executed when the corresponding menu entry in the **Processes** menu of the Hierarchy Navigator is selected. It appears to the right of the equal sign.
Flags	Flags are options that apply to the ECS application. Refer to the *Interfaces* section for details on flags used with a particular application.

The following entries are standard. Other entries are described in the *Interfaces* section.

Net List By Pin =pcbnet -pin	Generic list in part/pin order

Tools->Navigator Tools

This section of the ECS Preferences Editor allows you to add menu entries to the **Tools** menu of the Hierarchy Navigator. The menu entries created in the Navigator spawn tasks for the Waveform Tool (WT) and the Design Analysis Tools (DAT). In addition, you can write your own interface programs that can be called from the **Tools** menu.

Adding a new tool to the **Tools** menu of the Hierarchy Navigator is identical to adding a new process to the **Processes** menu. Refer to the description of the **Controls->Processes** menu to see how this is done.

Tools->Schematic Tools

Any utilities, netlisters or other processes created using the Schematic PIK (SCPIK) can be added to the Tools menu of the symbol editor. The verilog and vhdl netlisters are the only tools shipped with the ECS that currently work in the **Tools->Schematic Tools** menu.

Adding a new tool to the **Tools** menu of the Schematic Editor is identical to adding a new process to the **Processes** menu of the Hierarchy Navigator. Refer to the description of the **Tools->Processes** menu to see how this is done.

Tools->Symbol Tools

Any utilities, netlisters or other processes created using the Schematic PIK (SCPIK) can be added to the Tools menu of the Symbol editor. The verilog and vhdl netlisters are the only tools shipped with the ECS that currently work in the **Tools->Symbol Tools** menu.

Adding a new tool to the **Tools** menu of the Symbol Editor is identical to adding a new process to the **Processes** menu in the Hierarchy Navigator. Refer to the description of the **Tools->Processes** menu to see how this is done.

3

Getting Started

This section gives an overview of the way the Engineering Capture System (ECS) treats schematics and hierarchy. A brief tutorial guides you through the fundamental operation of ECS while building a small hierarchical design.

The major topics covered in this section are:

- Schematics in the ECS environment
- How symbols are used in schematics
- Hierarchy in a design
- The Hierarchy Navigator in the ECS environment
- Symbol Editor tutorial
- Schematic Editor tutorial
- Hierarchy Navigator tutorial

Concept of Design Entry With ECS

This section explains the design flow using ECS. This includes defining a number of terms and concepts as well as showing how the design is organized.

The speed of ECS allows a great deal of flexibility. A true on-line connectivity database eliminates the extraction or compilation phase usually required before using the electrical connectivity information. Instead of waiting hours for a 20k transistor netlist to compile and be ready, you wait only seconds. This allows many more *What if?* questions to be asked and then actually tested in the design.

ECS supports a hierarchical design style. This allows pieces of a design to be worked on independently and then brought together near the completion of the design. It also allows you to complete much larger designs on your system with a given amount of memory.

The basic design flow can be either top-down or bottom-up. Top-down design proceeds as follows.

1. Create symbols for the highest-level blocks used in the design.
2. Connect these high-level blocks in a schematic.
3. Design the *insides* of the high-level blocks, and create symbols and schematics that underlie the high-level blocks.
4. Check for design errors using the ECS checking tools and iterate steps one to four until the design is complete.
5. Use additional tools such as simulators and the Waveform Tool to further verify the integrity of your design.
6. Generate an output file containing the essence of the design, usually a netlist. Transfer this design file to the next manufacturing step.

Bottom-up design is similar except the first three steps are replaced as follows.

1. Create all the primitive symbols that are needed for the design.
2. Connect the primitives into successively higher-level blocks using schematics.
3. Combine the higher-level blocks into the highest-level schematic.

ECS supports both design flows as well as combinations of the two methods.

The material in this section is presented in the following order.

- Schematics
- Hierarchy
- Design Verification

Schematics

A schematic is a drawing, consisting of one or more sheets, that represents a portion of an electronic circuit. In a simple design, one schematic is sufficient to represent the entire circuit in terms of its primitive elements. As the design becomes more complex, a hierarchy of schematics is used, first showing the design in terms of block elements and later expanding each block element into a schematic that contains the primitive elements. As the complexity increases, additional levels of hierarchy are used to describe the design.

Schematic definitions are stored in files that use the name of the schematic with a *.sch* extension. Design files can be maintained in both the main design directory and project directories in ECS.

Each schematic in the hierarchy is composed of four types of entities: symbols (including iterated instances or arrays), wires, attributes, and graphics.

Symbols

A symbol represents a physical component or a portion of a physical component in an electrical circuit. In hierarchical schematics, the symbol can also represent a sub-circuit consisting of several interconnected elements. Each placement of a symbol is called an instance of the component or sub-circuit.

The symbol definition is maintained as an independent file of data. Symbol file names have the *.sym* extension. This is added automatically by the system when you save a symbol drawing. A symbol can be stored in a library for use in many designs or can be kept in the design directory for use in a specific design. Updating a symbol causes the new version of the symbol to replace the old version wherever it was used.

A schematic symbol is composed of three types of entities: graphic entities, pins, and attributes.

Graphic Entities

Graphic entities provide the lines, arcs, circles, and text that comprise the picture of the symbol recognized by you. Graphics entities do not have any electrical meaning within the system. They provide a graphic indication of the placement of the symbol in a schematic.

Pins

The pins on the symbol represent the points where connections between the symbol and other elements in the circuit are made.

If the symbol represents a physical component, the symbol pin represents the physical pin where a conductor can be attached. If the symbol represents a sub-circuit, the symbol pin represents the connection to an internal net of the sub-circuit.

Attributes

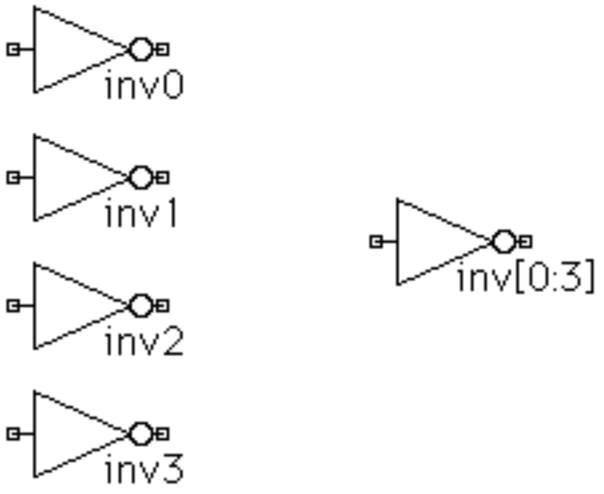
There are several data items that describe the features of a symbol and its pins. These data items are attached to the symbol by means of attributes. An attribute has a name and a value.

The attributes that apply to all instances of the symbol are assigned values when the symbol is first created. For example, the vendor part number and the simulation model are assigned during symbol creation. Attributes that apply to a single instance of the symbol are assigned after the symbol is placed in the design. For example, assignment of the instance name and the load on an input pin is done after the symbol is placed.

Iterated Instances (Arrays)

Iterated instances allow a single instance of a symbol to represent many identical instances in a parallel connection. Figure 3-1 shows two ways of representing four parallel buffers. In case A, four separate inverters are added to the schematic. In case B, one symbol with the instance name of *inv[0-3]* represents the bank of four inverters.

If a Block symbol is used in an iterated instance, you can push into each instance represented by the iterated instance.



(A) non-iterated (B) iterated

Figure 3-1 Iterated and Non-Iterated Inverter Arrays

Both examples represent exactly the same circuit connectivity. The iterated instance is a more compact way of representing parallel symbols. This is extremely useful for bit-slice designs where complicated structures are duplicated many times.

Wires

The electrical connections in a schematic are represented by lines called wires. When a wire touches a symbol pin, a connection is made between that pin and all other pins touched by the wire. All of the pins and wires that are connected form a net.

Each net has a name that is either assigned by the user or by the system. When the system assigns net names, it uses a unique name for each net. You have the option of merging two nets by assigning the same name to each net.

A wire that represents a single conductor is called a signal. A group of signals can also be represented by a single wire. Wires representing more than one signal are called buses.

Busing Features

Electronic schematics frequently contain groups of related signals. Representing this group of signals as a single wire produces a more functionally relevant drawing that is less cluttered in appearance. ECS supports simple buses, ordered buses, and bus pins.

Simple Buses

Simple buses are used to route groups of signals between symbols within the same schematic. Signals enter a bus at points called bus taps. A simple bus contains all the signals that enter the bus using bus taps. The order of the signals within the bus is not defined.

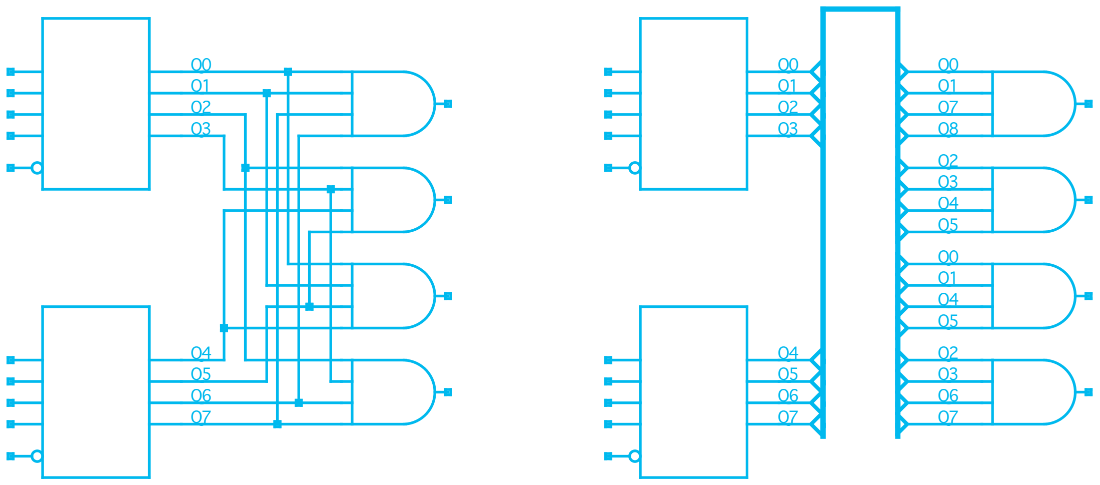


Figure 3-2 Same Circuit With and Without a Simple Bus

Simple buses can be named or unnamed. There is no relationship between the bus name and the signals contained in the bus.

A simple bus cannot make a connection to a symbol pin.

Ordered Buses

An ordered bus enforces a relationship between the name of the bus and the signals it contains. For example, an ordered bus containing the signals CS, PS, and WR has the name CS,PS,WR.

The ordered bus name is composed of the individual signal names separated with commas. No spaces are allowed in the names.

Another example of an ordered bus is A[0-7]. In this case an index is used as shorthand notation to represent eight signals with a single name. Ordered buses can combine single signals and other ordered buses. For example, the ordered bus name

CS,PS,WR,A[0-7] combines the single signals CS, PS, and WR with the ordered bus A[0-7].

Bus Pins
A bus pin is a symbol pin that represents a connection to a group of signals in the lower-level schematic. You create a bus pin by assigning a pin name that is a list of signals. Each of the signals listed in the bus pin's name must appear in the lower-level schematic. This is how connectivity is established between the symbol and the underlying schematic.

When an ordered bus is connected to a bus pin, the number of signals in the bus must match the number of signals represented by the pin. The first signal in the bus (by definition also the first signal in the bus's name) is connected to the first signal represented by the pin. The remaining signals in the bus are connected to the corresponding signals in the lower-level schematic.

Attributes

Each net, symbol, and pin in a schematic has attributes. Some attributes, like simulation delays, are assigned to the symbols and pins when the symbols are created. Other attributes are assigned to nets, symbols, and pins after they are added to the schematic.

Graphics

The schematic can also contain graphic lines, arcs, circles, and text that are not part of the circuit description. These graphics are included in the drawing to convey information to the reader. Title blocks and notes are typical uses for these entities.

A sample schematic is shown in Figure 3-3.

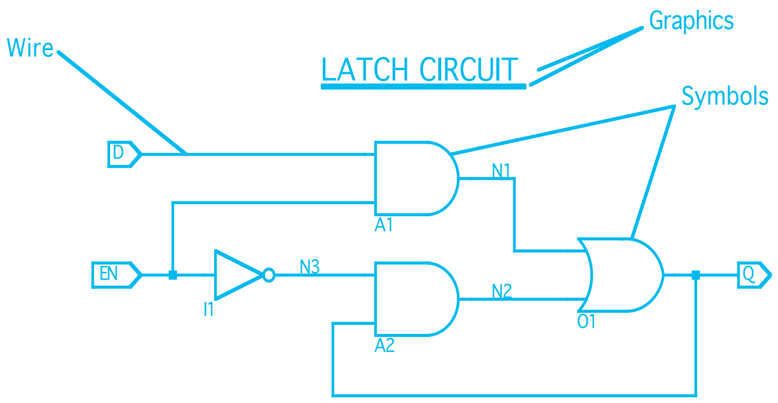


Figure 3-3 Sample Schematic

Hierarchy

ECS uses a hierarchical approach for working with complex electronic designs.

The hierarchical concept is used in many aspects of daily living. One common application of this concept is the hierarchy of maps. When you look at a map of the United States you see a representation of a vast expanse of land. On this map, your home state occupies a small area with only a few of the larger features shown. The largest cities are shown as dots and the smaller towns are not shown at all. Only the major interstate highways appear.

When you examine a map of your state, you see a smaller area in more detail. The major cities and all of the towns are shown on the map. All highways in the state and some of the major thoroughfares in the big cities also appear.

A map of your city or town represents a small portion of the overall country, but it is shown at a greater level of detail. Each section of the town and all of the streets are shown. You could easily locate your home or business.

The application of hierarchy to schematic design is similar to its application in mapping. The basic concept of working from an overview of the entire design to a detailed view of the individual part applies in both applications.

At the top level in the hierarchy, an entire design is represented as a complete, but not highly detailed, schematic. As you descend in the hierarchy, more detailed views are shown of smaller portions of the circuit. At the lowest level of the hierarchy, a small portion of the circuit is represented in terms of its primitive elements. The complexity of each level in the hierarchy allows you to understand the whole level.

Symbols, Schematics, and Hierarchy in ECS

Hierarchy is established in ECS by having a symbol replace a lower-level circuit schematic. There are three requirements placed on the symbols used to represent hierarchy.

- The symbol must have the same name as the schematic containing the underlying circuitry. Having the same name is how ECS establishes the relation between a schematic and the symbol that represents it.
- The net names in the underlying schematic must match the pin names in the symbol.
- The symbol must be of type *block* or, if the symbol is used in a model directory, it can also be of type *cell*.

In Figure 3-4, the symbol defined in the file **latch.sym** represents the schematic contained in the file **latch.sch.**

Names are also used within the Block symbol and the schematic to link the symbol pins to the nets within the schematic. A wire connecting to a pin named CLK on the symbol is also connected to the net named CLK in the underlying schematic. The **Misc->Check** command of the Schematic Editor or the **Tools->ERC Check** command in the Navigator flag an error if a Block symbol has a pin that does not have a corresponding net in the related schematic.

In the latch in Figure 3-4, the symbol pin D corresponds to the net in the schematic, which is named D. The other pins, Q, and EN, also represent named nets in the schematic.

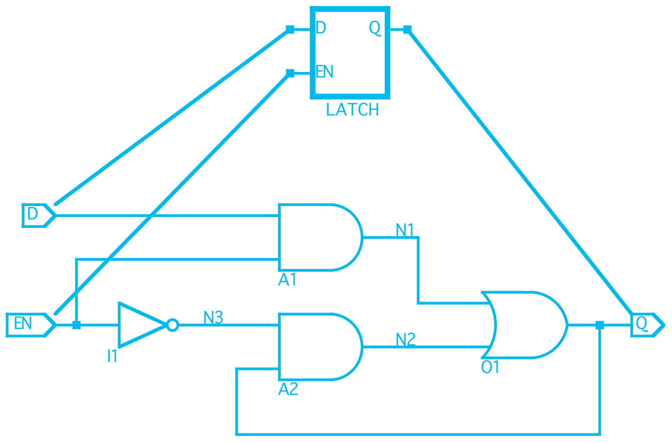


Figure 3-4 A Symbol and Its Underlying Schematic

Hierarchical Design Structure

When a symbol is placed in a schematic, the component or sub-circuit that the symbol represents is added to the circuit. If you placed a *latch* symbol in a schematic named *reg4*, you would actually be including the OR gate, inverter, and two AND gates that are in the schematic *latch*. If you then added three more copies of symbol *latch* to schematic *reg4,* you would be building a 4-bit register circuit consisting of 8 AND gates, 4 OR gates, and 4 inverters.

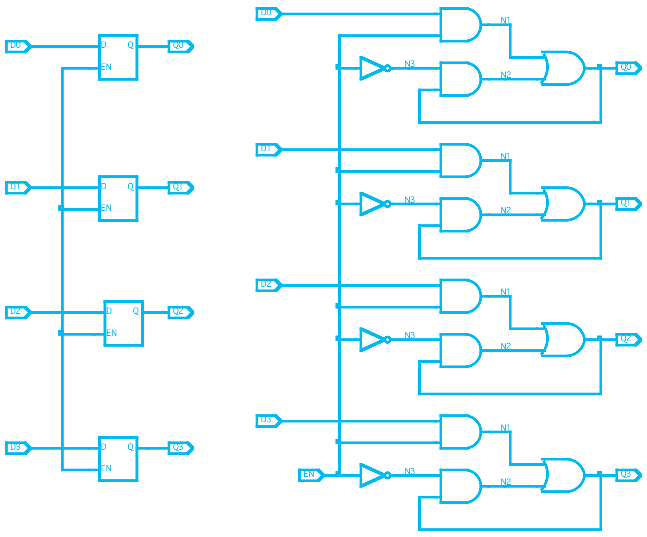


Figure 3-5 Circuit reg4 and Its Equivalent Circuit

This process can be repeated by defining a symbol for schematic *reg4* and placing that symbol in a higher-level schematic. If you created a schematic for a 16-bit register, *reg16*, by placing four copies of symbol *reg4* you would be defining a circuit with a total of 64 gates. Rather than placing 64 gates, the circuit would be defined by placing a total of 12 symbols among the three levels of hierarchy.

Hierarchical Naming

In the latch circuit example, the inverter has the instance name *I1*. In schematic *reg4*, four copies of the symbol *latch* are placed and assigned instance names *L1* through *L4*. Schematic *reg4* therefore contains four copies of the inverter named *I1*. You can distinguish between these inverters by combining the inverter's instance name with the instance name of the circuit containing it. The four inverters are named:

- L1.I1
- L2.I1
- L3.I1
- L4.I1

Reg16 contains four copies of *reg4* named R1 through R4. Each copy of *reg4* contains the four inverters as named above. The 16 inverters are named by combining the instance names of the four *reg4* symbols with each of the four instance names of the inverters in each *reg4*. The inverters are named:

- R1.L1.I1
- … R1.L4.I1
- ... R2.L1.I1
- ... R4.L4.I1

When you view the individual schematic *latch,* you see the instance names of the gates without any hierarchical context. When the schematic becomes part of a larger design, the instance names must include the hierarchical path in order to be unique.

Nets in the Hierarchy

The schematic definition for the circuit *latch* contains local and external nets. A local net connects the output pin of the inverter to the AND gate. Two other local nets connect the outputs of the AND gates to the inputs of the OR gate. These nets have been named N1, N2, and N3. When the 16 copies of this circuit are combined in *reg16*, 16 copies of these nets are created. Because the 16 nets named N1 are not connected, unique names are created by the system.

The method for specifying a unique name for each net in the hierarchy uses the instance name of the schematic where the net is defined as a prefix to the local net name. A hyphen delimits the name of the net from the instance name. The 16 N1s then become:

R1.L1-N1, R1.L2-N1 ... R4.L4-N1

This naming is done automatically by the system.

The schematic latch also contains three external nets: D, EN, and Q. The symbol pins on symbol *latch* connect these nets to nets on the hierarchical level above. The names for external nets are transmitted downward through the hierarchy. For example, the signal ENABLE in the schematic *reg16* replaces the signal EN in the original latch.

Design Debugging and Verification Tools

ECS is set up to prevent as many errors as possible during design entry. A large percentage of design time is spent tracking down floating pins, overloaded nets, and miswired connections. ECS contains two levels of checking that attempt to prevent or report errors as early in the design process as possible. The first level of error can be prevented as you design your schematic. The second type of error is detected after the whole design is completed.

The system prevents adding a signal to an ordered bus that is not contained in the name of the bus. It also prevents shorting differently named nets. These are examples of errors that are detected immediately upon design entry.

There are other errors that only become errors when the design is said to be complete. A wire or pin that is left unconnected or an unnamed signal that is tapped from a bus are normal conditions while the design is being built. In some cases it is practical to show some indication of these errors, such as the dots that are placed on open pins and hanging line ends. Otherwise, errors of this type are reported when the schematics and symbols are combined into the design hierarchy.

A very powerful feature in ECS allows you to run the Check utility and then query any of the errors. When you click on an error in the error report, the system automatically jumps to that portion of the schematic where the error is located. You can quickly step through any errors in your design, fixing them as you go.

Automatic Aliasing

Nets in ECS take their name from the highest level in the design. This allows the use of local names — such as *D* and *Q* — in a flip-flop cell. These local names are automatically replaced by the appropriate top level names, such as *clock1* or *data7*. This greatly speeds tracing of signals through a design containing many levels of hierarchy. It is not necessary to perform a mapping of names as is commonly done on other systems.

After Design Entry Is Complete

There are many things a good design entry package can do to make life easier for a designer once a design is free of entry errors. ECS provides a very powerful waveform viewing package, the Waveform Tool (WT). It supports true cross-probing between the schematic and waveforms. Point to a waveform and it highlights that net in the schematic. Point at a waveform on a schematic and it adds that waveform to the display.

You can point to a waveform and query the system to find from where in the circuit that waveform is driven. The system automatically zeros in on the proper location on the correct sheet of the correct level in the hierarchy to show the device that is driving a particular waveform.

A very strong interface exists between the waveform display and the schematic. The logic values of nodes in the schematic can be displayed right on the schematic. This can be set up to change displayed values as the simulation is actually progressing. Many interfaces are currently supported including SPICE, Verilog, VHDL, EDIF, Timemill, PC/SILOS, SILOS II, PADS, Racal Redac, Cadnetix, Futurenet. Using the Processor Interface Kit, you can make your own simulation environment truly interactive.

A Brief Tutorial

This section guides you through the design of a small circuit using ECS. You use many of the features of the Schematic and Symbol Editors to develop the symbols and schematics that define a hierarchical design. Then you use the Hierarchy Navigator to review the design and to extract a netlist. Finally, you print a hardcopy of the design using the ECS Executive Program.

The design to be completed is the 4-bit register, *reg4*, which is described in the previous section. This register is built by combining four latch circuits. Each latch is built with two AND gates, an OR gate, and an inverter.

Conventions Used in This Tutorial

One important convention used throughout this manual is the way of showing the parent menu of a command. For example, the **ZoomIn** command is located in the **View** menu. The command is referenced in this manual as **View->ZoomIn**. Referencing the command in this way tells you where the command is located.

Starting the System

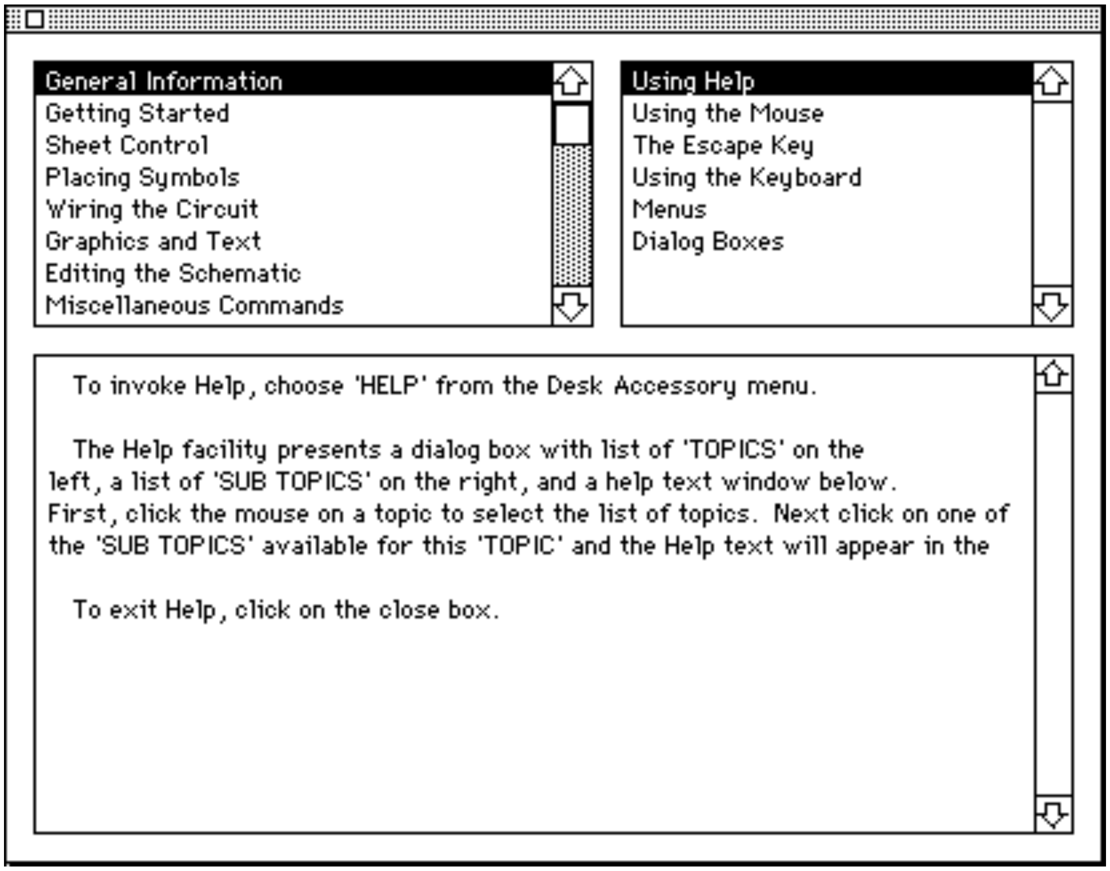
If you have not already installed the ECS software (including setting the paths to your symbol libraries) on your machine, refer to the *Before You Begin* section.

The design you are about to create needs a folder to contain its files. Create a new folder with a meaningful name, such as *reg4*, wherever you are storing your design. It is necessary to run the ECS Preferences Editor at least once before starting your design. If you didn't run the ECS Preferences Editor as part of the installation procedure, use the following procedure to do so now.

1. Double-click on the ECS Preferences Editor located in the main ECS folder.
2. Use the **File->Open** command to open the file named **ecs.ini**.
3. Use the **File->Save As** command to save the file back overtop itself. The act of performing a save makes the configuration information accessible to the ECS applications. Once this file has been saved once, it is only necessary to run the ECS Preferences Editor and save the configuration file if changes are made.

On-Line Help

Each application in ECS has on-line help available. The Help utility is accessed under the **Apple** menu (). The help window looks like Figure 3-6.



To invoke Help, choose 'HELP' from the Desk Accessory menu.

The Help facility presents a dialog box with list of 'TOPICS' on the left, a list of 'SUB TOPICS' on the right, and a help text window below. First, click the mouse on a topic to select the list of topics. Next click on one of the 'SUB TOPICS' available for this 'TOPIC' and the Help text will appear in the

To exit Help, click on the close box.

Figure 3-6 Typical Help Window

The help dialog box has a list of topics on the top left and a list of sub-topics on the right. To access help for a particular command or technique:

1. Use the scroll bars to find the major topic in the left list box.
2. Click the mouse on the desired major topic. A list of sub-topics is presented in the right-hand list box.
3. Click the mouse on the desired sub-topic in the right-hand list box. Information regarding this topic is displayed in the large bottom window.
4. Scroll the bottom window, if necessary, to view all of the help information about the selected sub-topic.

Designing a Circuit

There are three major steps in entering a circuit design using ECS.

1. Creating the necessary symbols
2. Creating the schematics
3. Creating the hierarchy and analyzing the circuit

The first step in designing the circuit is to create symbols. There are already AND, OR, and inverter symbols defined in the library shipped with ECS, but you create your own version of an AND gate in this design.

Creating Symbols

The Symbol Editor is used for creating symbols. The following explanation is aimed at integrated circuit design. For a description of how to use ECS for PCB design, refer to the section, *Differences Between IC and PCB Design*.

Start the Symbol Editor by double-clicking on the application called *Symbol* in the main ECS folder.

The Symbol Editor's main window appears. There is also a dialog box that contains the various types of symbols that can be created. The choices are listed below.

Gate	Primitives used for printed circuit applications.
Component	Primitives used for printed circuit applications.
Cell	Primitives used for integrated circuit design.
Block	Used in all applications for representing hierarchy.
Pin	Used for indicating special pins, such as edge connectors and test points.
Graphic	Used for notes. These symbols have no electrical information associated with them.
Master	Used for title blocks and other forms that always appear at some fixed place on the drawing.

The first symbol you create is an AND symbol, which is a primitive symbol. Click the mouse on the word *Cell*; and then click the OK button. The word CELL now appears at the end of the title bar indicating the type of symbol currently being edited. Figure 3-7 shows the Symbol Editor window at this point.

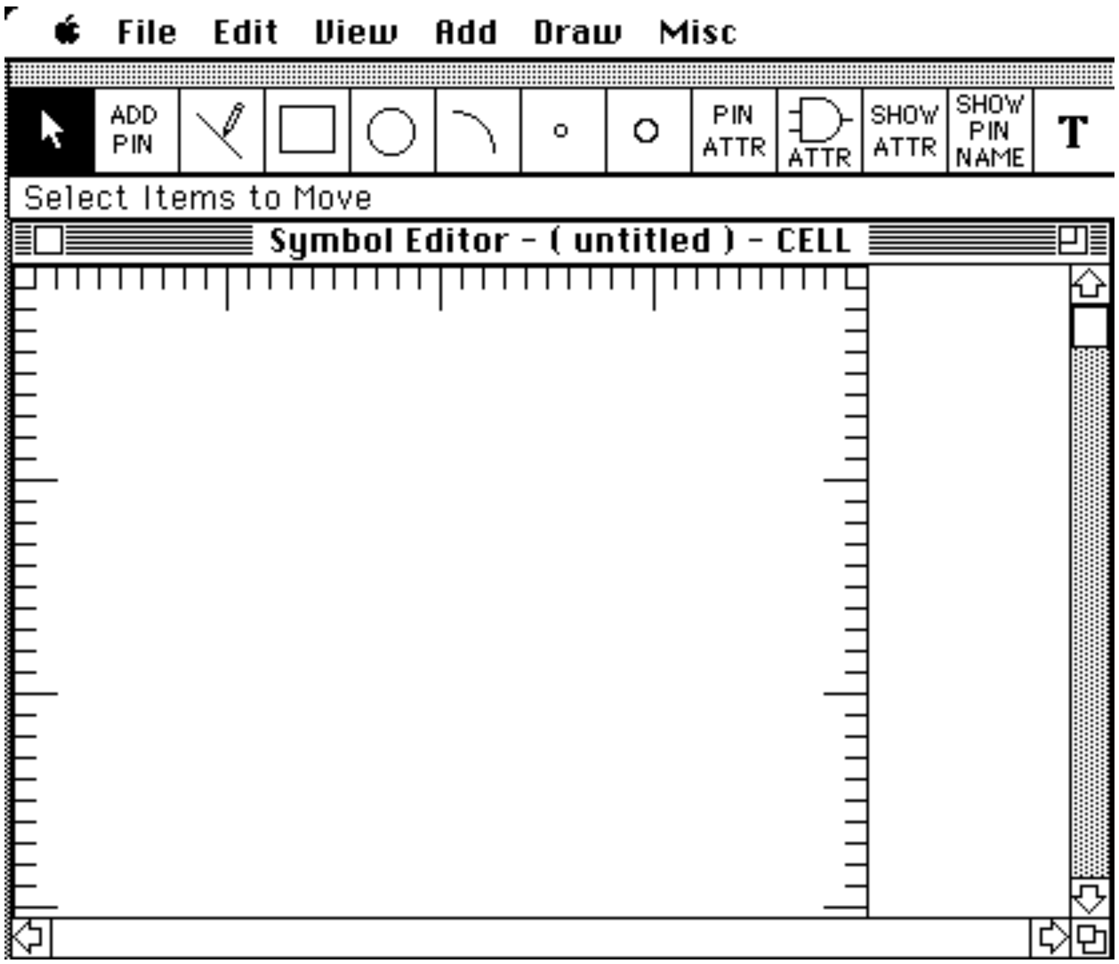


Figure 3-7 Symbol Editor

The window contains a square with tick marks around its perimeter. The marks show the location and size of the main grid. The box represents a square of 40 x 40 grids that are typically set at 0.1 inch spacing (or 2.5 mm).

The **View** commands change the magnification and position of the picture in the window. Select the **View->Window** command.

To try this command:

1. Move the cursor to the upper-left corner of the window.
2. Click the mouse. The magnification doubles and the location of the cursor becomes the center of the window. The tick marks are twice as far apart. The system avoids displaying area outside the border, so the border seems locked to the upper-left corner.

You can also zoom in to a particular area by dragging a box around the target area. To do this:

1. Select the **View->Window** command.
2. Press the mouse at one corner of the desired zoom area.

3. Drag the cursor to the opposite corner. A box follows the cursor as you drag across the window. This is known as a rubber band box because it stretches to wherever the cursor is.
4. When you release the mouse, the area inside the rubber band box is scaled or *zoomed in* to fit the whole screen available.

The **View->ZoomOut** command decreases the magnification. Try using the **Zoom** commands to adjust the picture so that the top and left borders are visible with about 15-20 grids visible on the vertical axis. On most displays, this is as far as you can zoom in because the system limits the range of magnification. Figure 3-8 shows the Symbol Editor window at the correct scale to begin drawing the AND gate.

It is often helpful to have a grid displayed when you are drawing symbols. A grid can be turned on with the following procedure.

1. Select the **Misc->Graphics Options** command. A dialog box opens with a number of option buttons.
2. Select the *Grid* button under the *Display* heading. Grids appear in the Symbol Editor as soon as this is selected.
3. Close the Graphics Options window.

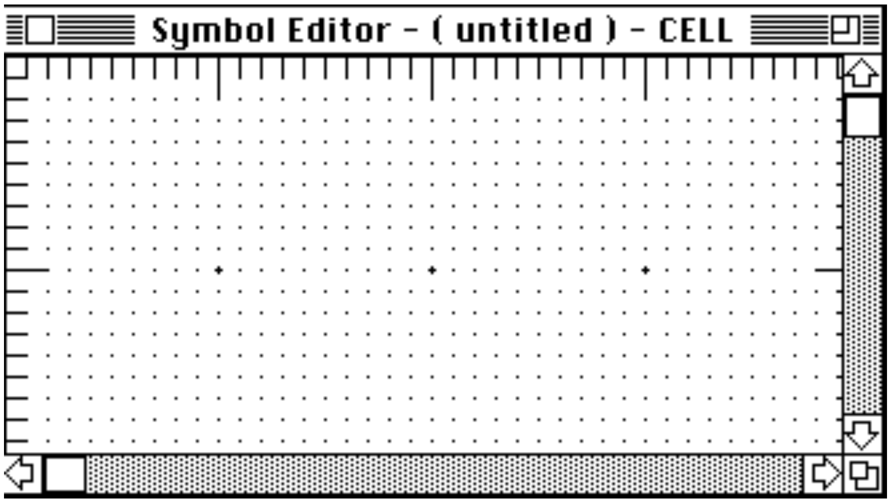


Figure 3-8 Symbol Editor Before Drawing Symbol

Drawing the AND Gate

Use the following procedure to draw the outline of the AND gate.

1. Select the **Draw->Line** command. This is selected from the main menu or from the tool palette. The cursor changes to the + cursor to indicate that a command is active. This cursor jumps from grid to grid. If you leave the drawing area, the cursor changes back to an arrow cursor.
2. Move the cursor to a point about five grids from the upper-left corner.
3. Click the mouse to start the line segment.

4. Move the mouse to the right. A line appears from the original point to the new cursor location.
5. Move the mouse until this line is four grids long, then click the mouse again. The line changes color indicating that the line has been drawn. This command expects a series of line segments and continues to track the mouse.
6. Click the mouse on the end point of the line to terminate the line segment and restart the command.
7. Move back to the start of the line and click the mouse to start the next line.
8. Move down six grids and click the mouse, drawing the vertical side of the symbol.
9. Continue by drawing the bottom of the symbol and then restart the command.

The line is constrained to the eight 45˚ directions. To draw lines at other angles, the drag mode must be used.

Next, draw the input pin leads using the drag mode of the **Draw->Line** command.

1. Move the cursor to a point one grid below the top of the vertical side.
2. Press the mouse and hold it down.
3. Start moving the mouse to the left. The line follows the cursor and is free to form any angle.
4. Move the cursor to the left until the pin leader is two grids long, then release the button. The line changes to the graphics color and the command is ready to start the next line.
5. Add the lower input pin leader one grid above the bottom of the symbol. Pins should be spaced at least two grids apart. Symbols with only two pins on a side usually have their pins spaced further apart.

Figure 3-9 shows the AND symbol at this point.

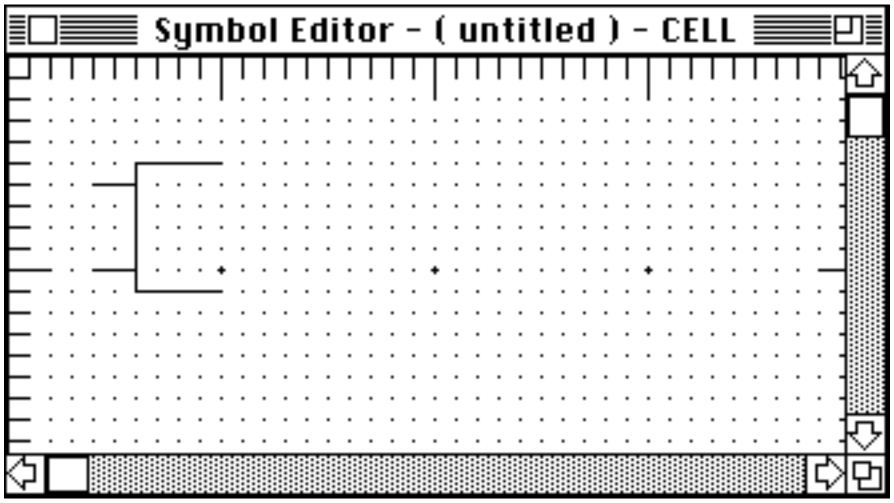


Figure 3-9 Part of AND Symbol

If you made any errors, use the **Edit->Undo** command to back up. You can also use the accelerator key, Command-Z, to select **Undo**. Each time you select this command, you undo one command. If you use **Undo** too many times, the **Edit->Redo** command restores the Undos. Try using these commands to erase the symbol and then restore it.

Finish the symbol outline by drawing the arc for the AND gate.

1. Select the **Draw->Arc** command.
2. Select the start point of the arc by clicking the mouse on the right end of the top line.
3. Click the mouse on the right end of the bottom line to select the end point. As you move the cursor, an arc stretches between the two points.
4. Move the mouse to a point three grids to the right of the end points and vertically centered between them. The arc appears as a semicircle.
5. Click the mouse a third time to fix the arc in position.
6. Use the **Draw->Line** command to add the pin leader for the output pin.

Adding pins is the next step in designing the symbol.

1. Select the **Add->Pin** command. This command is available as an icon.
2. Move the cursor to the left end of the top pin leader and click the mouse. A small square is the mark used to show the pin location.
3. Place pins on the ends of the other two pin leaders.

The names and polarities of the pins are entered as attributes. The following steps explain how to assign the pin names.

1. Select the **Add->Pin Attribute** command. This command is available as an icon. The Pin Attributes dialog box appears.
2. Select the pin that you wish to name by clicking the mouse on the pin. The pin and the line representing it in the left list box of the Pin Attributes dialog box are highlighted.
3. Select the PinName attribute by clicking the mouse on the PinName line in the right list box. See Figure 3-10.
4. Type the name for the selected pin.
5. Pressing the *Enter* or *Return* key advances to the next pin.
6. Enter the names IN1, IN2, and OUT for the three pins. Pin names, net names, and instance names are all forced to be upper case, so it does not matter whether you enter them as lower case or upper case.
7. Select the polarity attribute for the output pin and assign the value OUT. It is not necessary for the input pins to have polarity attributes because IN is the default value.

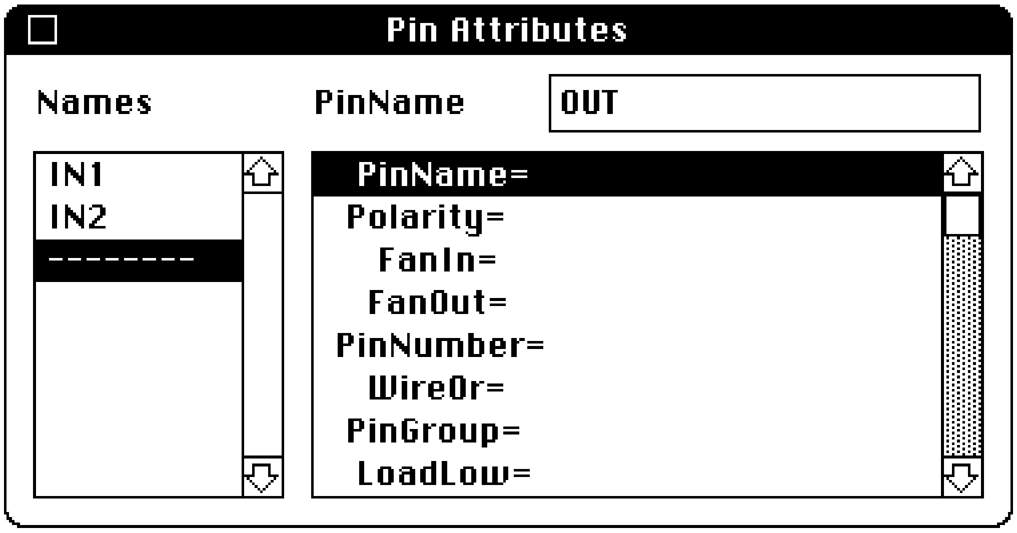


Figure 3-10 Pin Attributes Dialog Box

The last procedure specifies the location where the instance name is to appear.

1. Select the **Add->Show Attributes** command to display the list box containing the symbol attributes that can be displayed.
2. Click the mouse on the line containing *InstName*. As you move the cursor onto the drawing, the word *InstName* is attached to the cursor.
3. Place this name in the symbol by clicking the mouse when the name is in the correct position. When the symbol is placed in a schematic and an instance name is assigned, this word is replaced by the real instance name.

To save the symbol:

1. Select the **File->Save** command. The command pops up the standard file Save dialog box. You should find the folder *reg4* that you created earlier.
2. Type *and*, then click the mouse on the *Save* button.

The AND gate symbol is now complete. As you finish the design, use the standard library symbols for the inverter and OR gate. They are found in the *sym_libs:stduni* library.

The completed symbol should look like Figure 3-11.

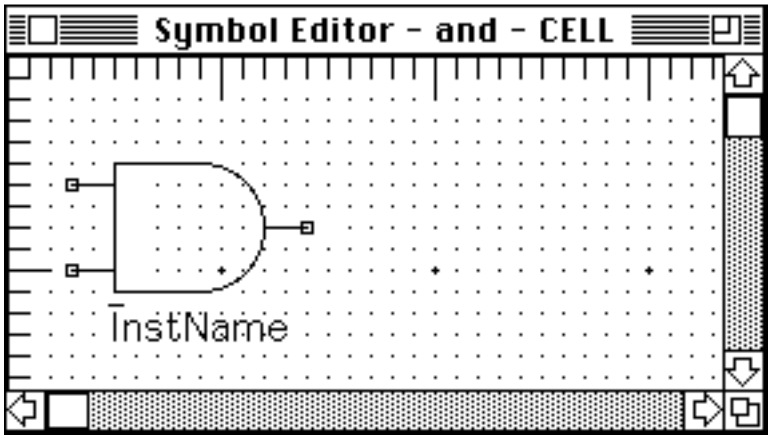


Figure 3-11 Completed AND Gate Symbol

Creating the Latch Symbol

The *latch* symbol is slightly different from the AND symbol. The *latch* symbol represents a schematic in the hierarchy and is not a primitive element. When you create this symbol, you designate it as a Block symbol instead of a Cell.

Use the following procedure to begin the *latch* symbol.

1. Select the **File->New** command to start creating the *latch* symbol. The Symbol Editor returns to its initial state.
2. Select the Block type for this symbol.
3. Use **View->Window** to magnify the upper-left corner of the box as you did for the AND symbol.

Drawing the Symbol

Use the following procedure to draw the symbol for the *latch*.

1. Select the **Draw->Rectangle** command. This is also available in icon form. Click the mouse five grids from the upper-left corner. As you move the mouse, a box is drawn in the foreground color from the first point to the cursor.
3. Move the cursor down and to the right until you have a box that is eight grids high and seven grids wide.
4. Click the mouse a second time to add the rectangle to the drawing.
5. Draw the pin leaders for the three pins.
6. Add the pins and add their name attributes as you did with the AND symbol. Name the upper-left pin D, the lower-left pin EN, and the right-most pin Q.
7. Add the polarity attribute to the output pin.
8. Use the **Misc->Graphics Options** to select center justification. Add the attribute window for the instance name.

Your symbol should appear as in Figure 3-12.

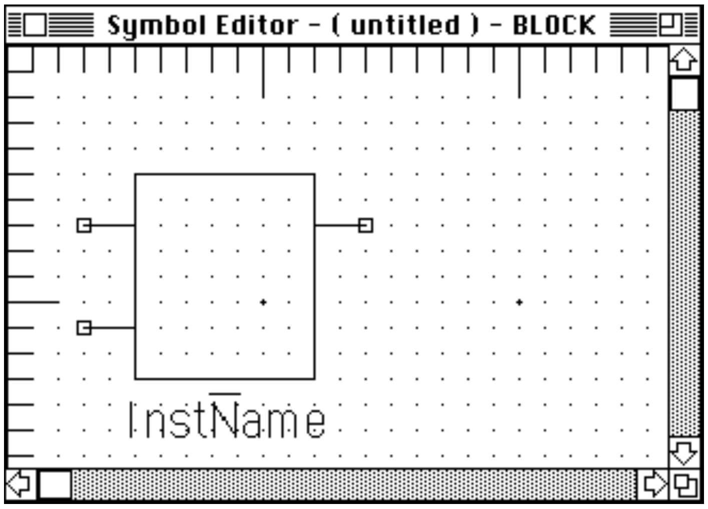
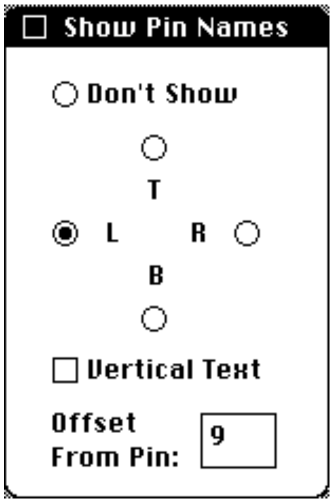


Figure 3-12 Symbol Representing Latch

Displaying Pin Names on Schematic

The following procedure explains how to set up the symbol to display its pin names on the schematic.

1. Select the **Add->Show Pin Names** command. A small dialog box that looks like Figure 3-13 appears in the lower-right corner of the screen. This dialog box is used to specify the location of the name relative to the location of the pin. The four direction buttons refer to the position of the pin on the symbol. Start with the input pins that are on the left edge.



Show Pin Names

○ Don't Show

T ○ · [✓ L ●] · R ○ · B ○

☐ Vertical Text

Offset From Pin: 9

Figure 3-13 Pin Name Location Dialog Box

2. Select the left direction button by clicking the mouse on it.
3. Specify the offset between the pin and the name in the Offset field by replacing the 5 in the edit field with a 9. The offset is specified in quarter grids. Because the pin is two grids from the edge of the square, an offset of 9 quarter grids places the name just inside the box.
4. Click the mouse on the two input pins and observe that the names D and EN appear.
5. Change the dialog box to select right pins by pushing the right direction button and then click the mouse on the output pin, Q. Figure 3-14 shows the completed *latch* symbol.

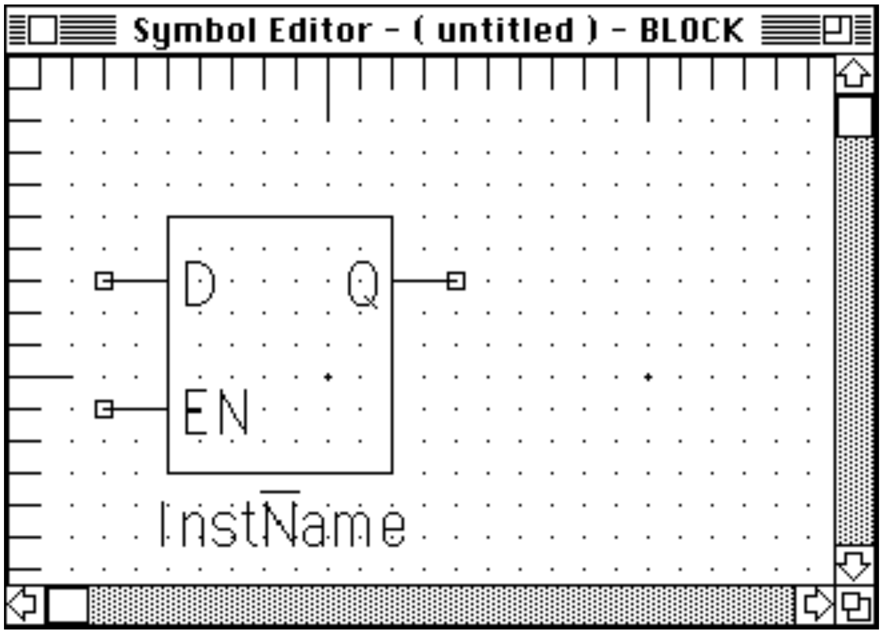


Figure 3-14 Latch Symbol

6. Save the symbol by selecting the **File->Save** command. This symbol is named *latch* when you save it.
7. Select **File->Quit** to exit from the Symbol Editor.

Creating Schematics

The symbols for the register design are complete and you are ready to create the schematics. You create two schematics for this design. The top-level schematic, *reg4*, shows the circuit as four instances of the *latch* symbol. The schematic for the *latch* circuit shows the sub-circuit in terms of its primitive symbols. Start by creating the *latch* schematic.

This procedure explains how to start the Schematic Editor and how to display grids used for alignment.

1. Start the Schematic Editor by double-clicking on the schematic application in the main ECS folder. A blank window is displayed with an untitled schematic.
2. Name the schematic *latch.sch* by using the **File->Save As** command. Save this schematic in the *reg4* folder you created earlier.
3. To display the alignment grid:
    - Select **Misc->Graphics Options**.
    - Choose *Grid* under the heading of *Display*.
    - Close the Graphics Options dialog box.

A sheet border is shown around the outside of the drawing instead of the tick marks used in the Symbol Editor.

Placing Symbols

The first step in building a schematic is to place some symbols. The latch you are drawing consists of two AND gates, an inverter, and a NOR gate. The following procedure explains how to place these on the blank schematic drawing.

1. Select the **Add->Symbol** command. You can click on the unlabeled AND gate icon in the tool palette to select this command. The system prompts you for the name of the symbol to place. It also displays a list box showing the enabled libraries.
2. Type *and*, then press the *Return* key. The AND symbol that you created is attached to the cursor.
3. Move the symbol to the center of the screen and click the mouse. One copy of the symbol is placed on the sheet.
4. Enlarge the AND symbol with the **View** commands:
    - Enlarge the symbol until it is about one inch high and centered on the screen.
    - Click the mouse on the ends of the scroll bars to adjust the position.
5. An alternate way to add symbols is by using the Symbols Libraries dialog box. The upper list box shows the available libraries and the lower box shows the list of symbols in the selected library. The first library shown is the current design folder.
6. When the local design folder is highlighted in the top list box (as it is when you first bring up the symbol selection dialog box as shown in Figure 3-15), you can see the symbols *latch* and *and*, which you just created.
7. Click the mouse on the library named *stduni*.

    This library contains a variety of symbols that are listed in the lower list box.
8. Use the scroll bar on the lower list box to find the OR2 symbol; select it by clicking on its name. The OR2 symbol appears attached to the cursor.

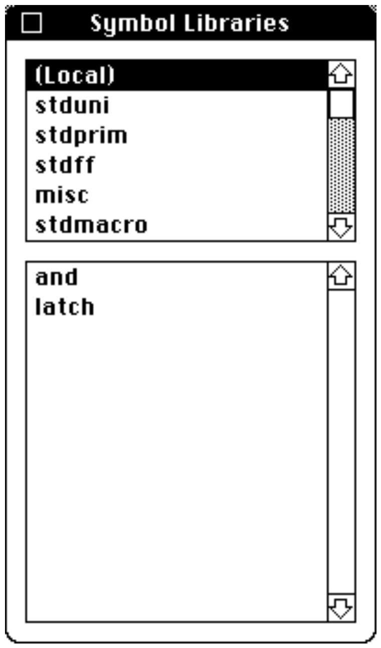


Figure 3-15 Symbol Libraries Dialog Box

9. Place this to the right of the AND symbol.
10. Scroll through the symbol selection dialog box until you find the symbol *inv*.
11. Select and place it to the left of the AND symbol. The **Add->Symbol** command remains active until you select a new command from the menu.

Editing the Schematic

The **Edit** commands are used to align the symbols and finish the symbol placement.

1. Choose the **Edit->Select** command. This command is the arrow on the left of the tool palette.
2. The OR2 symbol is moved by dragging.
3. Move it until it is one inch (10 grids) to the right of the AND gate with its lower input pin aligned with the output of the AND symbol.
4. Release the mouse to fix the OR2 symbol in its new position.
5. Move the inverter one inch to the left of the AND gate with its output aligned with the top pin of the AND gate.
6. Select the AND gate by clicking on it. Choose the **Edit->Duplicate** command. An AND symbol is attached to the cursor, but the original copy remains on the drawing.
7. Place the second AND symbol directly above the first, leaving about one-half inch between them. The drawing should contain four symbols arranged as shown in Figure 3-16.

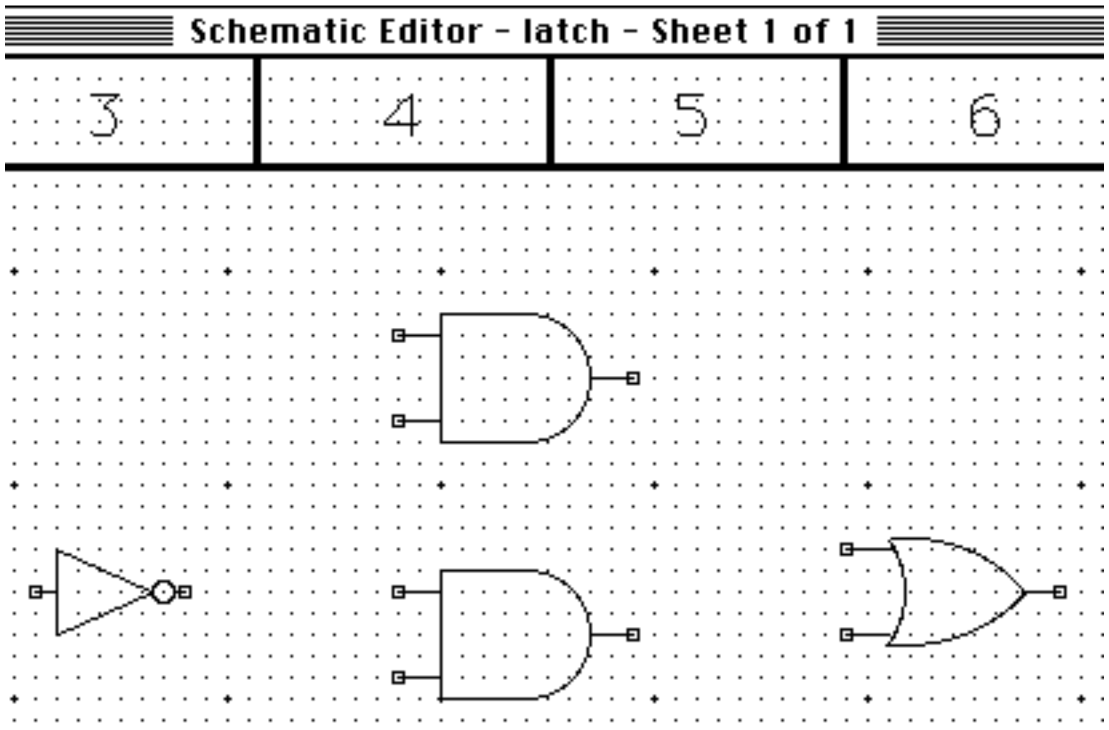


Figure 3-16 Symbol Placement in Latch Schematic

Wiring the Schematic

The next procedure explains how to wire the circuit.

1. Select the **Add->Wire** command, place the cursor on the input pin of the inverter symbol, and click the mouse. This command is also available in icon form.

   A line follows the cursor as you move it on the drawing. This line is constrained to the four 90˚ directions.
2. Move the cursor to a point one inch to the left of the inverter and click the mouse again. A wire is placed on the drawing and the foreground-colored line is now connected to the end of the wire.
3. Click the mouse on the end of the wire to terminate the wire.
4. Repeat the sequence to add a wire to the output of the OR2 symbol.
5. Connect the output of the inverter to the upper input pin of the lower AND symbol.

   The system automatically restarts the command when the wire connects to a pin or another wire.

6. Use the drag mode of this command to make the next connection:
    - Place the cursor on the output pin of the lower AND and press the mouse.
    - Drag the cursor to the input pin of the OR2 and release the mouse. A single horizontal wire segment is placed and the command restarts.
7. Connect the output of the OR2 symbol to the lower input pin of the lower AND symbol by clicking on the starting point and each bend point of the wire.
8. Add a wire to the upper input of the upper AND symbol.
9. The **Add->Wire** command also operates in a three-segment connect mode (also called Z and C connect). Use this mode to make the next connection:
    - Place the cursor on the output of the upper AND symbol and press the mouse.
    - Drag the cursor to the input pin of the OR2 symbol. A diagonal line follows the cursor.
    - Release the button and the diagonal line changes to three connected lines that form a Z pattern.
    - Move the cursor to adjust the pattern with the vertical segment between the two pins.
    - Click the mouse to add the wires to the drawing.
10. Repeat the sequence to connect the lower input of the upper AND symbol to the wire on the input of the inverter symbol. This completes the wiring process. The circuit is shown in Figure 3-17 with the wiring completed.

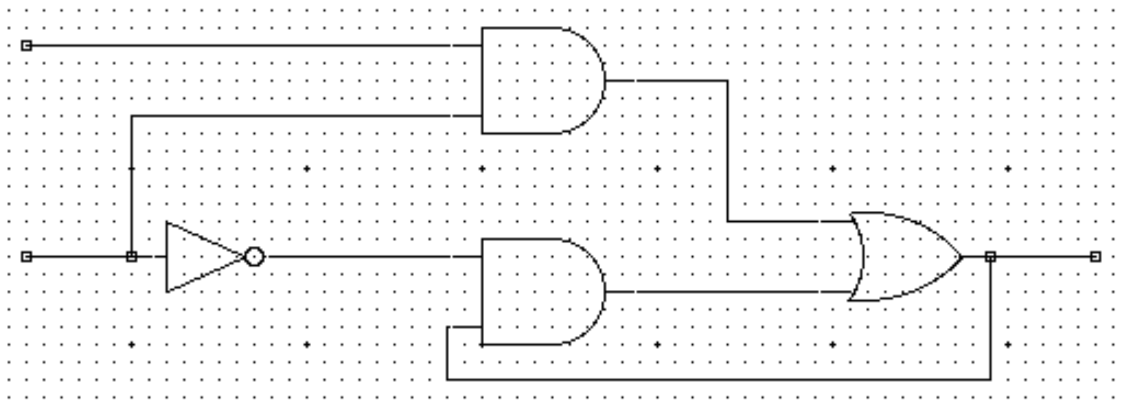


Figure 3-17 Latch Schematic

Naming Symbol Instances

The next step is to name the symbol instances. In this example, use the first letter of the symbol's type followed by a number as instance names.

1. Select the **Add->Instance Name** command. Type *A1+* and press the *Return* key. The system attaches the name *A1* to the cursor. By appending the plus sign (+) to an instance name, ECS automatically increments the instance name on each additional instance name.
2. Click the mouse on the AND symbol and the instance name A1 is assigned to the symbol. The name appears in the window that you defined when you created the symbol. The name A2 is now automatically attached to the cursor.
3. Click the mouse on the other AND symbol to name it.
4. Press the *Escape* key to restart the command. This gives you a chance to change the name.
5. Type *I1* and name the inverter symbol. If the auto-increment character is not specified, then you are prompted for a new instance name.
6. Name the OR2 symbol O1.

Naming Nets

The nets in the circuit can also be named. The three nets that are the inputs and output of the circuit must be given the same names that are assigned in the *latch* symbol.

1. Select the **Add->Net Name** command.
2. Type the name *D* and press the *Return* key to attach the name to the cursor.
3. Position the cursor on the end of the wire that connects to the top input pin on the upper AND symbol and click the mouse. The name becomes attached to the end of the wire and is automatically aligned.
4. Repeat the process to name the input to the inverter symbol with the signal name *EN* and the output of the OR2 symbol with the name *Q*. These names link the pins on the symbol latch to the nets in the schematic. The names on the three internal nets are for user convenience. If you do not name them, the system assigns the names N_1, N_2, ... N_n when the schematic is saved.
5. Use the **Add->Net Name** command to name the three internal nets N1, N2, and N3.

Any primary inputs or outputs to a schematic must be marked with I/O markers. The following steps explain how to attach the markers.

1. Select the **Add->I/O Marker** command. A dialog box pops up allowing you to choose input, output, or bidirectional I/O pins. Figure 3-18 shows the I/O Marker dialog box.

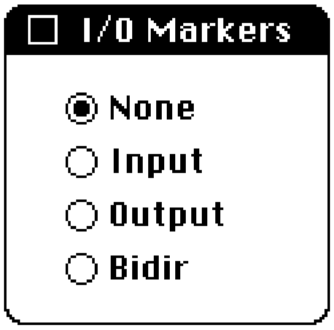


Figure 3-18 I/O Marker Dialog Box

2. Choose input and click the mouse on the two input pins.
3. Change the I/O marker type to output and click on the Q output. The final latch schematic should look as shown in Figure 3-19.

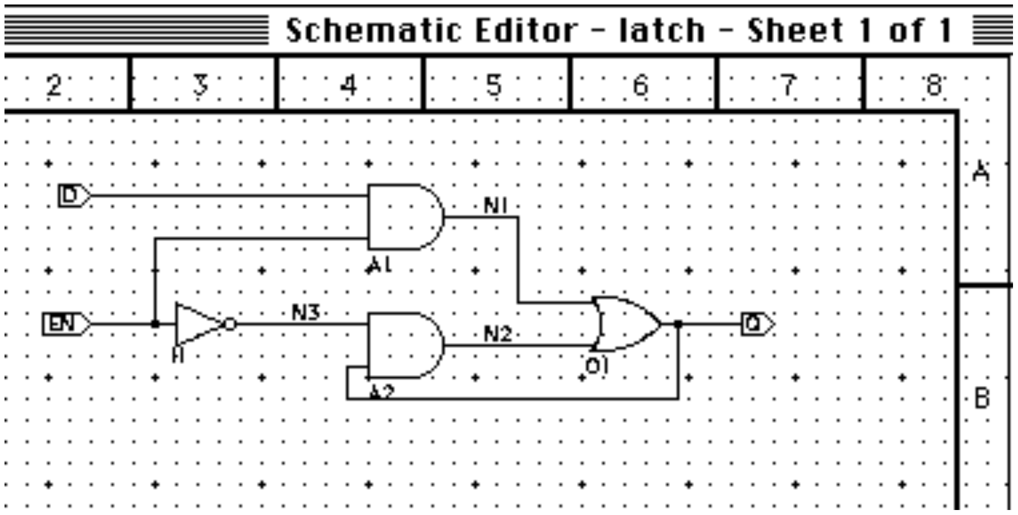


Figure 3-19 Completed Latch Schematic

Saving the Schematic

Use the **File->Save** command to save the *latch* schematic. Using the same name for a symbol and the schematic it represents lets the system know that the two are related. The files in which the symbol and schematic are stored have different file extensions.

Check for Errors

You can check for errors in the schematic at this point using the following procedure.

1. Select **Misc->Error Check**. The circuit is checked for open pins, dangling wires, and other connectivity problems.
2. An error report is generated in a pop-up window. If any specific errors are displayed, you can click the mouse on the error in the pop-up window and

then query the error. The window zooms to the sheet and area where the error is located.

Creating the reg4 Schematic

The next step is to create the schematic for the circuit *reg4*. The following steps are abbreviated and rely on information provided in the previous procedures.

1. Restart the Schematic Editor by selecting the **File->New** command.
2. Place four copies of the symbol *latch*.
3. Wire the schematic, name the nets, and name the symbol instances as shown in Figure 3-20.

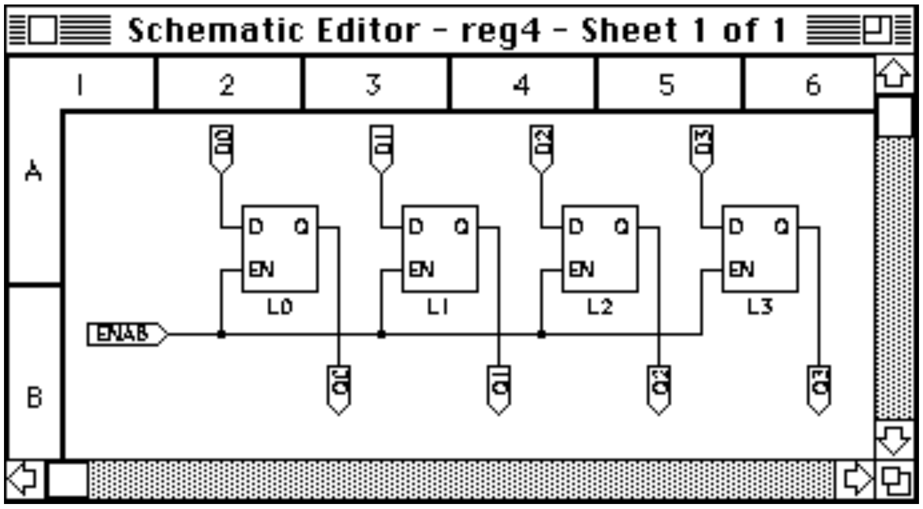


Figure 3-20 Schematic Reg4

4. Save this schematic with *reg4.sch* as the name.

   This completes the creation of the individual symbols and schematics for your simple design.
5. Close the Schematic Editor by selecting the **File->Exit** command.

Now you are ready to combine the symbols and schematics into a hierarchical design using the Hierarchy Navigator.

Navigating the Hierarchy

The Hierarchy Navigator forms the bridge between the individual schematics produced in the Schematic Editor and the production environment, which requires that the whole design be in a single database. The Hierarchy Navigator loads the complete design at once. This includes all schematic sheets and all levels of hierarchy. Having access to the complete design database allows you to trace signals and connectivity between levels of hierarchy.

A hierarchy can be compiled for any schematic. When you compile a new hierarchy, all schematics in the current directory are listed as possible candidates to be compiled into a hierarchical description.

To continue the design example, start the Hierarchy Navigator using the following procedure.

1. Double-click on the Navigator application in the main ECS folder. Find the *reg4* folder containing your design. There are no design files listed in the list box because you have not yet saved the hierarchy for *reg4*.
2. Click the *New* button to indicate that a new hierarchy is to be built. The names of the two schematics, *reg4* and *latch*, appear in the list box.
3. Select *reg4* as the root schematic of the hierarchy by clicking on the *reg4* entry and then pushing the OK button. The *reg4* schematic is drawn in the window of the Navigator.
5. Use **View->Window** to enlarge the drawing so that the text is readable. The drawing should be identical to the drawing as it appeared in the Schematic Editor.
6. To view the schematic for the latch symbol:
    - Select the **Push/Pop** command from the **Push/Pop** menu.
    - Click the mouse on the first latch symbol to push into the latch schematic.
    - Use **View->Window** to enlarge this schematic so that the text is visible. The name on the title bar is REG4.L0, indicating that you are viewing symbol instance L0 in schematic *reg4*.

      The instance names of each of the symbols in the latch are named with the instance name, L0, as a prefix. This indicates that the symbols are below the top level in the hierarchy. The local nets in the latch schematic also have an instance name prefix, as in L0-N1. The external nets D, EN, and Q are named D0, ENAB, and Q0, which are the names of the nets in *reg4* to which they are connected. Figure 3-21 shows the local net names in the Navigator.

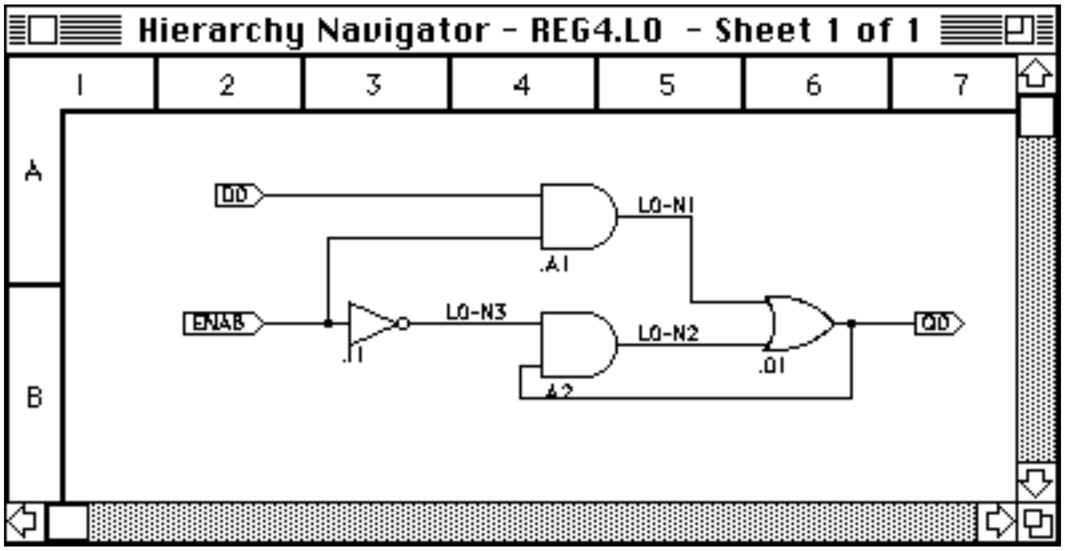


Figure 3-21 Display of Local Net Names in Hierarchy Navigator

7. To display information about a signal:
   - Select the **Misc->Query** command in the **Misc** menu.
   - Query the net D0 by clicking the mouse on the wire. The Query Report shows the list of primitive elements that are connected to this net. In this case, only one pin is attached to this net — the AND symbol's pin, IN1. The pin is named with the hierarchical name of L1.A1-IN1.
   - Click the mouse on the net named ENAB. This net connects two symbols in each of the four instances of latch. The sheet/zone information for the connections made in the current schematic are shown to the left of the pin name. Connections made in other instances of the latch schematic are not labeled with the sheet locations.
8. Select the **Push/Pop** command again and click the mouse anywhere in the schematic except on another symbol. This moves back up to the top level of the hierarchy.
9. Push into the second latch circuit by clicking the mouse on the second symbol. The names have the prefix L1 instead of L0, indicating that this is the L1 instance.
10. To trace a net through the hierarchy:
    - Select the **Misc->Mark** command.
    - Mark the net Q1 by clicking the mouse on its wire. The net is highlighted indicating that the net is marked.
    - Pop back to the *reg4* schematic and notice that the Q2 net is marked on this level too.

- Unmark the Q2 net by clicking on it a second time.
- Mark other nets and use the **Push/Pop** command to trace them through the hierarchy.

Error Checking

ECS provides three levels of error checking. The first level catches errors as you enter data into the system. The second level checks for errors after the individual symbols and schematics are completed. The final level of checking catches errors that are detectable only after the design is complete.

Correct by Construction
Any object you attempt to add to the database is checked for correctness before it is accepted. For example, the system does not allow you to short differently named nets together.

Symbol and Schematic Checking
Both the Symbol Editor and the Schematic Editor have checkers that can check the circuit at any time. They check for such things as:

- Block symbols should have a schematic with the same name present in the current directory.
- Unconnected wire ends are flagged.
- Each symbol instance must be named.

Follow the procedure below to see how this feature works.

1. Edit the *reg4* schematic.
2. The default command after opening a schematic is the **Select** command.
3. Position the cursor over the intersection of the Q1 label and the wire to which it is attached.
4. Press the mouse. This selects the net name flag and I/O marker. Press the *Delete* key on the keyboard to delete the name flag and I/O marker. The **Edit->Clear** command achieves the same function.
5. Use the **View** commands to move the schematic out of view in the window. The window is blank.
6. Select **Misc->Error Check**. A window displays the errors found. One of the errors is *Hanging Wire End.*
7. Click the mouse on this error message. The view is panned so that the error violation is displayed in the schematic window. This feature allows you to quickly locate and fix any errors.
8. Select **File->Quit** and discard the changes when exiting the Schematic Editor.

Design Analysis Tools
The Design Analysis Tools package (DAT) provides checking that can only be performed on the completed design. It also provides a PCB auto-packager and derived attribute capability. It is an optional module that is run from the Hierarchy Navigator. Following are examples of the kind of checking performed.

- Loading or fanout analysis
- Pin assignment for PCB applications
- Connectivity checking

Creating a Netlist

The normal method of transferring a design from one step in the design cycle to the next is with netlists. For example, if you have an in-house timing verification system, you can output a netlist of the design from the Navigator and then read this netlist into the timing verification application.

The Navigator supports several standard or generic netlists. These are:

- Berkeley SPICE (flat and hierarchical)
- HSPICE (flat and hierarchical)
- PSPICE (flat and hierarchical)
- Generic netlist by pin
- Generic netlist by net
- PADS
- Verilog
- Cadnetix
- Silos
- EDIF netlist
- Timemill
- Racal-Redac
- Futurenet

Other netlist formats are supported as described in the *Interfaces* section.

Use the following procedure to generate and look at a netlist of the *reg4* example.

1. Select **Processes->Netlist by Net** from the **Navigator** menu to create a netlist of the circuit.

   When the command is selected, the hourglass cursor appears momentarily. This cursor indicates that the system is busy while the netlist is extracted. When the normal cursor returns, the process is complete.
2. To view the netlist, double-click on the netlist file **reg4.net** after it is created.

The first part of this file provides a list of the primitive elements in the design. The type and instance name of each of the 16 primitive symbols are shown. The second section of the file shows each net with a list of the primitive symbol pins. Review this list with the schematics shown in the Hierarchy Navigator.

Printing

If you have installed and selected a printer on your system, you can print the netlist and the drawings for your design.

To print from the Navigator, select the **File->Print** command. You can modify the printing settings using the **File->Page Setup** command.

Exiting From the System

You have completed the design of the *reg4* circuit. Use the **File->Save** command in the Hierarchy Navigator to save the context of the session. The file that is created has an extension of *.tre* and contains information regarding the marked nets; it also has window scales and offsets for each schematic.

The **File->Quit** command closes the Hierarchy Navigator task. After closing the Hierarchy Navigator, the ECS Executive Program remains.

Summary

During this session, you used the major modules in ECS. In each module, you used some of the commands to create a small design. There are many other commands that are available in the system.

Scan the remaining sections of this manual to become familiar with the available commands. As you work with the system, refer to the manual and the **Help** facility to learn more about each of the commands.

4

Entering Your Design

This section explains in detail how to enter your design into the Engineering Capture System (ECS) This section includes various procedures used in the Symbol and Schematic Editors.

Building the hierarchy of a design using the Hierarchy Navigator is explained. Also covered are the differences in entering a design for an integrated circuit (IC) versus a printed circuit board (PCB) application.

A detailed explanation of attributes in the ECS environment is also provided.

The major topics covered in this section are:

- The ECS environment
- Running utility programs
- Symbol Editor
- Schematic Editor
- Hierarchy Navigator
- Attributes in the ECS environment
- PCB design example

The ECS Environment

ECS is comprised of several core application programs, some utility programs, a number of netlist interfaces, and some standard libraries. You need to understand the relationships between these software programs and how to access the various programs in order to use ECS effectively.

The file structure for ECS consists of a single folder containing the core applications, many utility programs and netlisters, and several folders containing the standard ECS libraries. The sections below describe how these various groups of programs are run.

Before you begin using ECS in earnest, you should read the sections *Before You Begin* and *System Configuration*. These sections guide you in setting up your own ECS system.

ECS Core Applications

The following programs are stand-alone applications that are fundamental to the use of ECS.

ECS Preferences	Configures the ECS design environment
Navigator	Loads complete design hierarchy and facilitates various interfaces
Schematic	Creates schematic files
Symbol	Creates symbol files
Waves	Digital waveform viewing tool (WT module)

They are all invoked by double-clicking either on the application itself or a file created with one of the applications. The icons for the above applications and files created with these applications are shown in Figure 4-1. If you have multiple versions of any of these applications on your hard disk, you should start the application by double-clicking on the application itself. This ensures the proper version of the application is selected.

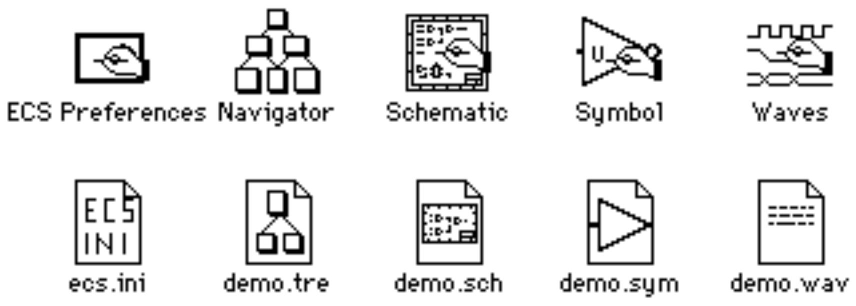


Figure 4-1 Icons for ECS Core Applications

The ECS Preferences Editor is explained in the *System Configuration* section. The configuration information that can be modified with the ECS Preferences Editor include:

- Color assignments
- Sheet sizes
- Processes and Tools included in the Navigator menu
- Graphic display defaults
- Attribute definition
- Search paths for symbols and schematics

Netlist Interfaces and Tools

The following program files are accessed through the **Processes** or **Tools** menu of the Hierarchy Navigator. Additional programs may be added to this list from time to time.

pcbnet -cadnet	Generates Cadnetix netlist
CheckCkt	Checks electrical loading characteristics (part of DAT module)
CheckPCB	Checks reference designators and symbol types (part of DAT module)
EdifNet	Generates EDIF netlist
FutureNt	Generates Futurenet netlist
HspiceNt	Generates hierarchical spice netlist
lister -ListInst	Generates file containing a list of all instances in a design
lister -ListMark	Generates file containing a list of all marked items in a design
lister -ListPart	Generates part list
lister -NetOrder	Generates netlist by net
PackList	Generates a bill of materials
pcbback -Pads	Back annotates data from PADS to design
pcbnet -Pads	Generates PADS netlist for PCB applications
lister -PinOrder	Generates netlist by pin
pcbback -Rinf	Back annotates data from Racal Redac to design
pcbnet -Rinf	Generates Racal Redac netlist
SilosNet	Generates SILOS netlist
SpiceNet	Generates flat spice netlist
TimeNet	Generates Timemill netlist
vericode	Generates verilog netlist
vhdl	Generates VHDL netlist

All of these programs are tightly coupled with your design database as built by the Hierarchy Navigator. They are initiated by selecting them from either the **Processes** or **Tools** menu. The purpose of most of these programs is to transfer the design database to another program such as a simulator or PCB layout program. The netlist interfaces are documented in the *Interfaces* section while the CheckCkt and CheckPCB tools are documented in the *Design Analysis Tools* manual. Additional tools and interfaces will be added as they become available.

Utilities

The following programs are either accessed transparently by the core applications or they can be initiated directly by double-clicking on them and then selecting the file on which they are to operate.

ASCII to SCH	Translates ASCII files into schematic files
ASCII to SYM	Translates ASCII files into symbol files
ECS.INI Maker	Compiles configuration information into binary file
EDIF to Symbol	Translates EDIF files to symbol files
SCH to ASCII	Translates schematic files to ASCII files
SYM to ASCII	Translates symbol files to ASCII files
Symbol to EDIF	Translates symbol files to EDIF files
Update Schem	Updates the time stamps of symbol references in schematic files

These utilities comprise a number of generic translators that can convert between ASCII text files, symbol files, schematic files, and EDIF format files. These interfaces allow interchange of data between different CAD/CAE systems. These interfaces are documented in the *Interfaces* section of this manual.

Symbol files maintain a time and date stamp to keep track of changes. Schematic files keep track of the time and date for each symbol used in that schematic. Whenever a symbol file is changed in the Symbol Editor, any schematic files that reference that symbol become out of date. This flags potential changes in your design or the possibility of having the wrong symbol libraries specified in your search path. The Update Schematic utility loads a schematic, updates any symbols which were out of date, and saves the file.

Naming Design Files

The ECS uses a file-naming convention that allows it to transport files easily between UNIX, DOS, and MacOS environments. When you name design files that may be moved between these different platforms, observe the following rules:

- File names are limited to eight characters that you choose and a three-character file extension automatically chosen by the system. If your design files stay on the Macintosh, normal Macintosh naming conventions apply.
- Alphanumeric characters, the underscore (_), and dollar sign ($) characters are allowed.
- Names cannot start with a dollar sign character ($).

ECS has reserved several file extensions for use by the system. Avoid using these extensions in directories where ECS is used. These reserved extensions are:

._LN	log file extension
._SC	schematic log file extension
._SY	symbol editor log file extension
._WT	log file extension
._WV	log file extension
.asc	ASCII schematic file
.asy	ASCII symbol file
.atr	attribute override file
.bin	binary waveform file
.cdx	netlist for cadnetix netlister
.edf	EDIF netlist
.err	error output file
.frp	Racal Redac netlist
.his	history file for waveform viewer, contains simulation results
.net	SILOS netlist file
.ntl	Timemill netlist file
.pad	netlist for PADS PCB
.pin	netlist file for generic netlist by pin and futurenet netlist by pin
.sch	schematic files for use with the ECS Schematic Editor
.spi	spice netlist
.sym	symbol files for use with the ECS Symbol Editor
.tre	hierarchy navigator files
.ttt	temporary file used by Navigator
.vtr	temporary file used by Navigator
.wav	names of waveforms and trigger information, used by Waveform Tool
.wdl	waveform Editing Tool database file

Crash Recovery

Crash recovery on the Macintosh works in the following way. A log file is created when the Schematic Editor is first started. The name of the log file is *root_design_name._sc*. The schematic editor checks for the existence of this file before opening an existing schematic. If this file exists, it means

- either someone else on the network has opened this particular schematic
- the last time the schematic was open, the Schematic Editor did not exit gracefully (i.e.,. it crashed).

If someone else on the network is editing the file, you should not edit it or you would overwrite each other's changes. In order to prevent this from occurring, ECS creates a warning message that asks you to verify that no one else on the network is editing the target schematic file.

If you answer that no one else on the network is using the file, then it assumes that the log file was left from a Schematic Editor aborted session. The Schematic Editor prompts you asking if you want to recover the file. Responding with a yes causes the Schematic Editor to automatically recover any changes made to the file before the Schematic Editor crashed.

A side effect of this is that crash recovery is not possible on new schematics that have not yet been saved. It is recommended that you perform a save shortly after starting a new schematic.

The ECS Symbol Editor

The ECS Symbol Editor allows you to construct symbols for use in schematic drawings. It facilitates drawing lines and boxes, placing text, and adding information that gives meaning to the symbol in the larger context of a schematic. Symbol information is added in the form of attributes, attribute windows, and pins.

Symbols in the ECS environment are the building blocks used to construct schematics. Symbols can represent various electronic entities, such as capacitors, transistors, CPUs, or whole integrated circuits. A design is composed of symbols that are connected into an electrical network.

This section covers the following material:

- Symbol components
- Different symbol Types
- Grids
- Starting the Symbol Editor
- Saving and printing a symbol
- Drawing graphics and fixed text
- Editing
- Preparing symbols for integration into schematics
- Attributes
- Setting the origin of a symbol
- The Check Utility

Symbol Components

A schematic symbol is composed of three types of elements: graphic entities, pins, and attributes.

Graphic Entities

Graphic entities provide the picture of the symbol that is recognized by the reader of the schematic. These lines, arcs, circles, and text do not have any electrical meaning within the system. They provide a graphic indication of the placement of the symbol in a schematic.

Pins

Pins on a symbol represent points at which a wire can be attached. The pins and wires represent connections between a symbol and other elements in a circuit.

Just as the symbol represents a physical component, the symbol pin represents the physical pin to which a conductor can be attached. If the symbol represents a sub-circuit, the symbol pin represents the connection to an internal net of the sub-circuit.

Buses cannot be connected to pins unless the pin is a bus pin. Bus pins can only be used in Block symbols and Component symbols and you can only connect ordered buses to bus pins.

Attributes

An attribute in the ECS environment is a characteristic or property belonging to or associated with a symbol, pin, or net. ECS attributes can describe the width or length of transistors, the price of a particular chip, the size of a chip or a cell, the number of connections to a block, or the length of time it takes to design a block or cell.

Symbol Types

There are seven classifications that are applied to symbols. They are component, gate, cell, block, pin, master, and graphic. The symbol type affects the way certain symbol attributes are handled when used in a design.

The symbol type is set in one of three ways.

- A default symbol type can be fixed in the configuration file using the ECS Preferences Editor. Any time the Symbol Editor is invoked on a new symbol, the symbol type is automatically the default type specified in the configuration file.
- If the default in the configuration file is set to *none*, then any time a new symbol is created, you are prompted as to which symbol type to make the drawing.
- You can change the symbol type using the **Misc->Change Symbol Type** command in the Symbol Editor.

Component and Gate Symbols
You use these symbol types to design printed circuit boards. Component symbols correspond to complete physical devices. Gates represent portions of the physical devices. A complete 7400 quad NAND gate is represented with a Component symbol, while one single NAND gate in a 7400 package is represented with a Gate symbol. Gate symbols allow you to design a PCB system without making the actual packaging assignments. The package assignment can be performed by a PCB automatic packager.

The pins on Component and Gate symbols are numbered to reflect their actual pin assignments in a physical package. Bus pins are not permitted on Gate symbols.

Cell Symbols
These symbols represent the primitive cells used to design integrated circuits. The pins on Cell symbols are named or numbered for identification in netlists. Bus pins are not permitted in Cell symbols.

Block Symbols
Block symbols are used to build a hierarchical design. A Block symbol represents the schematic at the next-lower level of the hierarchy. The naming of pins in a Block symbol is critical. The pin names are the means by which the Hierarchy Navigator links the pins on the symbol with the nets on the underlying schematic. Bus pins are permitted on Block symbols.

Pin Symbols
Pin symbols represent physical pins on a PCB. For example, you can define a special type of pin for test points and another for edge connector contacts. Figure 4-2 shows two Pin symbols. This symbol represents a test point added to a PCB. The pin symbol includes a *pin* as well as some graphics to indicate the special purpose of the Pin symbol.

Many pin symbols can have the same reference designators. This allows you to spread pin symbols across a schematic (even across sheets) and still group them into the same connector by assigning the same reference designator to the different instances. Each instance of a pin symbol with the same reference designator should have a different pin number.

Figure 4-2 Example Pin Symbols

Graphic Symbols
These symbols are used to add information that is not part of the circuitry. Graphic symbols are ignored when the hierarchy is built. Tables and notes are typical uses for Graphic symbols. There are no pins associated with Graphic symbols.

Master Symbols
Master symbols are used on schematics for title blocks and other tables that are required to appear on every design. They are automatically placed at the time a new schematic is created. There is a parameter in the **Controls** section of the ecs.ini file that sets which master symbols are to be placed.

The origin of a master symbol snaps to one corner of the schematic drawing. The corner is determined by the relationship between the origin of the symbol and the symbol data. If the origin is to the lower right of the symbol data, the master symbol is placed in the lower right hand corner of the schematic. If the origin of the symbol is to the upper left of the symbol data, the master symbol is placed in the upper left corner of the schematic. Similarly for the other two corners.

The origin must be located at least 1 grid units outside of a rectangular area enclosing the symbol data. For example, if you want the master symbol to appear in the upper right hand corner of the schematic, you must set the origin 1 grid units above and 1 grid units to the right of any of the symbol's data.

There are no pins associated with Master symbols.

Creating Symbols

This section describes the basics of creating a symbol. More detail on any of the commands mentioned in this section can be found in the *Command Reference* section of this manual.

Starting the Symbol Editor

The Symbol Editor is started in one of two ways. Double-clicking on an existing design symbol file invokes the Symbol Editor and allows you to start editing the selected symbol file. Alternatively, you can double-click on the Symbol Editor application file to begin editing a new symbol. Once in the Symbol Editor application, you can edit other existing symbols or create additional new symbols.

Figure 4-3 shows the Symbol Editor before drawing begins.

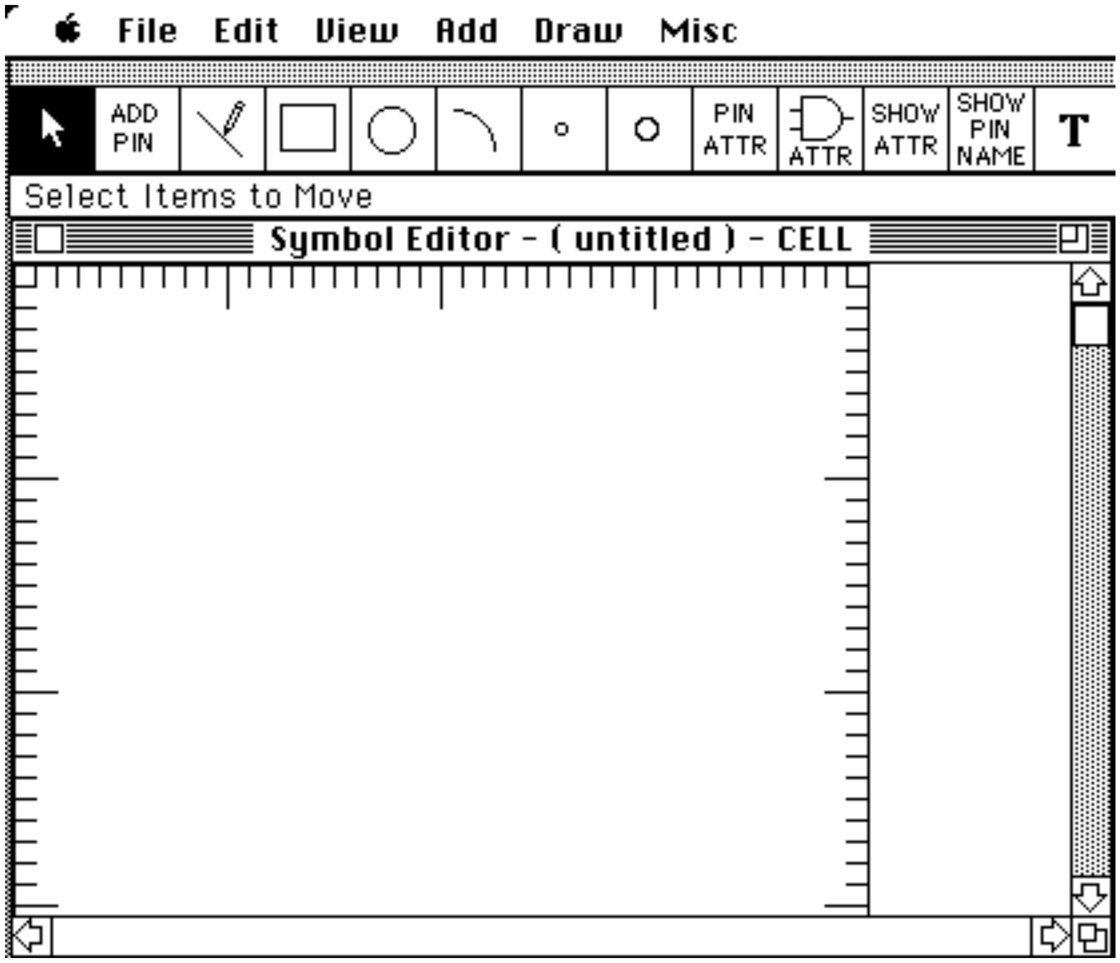


Figure 4-3 The Symbol Editor Before Drawing a Symbol

Grids

Symbols are drawn and stored on a grid. The spacing of the grid is set using the ECS Preferences Editor. The grid is typically set to 1/10-inch spacing. All connectivity information in a design must fall on this major grid.

A secondary grid exists that has four times the resolution of the major grid. Graphics entities can be placed on this secondary grid. This finer resolution allows you to create symbol bodies with more detail.

The **Graphics Options** command under the **Misc** menu is used to access this finer secondary grid. **Graphics Options** can also be used to display the grids while you are creating a symbol. This helps to align features.

The tick marks around the outside of the drawing area in Figure IV-3 are one major grid unit each. The grid spacing can be modified using the ECS Preferences Editor. The size of the drawing area is initially 40 x 40 major grid units. This size can be increased with **File->Expand**. This is necessary if you are drawing large symbols. The largest size available is 1000 x 1000 grids.

Drawing Graphics and Fixed Text

The graphical rendition of the symbol is created with a combination of lines, rectangles, circles, arcs, and fixed text. Graphic objects can be drawn in two weights. The normal lines are the same width as the wires in a schematic. The wide option uses heavier lines that match the weight of the schematic buses. The wide line option can be selected from the **Misc->Graphics Options** menu.

The commands for actually drawing graphics entities are located on the **Draw** menu. They consist of **Line**, **Rectangle**, **Circle**, **Arc**, **Bubble**, **Big Bubble**, and **Text**. These commands are most easily accessed through the Symbol Editor tool palette. Figure 4-4 shows these commands highlighted in the tool palette.

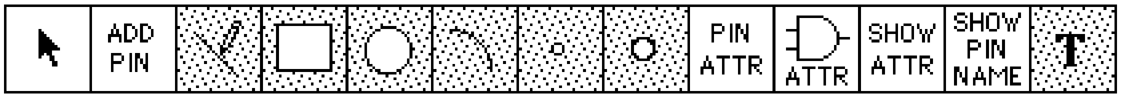


Figure 4-4 Drawing Commands Highlighted in Tool Palette

The **Rect**, **Line**, **Circle**, and **Arc** commands each have several different modes of operation. Arguments for the commands can be entered by separate mouse clicks or with a click and drag of a mouse button. Lines entered with separate mouse clicks are terminated by clicking the mouse twice in succession in one place. Complete descriptions are located in the *Command Reference* section of this manual.

Rectangles
Many symbols are based on a rectangular body. Rectangles are created with the **Draw->Rectangle** command. For symbols that are not rectangular, such as inverters or ALUs, use the **Draw->Line** command to draw the outline of the symbol. Usually

lines are constrained to the three major directions: vertical, horizontal, and 45°. This helps to keep a drawing aligned squarely.

The drag mode operation allows you to draw lines at any angle as long as the end points fall on the grid being used.

Circles and Arcs
Circles are placed with the **Draw->Circle** command. Portions of circles are placed with the **Draw->Arc** command. This is useful for the curved sections of NAND and NOR gates.

Negation Bubbles
Negation bubbles are frequently used in drawing electrical schematic symbols. These bubbles are graphic only and do not have any electrical significance. Two commands that facilitate placing negation bubbles are:

- **Draw->Big Bubble** places a bubble one major grid unit in diameter.
- **Draw->Bubble** places a bubble one-half major grid unit in size.

When either of these commands are chosen, a bubble of the correct size is attached to the cursor. This bubble is affixed to the drawing when the mouse is clicked.

Text
The final graphics entity that can be added to the symbol is text. Text can be added anywhere in the symbol window. Typical uses of text include:

- adding notes pertaining to the symbol
- title blocks

Examples of fixed text on a symbol are:

• Do not exceed 2000 volts ESD on any pin of this device	(note)
• This is test point 32	(note)

Fixed text in a symbol can be drawn in one of three different font sizes. The height of these fonts is either 5, 7, or 9 secondary grids. The text can be justified left, center, or right. This text is always center-justified vertically. The controls for the font size and justification are located in the **Misc->Graphics Options** menu.

Saving a Symbol

Save the symbol by selecting **File->Save**. The symbol definition is saved as an independent file of data. A symbol can be stored in a library for use in many designs, or it can be kept in the design directory for use in a specific design. Updating a symbol causes the new version of the symbol to replace the old version wherever it was used.

If you are saving a new symbol, you are prompted for a name. If you are editing a previously defined symbol, the changes you have made are saved to the already existing file.

You can save the symbol with another file name. This is useful if you are making many symbols that are very similar. You can complete one of them, modify the original, and then use **File->Save As** to save the new symbol under a different name.

Symbol files are named by adding the extension *.sym* to the name of the symbol. This is done automatically by the system when you save a design file of a particular type.

Printing a Symbol

Symbols are printed with the **File->Print** command. This automatically sends a print file to the print spooler. The **File->Page Setup** command allows you to change the orientation (portrait or landscape) of the page, change the sheet size, scale the plot, and choose various bitmap options. Refer to the *Command References* section for more details on printing.

Editing Symbols

It is often necessary to modify a symbol, both as it is being created and some time after it is created. The ECS editing commands provide many ways to modify symbols. In many cases you must first *Select* an object before choosing an action to perform on the object.

To select an object, use the following procedure.

1. Click the mouse on the select arrow at the left of the tool palette.
2. Click the mouse (cursor is a small arrow) on the object to be selected. It grays to indicate it is selected. Many objects in an area can be selected by dragging the mouse to form a rectangle around the target objects. Many separate objects can be selected by holding the *Shift* key down while clicking on the target objects.

A brief look at the editing commands and how they are used is presented here. For a more detailed discussion of the editing commands, refer to the *Command Reference* section.

Clear	Deletes previously placed pieces of the drawing without affecting the contents of the clipboard buffer
Move	Moves pieces of a drawing, faster than cut-and paste-sequence.
Duplicate	Copies objects to another place on the same drawing without modifying the contents of the clipboard buffer.
Drag	Stretches existing objects on a symbol drawing. Lines can be lengthened, boxes widened, circles' radii changed and arcs modified. **Edit->Drag** operates on a single object at a time in the Symbol Editor.
Copy, Cut, and Paste	The clipboard allows you to copy or cut material from one drawing or application and paste it into another. Performing a **Cut** or **Copy** place the selected objects onto

the clipboard. Objects in the clipboard can be **Pasted** into the same symbol drawing or a different symbol drawing.

Undo and **Redo**

Edit->Undo allows you to back up the editing process and return the drawing to an earlier state. Any command that changes the database can be undone. Commands like **View**, which do not access the database, cannot be undone. All database changes back to the last time the data were saved can be undone.

If you realize that you used **Edit->Undo** too many times, you can restore the changes with the **Edit->Redo** command.

Symbol Preparation for Schematics

After the outline of a symbol is created, it is necessary to provide the proper hooks to allow the symbol to be recognized and placed in a schematic. These hooks are pins, attributes, attribute windows, and the symbol origin.

Pins

Symbol pins are connection points where wires connect to the symbol when the symbol is placed in a schematic. Pins on a Block type symbol represent a connection from one level of the hierarchy to the level below. Pins on Gate, Component, and Cell type symbols represent the physical connection points on the device (pins or pads).

Pins are the only symbol elements that are restricted to locations on the major grid. This is because pins provide electrical connectivity and anything having electrical connectivity properties must lie on the major grid.

Pins are added to the symbol with the **Add->Pin** command.

1. Select **Add->Pin** from the menu or tool palette (see Figure 4-5).
2. Click the mouse on the spot where you wish to place pin.

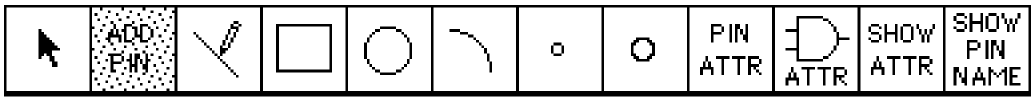


*Figure 4-5 **Add->Pin** Command in the Symbol Editor Tool Palette*

Pins are typically placed at the end of short line segments extending out from the sides of the symbol body. Pin names and numbers are displayed adjacent to the pin on Gate and Component symbols. Block and Cell symbols do not normally have their pin numbers displayed.

To add pin names using **Add->Pin Attribute**:

1. Select **Add->Pin Attribute** from the menu or the tool palette (see Figure 4-6). This displays the Pin Attribute dialog box as shown in Figure 4-7.
2. Select *PinName* from the attributes listed in the right-hand list box.
3. Click the mouse on the required pin.
4. Type the name and press the *Return* key.

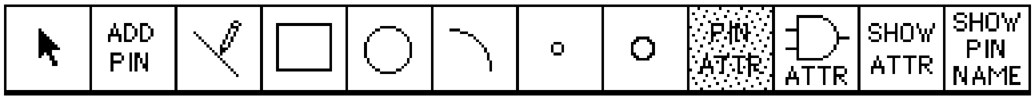
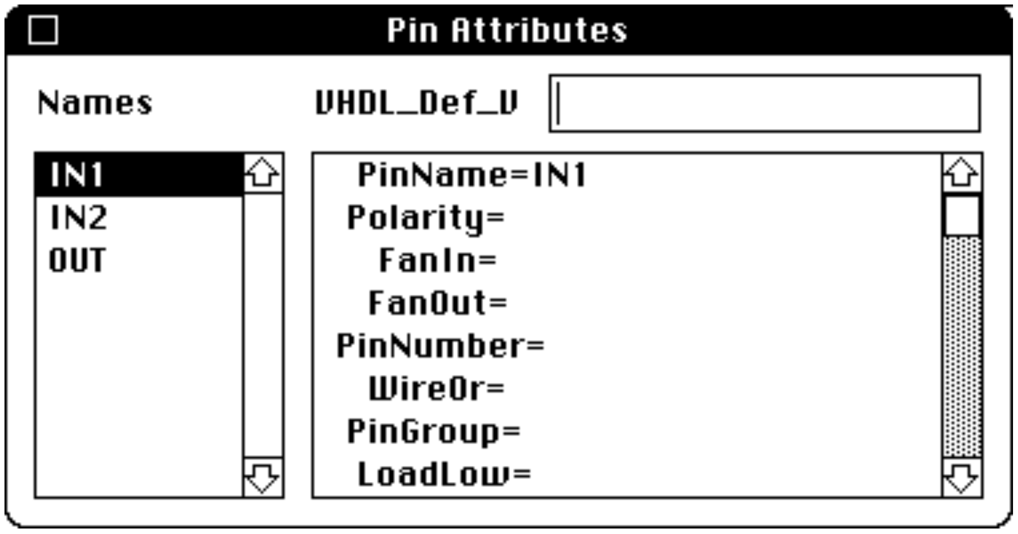


Figure 4-6 ***Add->Pin Attributes*** *Command in the Symbol Editor Tool Palette*

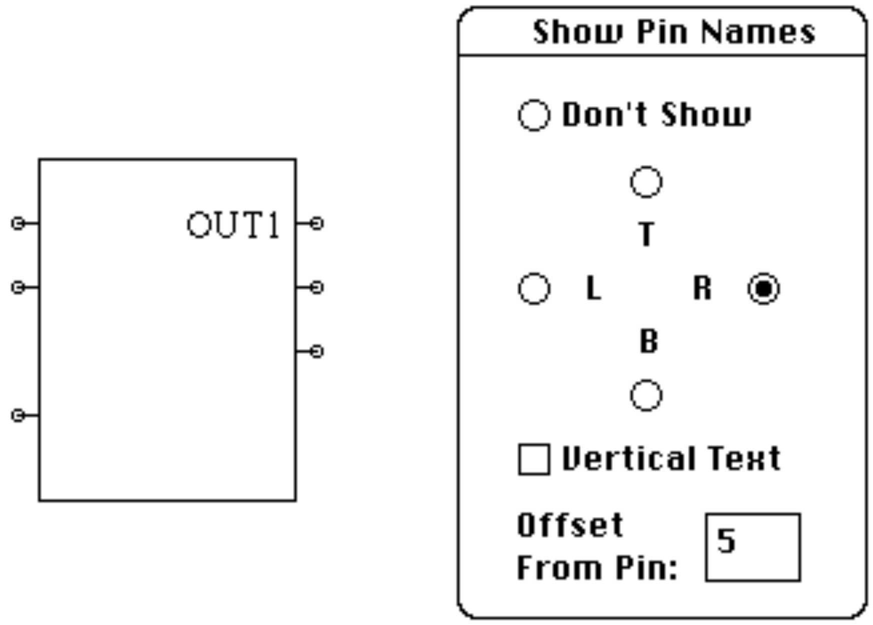


IN1
IN2
OUT

PinName=IN1
Polarity=
FanIn=
FanOut=
PinNumber=
WireOr=
PinGroup=
LoadLow=

Figure 4-7 Pin Attributes Dialog Box

The Show Pin Names option permits pin names to be displayed on Block, Cell, Gate, or Component symbols. The placement of pin names relative to the actual pin is accomplished with the **Add->Show Pin Names** command. You can specify the position of the name as a direction and displacement from the pin itself. The displacement is measured in minor grid units and is limited to a range of 0 to 31 minor grid units. The default value of displacement can be set using the ECS Preferences Editor. Figure 4-8 shows the pin name OUT1 and the settings used to place it.



Show Pin Names

○ Don't Show

○ T

○ L [✓ R]

○ B

☐ Vertical Text

Offset From Pin: 5

Figure 4-8 Pin Name Placement With Right Offset of 5

Bus Pins

A bus pin is necessary any time a bus is connected to a symbol. The number of bits of the bus pin must correspond to the number of bits in the schematic bus that connects to the pin. A bus pin is created by naming the pin as *yourname[index1-index2]*, where *yourname* is the name of an internal bus, and *index1* and *index2* specify the bit range you wish to connect.

Alternatively, a bus pin can be defined by making its name a list of names separated with commas, as in: *name1, foo, mux[0-3], toggle*. This is called a compound name. Compound names allow you to group many unrelated signals together into a bus and route them around the design together. Spaces are ignored in bus names.

Bus pins are allowed on component and block symbols only. When a bus pin is created on a component type symbol, the numbers of the physical pins must be specified as part of the symbol definition.

These numbers are a list of pins assigned to the pin attributes, *BusPin_A* through *BusPin_H*. The normal *PinNumber* pin attribute must be left unused. The list can be spread sequentially through the 8 attributes. Each individual attribute can hold about 200 characters. The list can be comma or blank delimited and can contain sequences of pins by using parentheses ()'s or square brackets []'s to delineate the numeric range.

Examples are:

BusPin_A = 1,3,5,(7:10)	represents 7 pins - 1, 3, 5, 7, 8, 9, 10
BusPin_B = A1 B[2:4] C1	represents 5 pins - A1, B2, B3, B4, C1

Attributes

The symbol and each of its pins can have attributes. An attribute has a name and a value that associates a characteristic with a symbol. The names are represented in the database by an integer. The correspondence between attribute names and these integers is defined in the *Symbol Attributes* and *Pin Attributes* sections of the ECS Preferences Editor. This approach permits the internal numbers to remain constant while the names used for the attributes change to accommodate local practice and language.

The attribute values in a symbol definition are the default values that an instance of a symbol assumes when it is placed in a schematic. These values are frequently overridden in the completed design.

Attributes that apply to all instances of the symbol are generally assigned values when the symbol is first created. For example, the vendor part number and the pin polarity to be used are assigned during symbol creation. Attributes that apply to a single instance of the symbol are assigned after the symbol has been placed in the design. For instance, the instance name and the load on an input pin are assigned to the symbol on the schematic.

For more detailed information about attributes refer to the *Attribute* section.

Pin Attributes

Pin attributes are characteristics or properties associated with a pin. *PinName*, *Polarity*, *Fanin*, *Fanout*, and *PinNumber* are examples of pin attributes. Pin attributes are created using the ECS Preferences Editor and edited in the symbol with **Add->Pin Attribute**. When you select this command, the Pin Attribute dialog box is displayed (see Figure 4-7).

To edit a pin attribute:

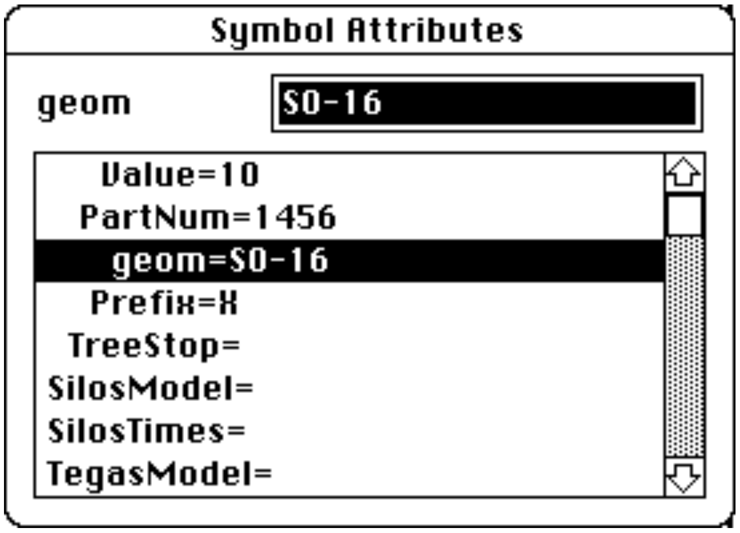
1. Select **Add->Pin Attribute** from the menu. This displays the Pin Attribute dialog box.
2. Click the mouse on the required pin.
3. Select the desired attribute from the attributes listed in the right-hand list box.
4. Type the attribute value and press the *Return* key.

The standard pin attributes, their attribute numbers, and a brief explanation are listed in the *System Configuration* section.

Most of the standard pin attributes are used by simulators or the Design Analysis Tools Checker. These tools use the pin attributes to analyze the circuit.

Symbol Attributes

Symbol Attributes are characteristics or properties associated with a symbol. Examples of symbol attributes are *PartNum*, *Prefix*, *SilosModel*, and *Value*. Symbol attributes are created using the ECS Preferences Editor and are edited in the symbol with **Add->Symbol Attributes**. The Attribute dialog box as shown in Figure 4-9 is displayed when you select this command. The editing procedure is the same as editing Pin attributes.


Symbol Attributes

geom SO-16

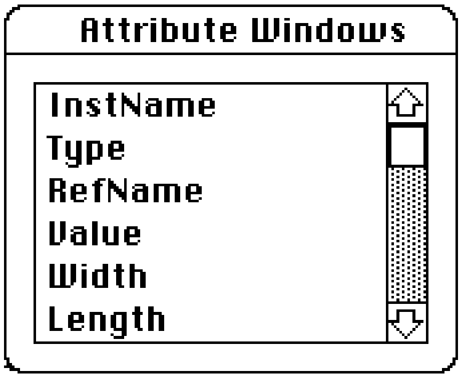
Value=10
PartNum=1456
geom=SO-16
Prefix=X
TreeStop=
SilosModel=
SilosTimes=
TegasModel=

Figure 4-9 Symbol Attribute Dialog Box

The default set of attribute names, numbers, and window assignments in the ECS Preferences Editor is shown in the *System Configuration* section.

Most of the standard attributes are used by the various simulators that interface to the ECS environment.

Attribute Windows
Attribute display windows in a symbol are used to display the symbol's attribute values when the symbol is placed in a schematic. Attribute windows are identified by number. The association between an attribute window and a particular attribute is defined using the ECS Preferences Editor.


Attribute Windows

InstName
Type
RefName
Value
Width
Length

Figure 4-10 Attribute Window Dialog Box

The attribute windows in a symbol are specified by location, text justification, and font choice. The justification determines how the attribute value is displayed on a schematic. The text can be left-justified, center-justified , or right-justified. This is adjusted using the **Misc->Graphics Options** command before placing the attribute window. This text is always center-justified vertically. The text in an attribute window cannot be rotated or mirrored within the symbol. A short line or bar appears both above and below a portion of the attribute window when it is placed on the symbol. This indicates the justification that is used for that window.

Attribute windows are placed on a symbol using **Add->Show Attributes**. The sequence is:

1. Select **Add->Show Attributes** from the menu. A list box appears (see Figure 4-10) containing all the attributes that currently have window numbers defined.
2. Select the attribute from the list box whose attribute window you wish to place. This attribute name is attached to the cursor in the symbol drawing area.
3. Position the cursor with the attribute window attached over the spot on the symbol where you wish the attribute to be displayed.
4. Click the mouse to place the symbol attribute window on the symbol. In the Symbol Editor, the window contains the attribute name. When the symbol is instantiated in a schematic drawing, the attribute window contains the attribute value for that particular instance rather than the attribute name
5. Make sure the attribute window is placed so that the attribute value has enough room to be displayed.

Set Origin

When a symbol is placed in the Schematic Editor, a picture of the symbol is attached to the cursor. The place on the symbol that is attached to the cursor is called the origin of the symbol.

When a symbol is created for the first time, it has no origin. When a symbol with no origin is saved for the first time, the origin defaults to the upper-left corner of the symbol. You can assign an origin or change the current origin with the **Misc->Set Origin** command in the Symbol Editor.

For example, you can change the default origin to coincide with a pin on the symbol. This allows the cursor to be clicked on the target wire in the schematic, rather than somewhere above the target wire as would be the case with the default setting.

To set the origin, select **Misc->Set Origin** and then click the mouse over the desired location. Once an origin is assigned, its location is marked with long tick marks in the pin color along the border of the symbol window.

Check

The **Misc->Check** command in the Symbol Editor finds errors in finished symbols. The error report is written to a file and then displayed in a pop-up text window.

The following types of errors are detected and reported by the **Misc->Check** command.

- Block symbols should have a schematic with the same name present in the current directory.
- Symbols of type other than Block are primitives and should not have a schematic of the same name.
- Each pin must have a *Name* in a Block or Cell and a *PinNumber* in a Gate or Component.
- Every pin in a Gate must have the same number of sections. This means the *PinNumber* attributes must all have the same number of numbers.
- Pins in Component and Pin symbols can only have one *PinNumber*.
- Pins in the same group of a Gate must all have the same *Polarity, Load* and *Drive*.
- Pins on Blocks should not have *Load* or *Drive* specifications.
- Pins on symbols that are not Blocks should have *Load* or *Drive*. Input pins should have *Load* but not *Drive*. Output pins should have *Drive* but no *Load* unless the pin is Tri-state, in which case it should have a *Load* that represents the load in the High-Z state. Bidirectional pins should have both *Load* and *Drive*.

Unconnected Pin Message - Schematic Check Function

The schematic check function may be told to ignore intentionally unconnected pins by setting an attribute on the pin. The attribute is #9 of the Pin Attributes and is named OpenOK. The attribute should be added to the ecs.ini file and set to the "+" option, freely assign.

The attribute can be set in either the schematic or the symbol editor. If set in the symbol editor, the pin is never flagged as unconnected. The attribute can also be selectively set in the schematic editor to disable the message on specific pins but not on all instances of the symbol.

Any value (non-NULL) can be used to inhibit the message. Suggest an attribute value of *Yes, OK, True* as appropriate.

Schematic Editor

The ECS Schematic Editor establishes electrical connectivity between various circuit elements. It allows you to place schematic symbols on a grid and draw wires between the symbols. You can also add attributes to the various components of a schematic drawing. The finished drawing is a schematic representation of an electrical circuit.

When the drawing is complete, you use the Hierarchy Navigator to analyze the circuit and perform simulations. When the simulation results are acceptable, you generate a netlist and send the circuit to be manufactured, either as a Printed Circuit Board (PCB) or an integrated circuit (IC).

This section covers the following material:

- Definition of terms used in the Schematic Editor
- Starting the Schematic Editor
- Grids
- Control Menu
- Multi-sheet schematics
- Drawing graphics and fixed text
- Adding symbols and attributes
- Wiring the schematic
- Editing the schematic
- Connectivity aids
- Saving and printing a schematic
- The Check utility
- The Update Schematic utility

Schematic Components

A schematic is a drawing, consisting of one or more sheets, that represents an electronic circuit. In a simple design, one schematic can represent the entire circuit. As the design becomes more complex, a hierarchy of schematics shows the design in terms of block elements. Each block expands or represents a schematic that contains the primitive elements. As the complexity increases, additional levels of hierarchy are used to describe the design.

Each schematic in the hierarchy is composed of four types of entities: symbols, wires, attributes, and graphics.

Symbols

Symbols placed on the sheet of a schematic represent a component or sub-circuit. Each placement of a symbol is called an instance.

Wires

Wires represent the electrical connections in a schematic. When a wire touches a symbol pin, a connection is made between that pin and all other pins touched by the wire. All of the pins and wires that are connected form a net.

Each net has a name that is either assigned by the user or by the system. When the system assigns net names, it uses a unique name for each net. You can merge two nets by assigning the same name to each net.

A wire that represents a single conductor is called a signal. A group of signals can also be represented by a single wire. Wires representing more than one signal are called buses. Buses are discussed as a separate topic in this section.

Attributes

An attribute is a property that is attached to a pin, symbol, or net. Some attributes, like simulation delays, are assigned to symbols and pins when the symbols are created. Other attributes, such as an instance name, are typically assigned to nets, symbols, and pins after they are added to the schematic.

Graphics

The schematic can contain graphic lines, arcs, circles, and text that are not part of the circuit description. These graphics are included in the drawing to convey information. Title blocks and notes are typical uses for these entities.

Beginning the Design

To enter a schematic, you first start the Schematic Editor. Several display controls can then be configured. The various components described above are added to complete the schematic drawing.

Starting the Schematic Editor

The Schematic Editor is started in one of two ways. Double-clicking on an existing schematic file invokes the Schematic Editor and allows you to start editing the selected schematic file. Alternatively, you can double-click on the Schematic Editor application to begin editing a new schematic. Once in the Schematic Editor application, you can edit other existing schematics or create additional new schematics. Figure 4-11 shows the Schematic Editor when first started.

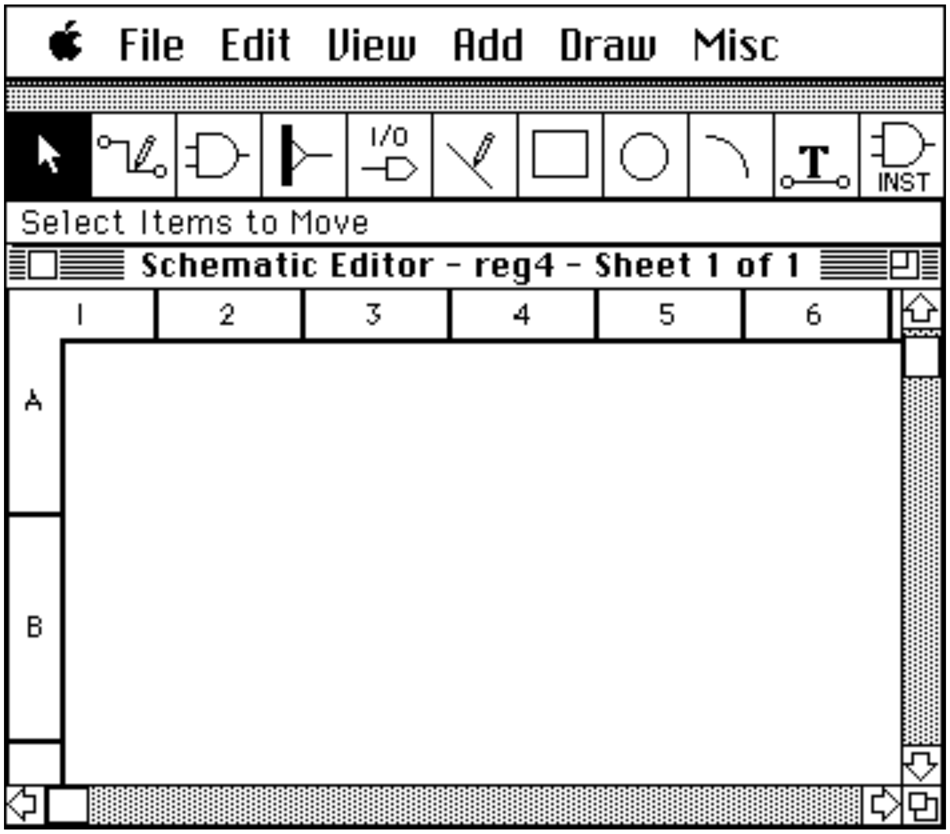


Figure 4-11 The Schematic Editor

Working With Sheets

The **File->Sheets** command allows you to select which of many sheets to view in a multi-sheet design. Several sheets can be open at the same time. Each successive window or sheet examined is opened on top of the previous sheet with enough offset so that some of the previous sheet can be seen. This allows you to move between sheets by bringing the desired sheet to the front.

Up to three views of a single sheet can be opened. This permits working with a combination of an overview and two detail views of the drawing. A total of eight windows can be open in the Schematic Editor at the same time.

Sheet windows can be resized and repositioned as needed. A window that is no longer needed can be closed.

The *Modify* option of the **File->Sheets** command allows you to resize and renumber sheets. The size choices are programmed using the **Controls->Sheet Sizes** section of the ECS Preferences Editor.

Grids

Schematics are drawn and stored on a grid. The spacing of the grid is set using the ECS Preferences Editor. The grid is typically set to 1/10-inch spacing. All connectivity information in a design is required to fall on these grids.

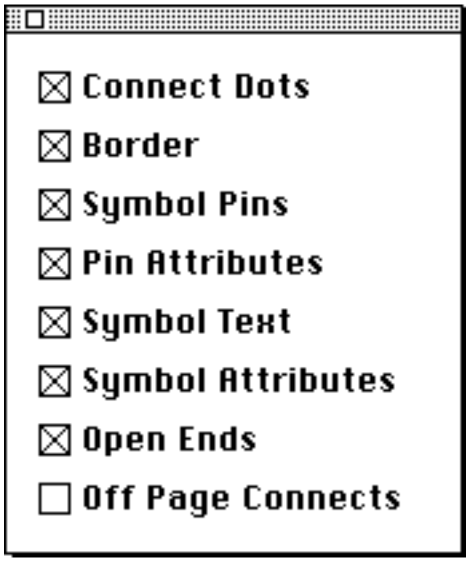
You usually place graphics entities on this primary grid, but you can also place them on a secondary grid that has four times the resolution of the primary grid. This finer resolution allows you to position names and minor embellishments exactly.

The **Graphics Options** command under the **Misc** menu is used to access this finer secondary grid. **Graphics Options** is also used to display the grids while you are creating a schematic. This helps to keep everything aligned.

Controlling Display Options

The **Misc->Display Options** command sets several of the display switches and choices in the Schematic Editor. The changes made with this command last for the duration of the editing session. The next time a schematic or design is invoked, the default values stored using the ECS Preferences Editor are used.

When the command is invoked, a dialog box as shown in Figure 4-12 displays the following options.



☒ Connect Dots
☒ Border
☒ Symbol Pins
☒ Pin Attributes
☒ Symbol Text
☒ Symbol Attributes
☒ Open Ends
☐ Off Page Connects

Figure 4-12 Display Options Dialog Box in Schematic Editor

Connect Dots
Three wires connected to a symbol pin or the junction of four wires are always drawn with a connect dot. This control determines whether the connect dot is also displayed when two wires are connected to a symbol pin or at the junction of three wires.

Border
The outside border is divided into zones. Usually the upper-left zone is *A* on the vertical axis and *1* on the horizontal axis. The numbering of these zones is configured using the ECS Preferences Editor. These zones are used by other ECS functions, such as the **Misc->Query** command, to reference the location of different schematic features.

The sheet border and the location zones can be turned off with this control.

Symbol Pins
Unconnected pins are usually drawn with an error dot to highlight them. This control turns these dots on and off.

Pin Attributes
The display of pin numbers and names is turned off to decrease the repaint time and to reduce the clutter on the screen.

Symbol Text
The display of fixed text in a symbol can be turned off with this switch to decrease repaint time and reduce screen clutter.

Symbol Attributes
The display of text for symbol attribute values is turned on and off with this control.

Open Ends
Wires not terminating on a net name flag, symbol pin, or another wire are considered errors in a schematic drawing. This control turns on an error dot at the end of such wires. It also places a dot on any net name flag that is not connected to a wire.

Off Page Connects
On multiple sheet schematics, you can show references to other sheets with nets that connect across more than one sheet. This control enables the cross-reference display on wire segments with their names at the end of the wire.

Drawing Graphics

Most of the information conveyed in the schematic comes from the symbols and wiring. In some cases, extra graphic information can make the schematic more readable. Commands under the **Draw** menu are used to add graphics that do not contain electrical connectivity information.

Notes, labels, title boxes, and symbolic information are added with the **Rectangle**, **Line**, **Circle**, **Arc**, and **Text** commands available under the **Draw** menu. These commands were described earlier in this section under the *ECS Symbol Editor* discussion as well as in the *Command Reference* section.

Adding Components

After the Schematic Editor is started and initialized with the correct sheet size and configuration information, symbols can be added to the drawing. Symbols represent the electrical components used in the design.

Placing Symbols

Symbols are placed with the **Add->Symbol** command. This command is selected from either the menu or the tools palette as shown in Figure 4-13.

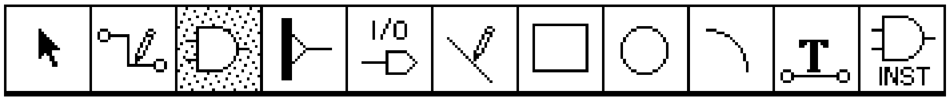


Figure 4-13 ***Add->Symbol*** *Command Highlighted on Tools Palette*

Use the following procedure to add a symbol to the schematic drawing:

1. Select the **Add->Symbol** command. You are prompted to select a symbol from the list box displayed. Figure 4-14 shows the library list box.
2. Type the desired symbol name if you know it. If you don't know the symbol name, use the library list box as shown in Figure 4-14 to browse and select the desired symbol. Libraries are listed in the top half of the list box and the individual symbol files corresponding to the selected library are displayed in the bottom of the list box.
3. The Schematic Editor searches the following places to find the specified symbol:
    - Current directory
    - Project directory search path
    - Symbol library search path

   The search stops as soon as the specified symbol is located. The specified symbol is then attached to the cursor.
4. Move the symbol to the required location and click the mouse to place the symbol.

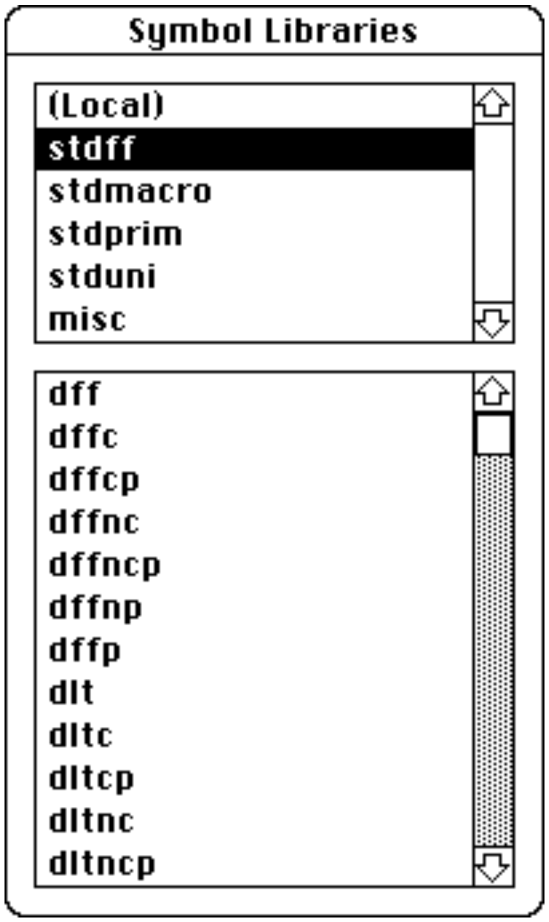


Figure 4-14 Symbol Library Selection Box

To replace an existing symbol, move the cursor so the new symbol overlaps the previous symbol by at least 50%. Click the mouse to delete the old symbol and place the new one. None of the override attributes associated with the previous symbol are transmitted to the new symbol.If the overlap is less than 50%, the new symbol is placed overlapping the original symbol.

Once a symbol is placed, it can be copied, moved, mirrored, or rotated with one of the **Edit** commands. These commands are discussed later in this section.

Type the *Escape* key to reset the command to accept a new symbol for placement or just select a new symbol from the list box.

Adding Instance Names

Once the symbols are placed on the schematic, you can attach instance names, reference designators, or pin numbers. To attach an instance name to a symbol:

1. Select **Add->Instance Name** from either the menu or the tool palette as shown in Figure 4-15. You are prompted to enter an instance name.
2. Type the desired instance name. This is attached to the cursor.

3. Position the cursor at the required instance and click the mouse.

Another method useful for adding sequential instance names (out1, out2, … outn) is to:

1. Select **Add->Instance Name**. You are prompted to enter an instance name.
2. Click the mouse on an existing instance. If the existing instance ends with a number, the number is incremented and the new name attached to the cursor. If the existing name did not end with a number, a numeric suffix is added to the root instance name and attached to the cursor.
3. Click the mouse on the instance to be named.
4. The name attached to the cursor is automatically incremented and is used to name further instances in the sequence. The auto-increment feature is intelligent and skips over names already assigned to other instances. This feature allows rapid naming of instances that are sequenced, as in a bank of eight buffers.

Other methods of naming are described in the *Command Reference* section under the **Add->Instance Name** command.

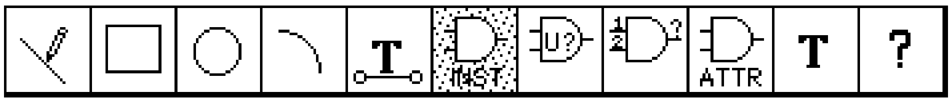


*Figure 4-15 The **Add->Instance Name** Command in the Tool Palette*

Iterated Instances

An array of identical instances can be generated using the **Add->Instance Name** command. This is a useful construct in bit-slice designs or designs where many identical structures are placed in parallel. To create an array of eight inverters, as might be required for an input buffer, make the instance name as follows.

1. Choose the base or prefix name, for example, INV.
2. Add a left bracket, [.
3. Add the starting number for the iteration, for example, 8.
4. Add a delimiter — either a minus sign, -, or colon, :.
5. Add the stopping number for the iteration, for example, 15.
6. Add the matching right bracket to close the expression,].

The finished example of *INV[8:15]* now represents a bank of eight inverters with names of INV[8], INV[9], INV[10]…INV[15].

The system checks before attaching an instance name to a symbol to make sure the name is not already in use. If it is used elsewhere, an error message alerts you, and you can choose another name.

Only block, cell and component symbol instances can be iterated. The count of the instances is specified by assigning an instance name of the form *XX[i:j]*.

Connections to the nets of iterated instances are made according to the following rules:

- a scalar net connected to a scalar pin connects to the corresponding pin on each iterated instance.
- a bus connected a scalar pin connects its n'th net to the scalar pin of the n'th instance. The width of the bus must be the same as the number of instances.
- A bus connected to a bus pin connects each successive pin of the bus to each successive pin of the bus pin. The bus and bus pin must have the same net count.

Assigning reference designators to iterated component symbols is accomplished by providing the list of reference designators as the reference designator attribute on the iterated symbol. The list is a comma or blank delimited list of reference designators. A sequence of reference designators can be included by enclosing the numeric range in ()'s or []'s.

For example, instance name *CC[1:20]* specifies 20 iterations of a symbol. Reference designator *C1,C3,C(5:20),C22,C24* assigned to the symbol causes the 20 iterations to be assigned the reference designators: *C1, C3, C5, C6, C7, …,C20, C22, C24*.

Attributes

The attribute values in a symbol definition are the default values that an instance of a symbol assumes when it is placed in a schematic. These values are frequently overridden in the completed design.

Attributes that apply to all instances of the symbol are assigned values when the symbol is first created. For example, the vendor part number and the pin polarity are assigned during symbol creation. Attributes that apply to a single instance of the symbol are assigned after the symbol is placed in the design. For example, the instance name and the load on an output pin are assigned when the symbol is used.

Attributes are described in the *Attribute* section in this manual.

Editing Pin Attributes

Pin attributes are characteristics or properties associated with pins, such as *PinName*, *Polarity*, *Fanin*, *Fanout*, and *PinNumber*. Pin attributes are created using the ECS Preferences Editor. Pin attributes that have values in the symbol can be edited on the schematic. Attributes that do not have values assigned on the symbol are not accessible at the schematic level unless the attribute is defined in the ECS Preferences Editor as being assignable in the schematic.

Use the following procedure to edit a particular attribute on a given pin.

1. Select **Add->Attribute** from the menu or from the tool palette as shown in Figure 4-16. This displays the Attribute Editor.
2. Click the mouse on the pin to be edited.
3. Select the desired attribute from the attributes listed.

4. Type the attribute value and press the *Return* key.

The standard pin attributes, their attribute numbers, and a brief explanation are listed in the *System Configuration* section.

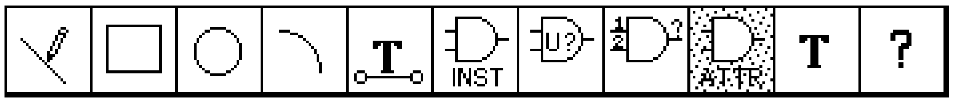


Figure 4-16 ***Add->Attribute*** *Command in the Tool Palette*

Symbol Attributes
Symbol Attributes are characteristics or properties associated with a symbol. Examples of symbol attributes are *PartNum*, *Prefix*, *SilosModel*, and *Value*. Symbol attributes are created using the ECS Preferences Editor. Symbol attributes with values on the symbol can be edited on the schematic. The Attribute Editor is displayed when you select the **Add->Symbol Attribute** command. The editing procedure is the same as described for editing pin attributes.

The default set of attribute names, numbers, and window assignments in ECS Preferences Editor is listed in the *System Configuration* section.

It is possible to change whether an attribute's original value, as defined in the Symbol Editor, can be overridden on the schematic. This is described in the *Attribute* section.

Attribute Windows
Attribute windows in a symbol are used to display the symbol's attribute values when the symbol is placed in a schematic. Attribute windows are created when the symbol is created. In order to display any attribute values on a schematic, an attribute window must be defined on the symbol. Refer to the *Symbol Editor* and *Attribute* sections to see how attribute windows are added to symbols.

Attribute windows can be moved on an instance by instance basis in schematics by using the **Add->Show Attributes** command. This is useful for making a crowded schematic more readable.

Wiring the Schematic

Wires electrically connect the various symbols together. Symbol pins are the mechanism by which electrical connectivity is established between wires and symbols. Connecting a wire to a symbol pin forms an electrical connection between the wire and symbol.

The information below explains:

- how to add wires
- net names
- buses
- bus taps

- reference designators
- pin numbers
- net polarity
- wiring constraints

Drawing Wires

Wires are added to a schematic using **Add->Wire**. The simplest wiring is *Point-to-Point* wiring, as follows:

1. Select the **Add->Wire** command from the menu or from the tool palette as shown in Figure 4-17. The Schematic Editor prompts you to begin the wire.
2. Click the mouse on the first point of the wire. A rubber band wire stretches between the point selected and the cursor.
3. Click the mouse on the second point of the wire. The wire changes from a rubber band to the *wire* color and becomes fixed on the schematic. Additional wire segments are added from this point by clicking the mouse.
4. Click the mouse twice in one place or type the *Escape* key to terminate the wire. The wire also terminates if one of the entry points falls on a pin terminal. The **Wire** command prompts for a new segment until another command is issued.

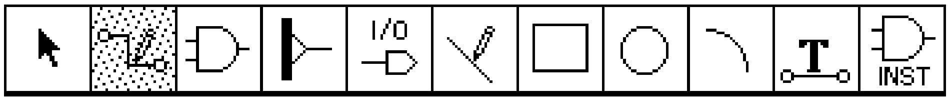


Figure 4-17 ***Add->Wire*** *Command in the Tool Palette*

You can draw 45° wire segments by holding the *Shift* key down while clicking the mouse for the first point of a wire. Z and C connection wiring speeds point-to-point wiring and is described along with other wiring techniques in the *Command Reference* section under **Add->Wire**.

The commands available under the **Edit** menu allow you to modify existing wires.

Net Names

Nets are one of the most important items in the schematic database. All of the connectivity information is carried by the nets. Net names are also important for a number of reasons outlined below.

Identification
Meaningful identifiers assigned to the nets in a schematic document the design and make it easier to understand. If no net name is assigned, the system assigns a unique name of the form *N_nn* where *nn* is an integer.

Interconnection
Non-contiguous wire paths having the same net name are by definition connected together. This automatically connects nets of the same name from different sheets. Wire paths that are either unnamed or named differently are never connected together.

Bus Entry
Wires entering and leaving a bus must be identified with a net name to distinguish them from the rest of the nets in the bus.

Hierarchy Connections
Block symbols are used to simplify schematics by replacing underlying schematics with a single symbol. The connection between the symbol pins and the nets in the underlying schematic are made by having the same symbol pin name and underlying schematic net name.

Ordered Bus Definition
A bus can be defined by naming a net with a compound name. This promotes the wire into a bus containing the nets listed in the compound name.

Simple and Compound Names
A simple name represents a single signal. Examples of simple names are:

READ
WRITE
MYGOODNESSTHISISALONGSIMPLENAME

A compound name is a list containing more than one net name. The compound name can be formed as a list of names separated by commas. For example:

READ,WRITE, MYNAME

represents the three signals READ, WRITE, and MYNAME. All blank characters inside compound names are ignored.

A compound name is also formed when you insert a sequence of numbers into the name. The sequence of numbers is specified as a starting number, an ending number, and an optional increment (default is 1). The numbers are positive integers and are delimited by commas, minus signs, or semicolons. The position of the sequence within the name is indicated by enclosing the numbers with [ ]s, { }s, or ()s and placing them in the desired position. The following are examples of sequential names:

DATA[0-7] represents DATA[0], DATA[1], ... , DATA[7]
ADDR(0,15,2) represents ADDR(0), ADDR(2), ADDR(4), ... , ADDR(14)

The number sequence might not include the ending number if the increment is greater than 1 (as in the second example).

Entering Net Names

Nets can be manually named with **Add->Net Name**. Nets not named in this fashion receive automatically generated names of the form *N_nn*, where *nn* is an integer. This automatic naming of the nets occurs when you *Exit* or *Save* from the Schematic Editor. The **Add->Net Name** command is selected from the menu or the tool palette as shown in Figure 4-18.

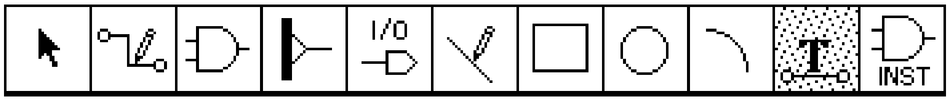


*Figure 4-18 **Add->Net Name** Command in Tool Palette*

The **Add->Net Name** command provides several ways to enter net names:

- You can type a simple name. This name is then assigned to the selected net.
- A name can be copied from an existing net or bus by clicking the mouse on the wire.
- A compound name can be entered using the keyboard. In this case, the complete compound name is attached to a wire selected by clicking the mouse. This creates a bus. The constituent signals of the bus are the individual signals represented by parsing the compound name.
- Individual elements of a bus can be sequentially attached to the cursor by first attaching a bus name to the cursor and then choosing the **Add->Expand Bus** command.

Renaming a Net

Nets are renamed using the following procedure.

1. Select **Add->Net Name**.
2. Type the new net name.
3. Hold down the *Shift* key while clicking the mouse on the net to be renamed. All name flags on all sheets are renamed.

Attaching the Net Name to the Schematic

Once a name or a sequence of names has been specified, the name (or first name in the sequence) appears attached to the cursor. **Add->Net Name** provides four options for placing the net name flag. In each case the command first performs tests to ensure that a valid connection will be made.

- The name flag is placed at an empty point on a sheet. This is considered an error unless a wire is eventually connected to the name flag.
- The name flag is attached to a wire by placing the cursor on the wire and clicking the mouse. A name placed at the end of a wire is left- or right-justified. A name placed in the middle of a wire is center-justified.

- The name flag and a single-wire segment can be simultaneously placed by pressing the mouse and dragging the cursor in either a horizontal or vertical direction to define the wire segment. If either end of the segment connects to a perpendicular wire or bus, a bus tap is created at that end of the segment. If the wire was not a bus, it is promoted into one. Placement of the name flag is determined by the ends of the segment.

  If neither end of the wire segment is connected, the name flag is placed on the starting point of the segment. If both ends are connected, the name flag is placed in the middle of the segment. If one end of the segment is connected, the name flag is placed at the unconnected end.
- A special macro command is built into the system to permit rapid connection of name flags to pins. This macro is activated when a name flag is connected to a symbol pin. Subsequent name flags can be placed by clicking the mouse with the cursor over another pin. A segment of the same length as that previously created is added with the name flag attached as above.

In each of these cases, a verification feature is activated when the mouse button is pressed. If the cursor is on a net, the wires of that net are drawn in the highlighted color while the mouse button is pressed.

When the name has been placed, the **Add->Net Name** command recycles to wait for the next net name.

Buses

Buses group several signals into a single wire. This improves the readability of the schematic.

Buses are of two types: ordered and unordered. An ordered bus uses a sequential name. Examples of ordered buses are:

DATA[0-7] represents DATA[0], DATA[1], ... , DATA[7]
ADDR(0,15,2) represents ADDR(0), ADDR(2), ADDR(4), ... , ADDR(14)

Unordered buses are a collection of unrelated signals that are grouped together. Examples of unordered buses are:

CS,RD,WR represents CS RD WR
CS, DATA(0;7) represents CS DATA(0) DATA(1) … DATA(7)

As the second example shows, unordered buses can contain ordered buses.

Buses can connect directly to bus pins on symbols. This is only possible for Block and Component symbols. For block symbols, the number of signals or bits attached to the bus I/O marker in the underlying schematic must match the number of signals or bits attached to the bus pin in the top-level schematic.

Bus Taps

Another way to make connections between buses and symbols are with bus taps. A bus tap allows a single signal or bit to be *tapped* off a bus. This single signal can then be connected to an ordinary symbol pin in the normal fashion.

The **Add->Net Name** command is often used to create bus taps because it permits the tap, a wire, and the net name flag to be placed simultaneously. To create a bus tap using the **Add->Net Name** command:

1. Select the **Add->Net Name** command.
2. Enter a net name that is contained in the bus to be tapped.
3. Press the mouse where the tap should go.
4. Drag the cursor to the bus and release the mouse. The tap is made and the free end is labelled with the signal name.

The **Add->Bus Tap** command also creates a bus tap that connects to an existing signal or is named at a later time. Bus taps can only be made on vertical or horizontal sections of a bus. To create a tap using **Add->Bus Tap**:

1. Select the command **Add->Bus Tap** from the menu or from the tool palette as shown in Figure 4-19.
2. Position the cursor on the bus where the tap is required.
3. Press the mouse and drag a wire perpendicular to the bus.
3. Release the mouse button when the wire is the correct length.

If a tap is made from a wire, that wire is promoted to a bus.

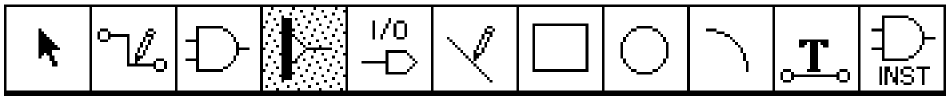


*Figure 4-19 **Add->Bus Tap** Command in the Tool Palette*

Implicit Net Connections

Implicit net connections are made when two or more wires with no physical connection have the same net name. This results in each piece of wire belonging to the same net. This is how connections are made between different sheets on a schematic.

This method provides a third method of connecting buses to symbols. A wire segment attached to a symbol pin is named with one of the signal names used in a bus. This results in a connection from the bus to the symbol pin even though there is no physical wire connection.

You can easily find implicit connections to a net by using the **Misc->Query** command to query a net you are interested in. All wires with the same name are highlighted on the schematic.

Specifying Polarity

An I/O marker is a special indicator that is attached to a net name flag to identify it as an input, output, or bidirectional signal. This determines the polarity of a net and makes the schematic easier to understand.

The Schematic Error Checker also uses I/O markers to ensure that the polarity of marked signals agrees with the polarity of any attached symbol pins. Any discrepancies are flagged by the Schematic Error Checker. Discrepancies in polarity are also flagged each time you start the Navigator.

To mark a net as input, output, or bidirectional:

1. Select the **Add->I/O Marker** command either from the menu or from the tool palette as shown in Figure 4-20.
2. Select the appropriate polarity from the pop-up dialog box.
3. Place the cursor on the net name flag at the end of a horizontal or vertical wire segment or bus. Click the mouse.

Several markers can be placed at one time by dragging a box around the wire ends that are to receive markers.

An I/O marker can only be added to net name flags that occur on the end of a horizontal or vertical segment of wire.

A net can only have one I/O marker per sheet.

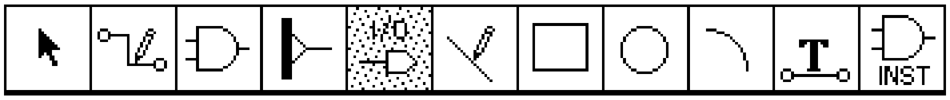


Figure 4-20 ***Add->I/O Marker*** *Command in the Tool Palette*

Pin Numbers

Pin numbers are used in PCB applications to indicate how the pins on a Gate or Component symbol correspond to each physical pin on a physical device. You only need pin numbers if you are doing PCB design.

The **Add->Pin Numbers** command of the Schematic Editor assigns pin numbers prior to packaging or layout. If the design contains hierarchy, the Hierarchy Navigator must be used to add the pin numbers; otherwise the Schematic Editor can be used. The command is selected from the menu or from the tool palette, as shown in Figure 4-21.

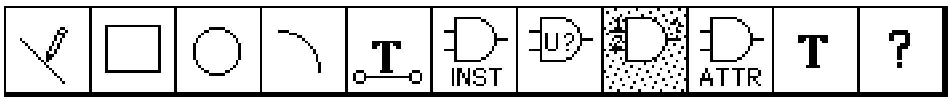


Figure 4-21 ***Add->Pin Numbers*** *Command in the Tool Palette*

Gate symbols represent one section of a physical device that has more than one section. Component symbols represent a complete physical device where the pins are fixed and cannot be interchanged with different pins of the same functionality in the component. Gate symbols have pin numbers assigned that are actually a group of numbers for each pin function. For instance, the symbol for the 7400 series NAND Gate has pins with the following names and numbers:

Name= IN1	Number= 1 4 9 12
Name= IN2	Number= 2 5 10 13
Name= OUT	Number= 3 6 8 11

This is a Gate symbol (:sym_libs:ttl:7400.sym) that represents one-quarter of a complete 7400 component. The symbol has four sections called *A*, *B*, *C*, and *D*. This is shown in Figure 4-22.

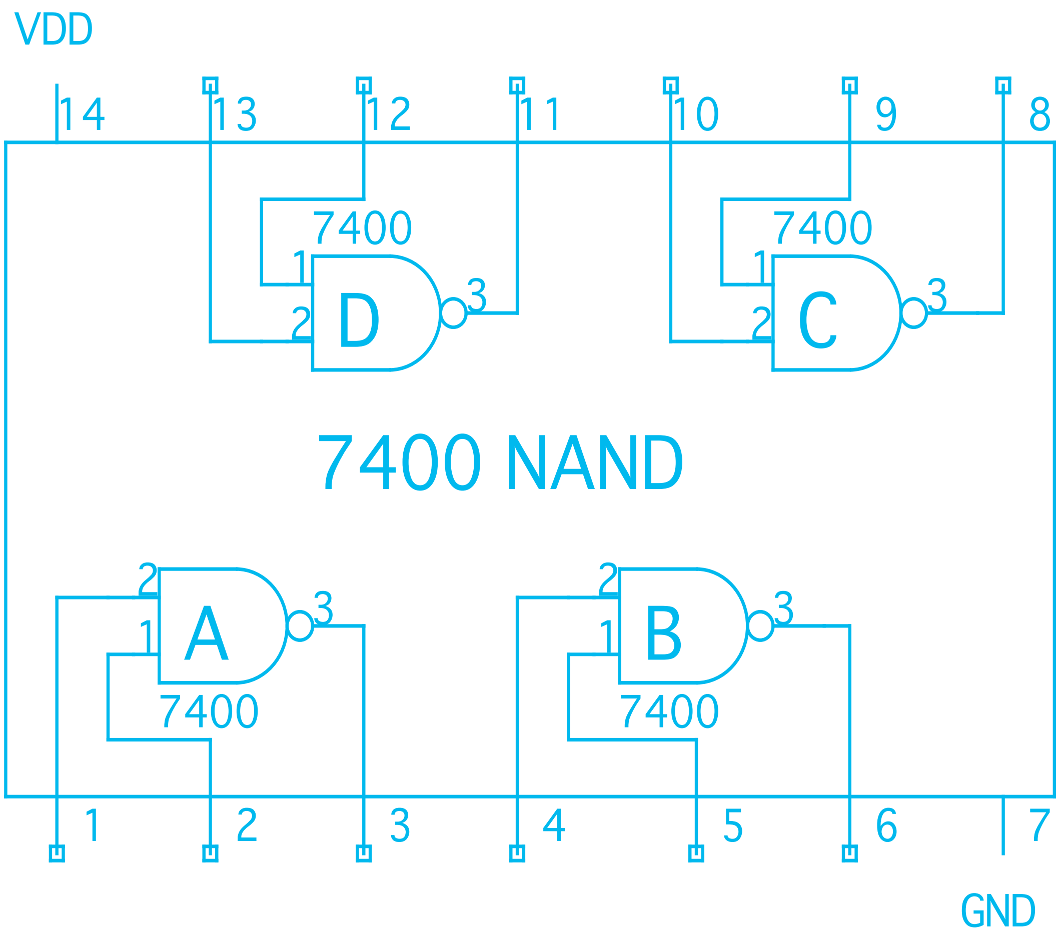


Figure 4-22 Gates in a 7400 Symbol

The numbers on the pins of Gate symbols are initially set to the first section in the physical device. **Add->Pin Numbers** changes the pin numbers to another section and can reflect a permutation of functionally equivalent pins.

To change the section of a gate, enter the pin number for any unique pin in the section and click the mouse within the outline of the symbol to assign the set of pins for that section. Alternatively, enter a letter corresponding to the required section (*A*, *B*, or *C*) and click the mouse within the outline of the symbol.

To swap two functionally equivalent pins, enter the desired pin number for one of the pins and click the mouse on that pin. The system checks the *Pin Group* attribute. If both pins have the same *Pin Group* (and if the symbol is a Gate, belonging to the same section), the selected pin is swapped with the pin that already has the given pin number.

Some Gates have common pins. For example, the 74175 quadruple D flip flop has a common clock and clear. The pin number attribute in this case has four identical entries as below.

Name= ENAB	Number= 4 4 4 4

These common pins cannot be swapped.

The **Add->Pin Numbers** command of the Schematic Editor allows you to click on an instance to pick up the current gate letter or to click on a pin to pick up the pin number. This makes it easier to copy the gate or pin from one instance to another.

Reference Designators

Reference designators designate the physical devices represented by the symbols used in PCB applications. You only need reference designators if you are doing PCB design. The Design Analysis Tools package has special commands for automatically assigning reference designators.

The **Add->Ref Designator** command of the Hierarchy Navigator and Schematic Editor assigns reference designators prior to packaging or layout. If the design contains hierarchy, the Hierarchy Navigator must be used to assign the reference designators; otherwise the Schematic Editor can be used. An auto packger is available in the Schematic Editor if you are licensed for the Deisgn Analysis Tools (DAT). The command is accessed through the menu or from the tool palette, as shown in Figure 4-23.

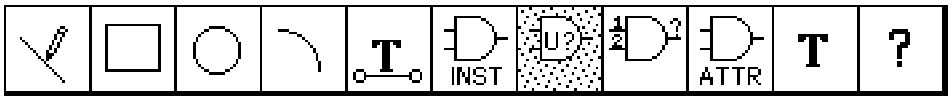


Figure 4-23 ***Add->Ref Designator*** *Command in the Tool Palette*

The reference designator must be unique for each Component symbol in the schematic. The same reference designator can be assigned to two different Gate symbols if they are the same type of symbol. This permits several symbol instances to represent different sections in the same physical part. The special connector Pin symbols (symbols of type *Pin*) are allowed to have duplicate reference designators. This permits edge connectors to have a single reference designator.

DeMorgan equivalent symbols, such as the 7400 and 7400D symbols, are considered to be the same type. That is, they are permitted to have identical reference designator assignments.

Wiring Constraints

There are several wiring constraints that are enforced by the system. Most of these constraints are intended to promote good drafting practice and prevent wiring patterns that could be interpreted incorrectly.

The constraints are:

- All wire segments must terminate on grid points.
- Wire segments must be oriented on the 90° and 45° axes.
- Wire segments cannot form acute angles. This applies to crossing as well as connected wire segments.
- A maximum of two diagonal wire segments can connect at a point. A third orthogonal segment can connect if it does not form an acute angle with either of the other segments.
- Crossing diagonal segments cannot be connected.
- A maximum of three wires can connect at a pin or net name flag.
- A net can have only one name. Two different net name flags cannot be connected by a wire path.
- A net name flag cannot co-exist with a symbol pin. Place the name flag near the pin. Use a short wire segment to connect the name flag to the symbol pin.
- A net name flag cannot be placed in the middle of a diagonal line. It can be placed at the end of a diagonal line.
- A pin or net name flag cannot be placed at the crossing point of two wires, even if the wires are connected.
- A bus can only be tapped on a vertical or horizontal section.
- Only one bus tap can be made at one point on a bus.
- A bus can only contain signals, not other buses. An attempt to name a net in a bus with either a compound name or the name of another bus is considered an error.
- The relationship between an ordered bus (named with a compound name) and the signals in that bus is maintained. Naming the bus or one of its signals in a way that violates this relationship is not permitted.

Editing the Schematic

It is often necessary to modify a schematic, both as it is being created and some time after it is created. The ECS editing commands provide many ways to modify schematics. In many cases you must first *Select* an object before choosing an action to perform on the object.

To select an object, use the following procedure.

1. Click the mouse on the select arrow at the left of the tool palette.
2. Click the mouse (cursor is a small arrow) on the object to be selected. It grays to indicate it is selected. Many objects in an area can be selected by dragging the mouse to form a rectangle around the target objects. Many separate objects can be selected by holding the *Shift* key down while clicking on the target objects. This is referred to as the group select.

A brief look at these commands and how they are used is presented here. For a more detailed discussion of the commands refer to the *Command Reference* section.

Clear	Deletes previously placed pieces of the schematic without affecting the contents of the clipboard buffer. This command works by clicking on single objects and also by dragging a box around several objects.
Move	Moves pieces of a schematic, faster than cut-and-paste-sequence.
Duplicate	Copies objects to another place on the same drawing without modifying the contents of the clipboard buffer.
Drag	**Edit->Drag** in the Schematic Editor tries to maintain electrical connections while sliding objects on the drawing. A typical application is to select a symbol on the schematic and **Drag** it to new location. The wires slide, stretch, and allow right-angle corners to form, in order that electrical connectivity is maintained.
Rotate and **Mirror**	**Edit->Mirror** and **Edit->Rotate** are associated with the **Edit->Move** and **Edit->Duplicate** commands. Objects (most commonly symbols) that are selected with the **Move** or **Duplicate** commands are rotated or mirrored if **Rotate** or **Mirror** is selected. The rotation is 90° clockwise unless an odd number of mirrors has been previously performed, in which case the rotation is 90° counter-clockwise. After the symbol is oriented correctly, it is placed in the normal fashion, by clicking **M1**.
Copy, Cut, and Paste	The clipboard allows you to copy or cut material from one schematic and paste it into another. Performing a **Cut** or **Copy** place the selected objects onto the clipboard. Objects in the clipboard can be **Pasted** into the same schematic drawing or a different schematic drawing.

Undo and **Redo**

Edit->Undo allows you to back up the editing process and return the drawing to an earlier state. Any command that changes the database can be undone. Commands like **View**, which do not access the database, cannot be undone. All database changes back to the last time the data were saved can be undone.

If you realize that you used **Edit->Undo** too many times, you can restore the changes with the **Edit->Redo** command.

Connectivity Aids

It can be difficult to keep track of the different pieces of a large design when the design covers many sheets. There are several commands that can help you find your way through a schematic.

Misc->Hilite displays the selected net in a different color (or in dotted lines on a black-and-white monitor). If the net is a signal in a bus, the bus is also highlighted. **Hilite** is especially useful in viewing connections of one net on adjacent schematic sheets. More than one net can be highlighted at once.

To remove highlighting from all nets, type the *Escape* key.

Misc->Query is mainly used to interrogate the system for information about circuit elements. **Misc->Query** is selected either from the menu or the tool palette as shown in Figure 4-24.

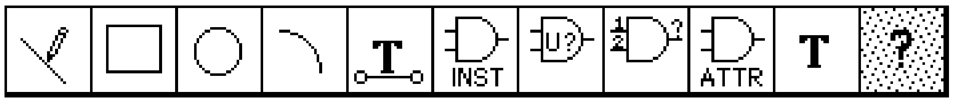
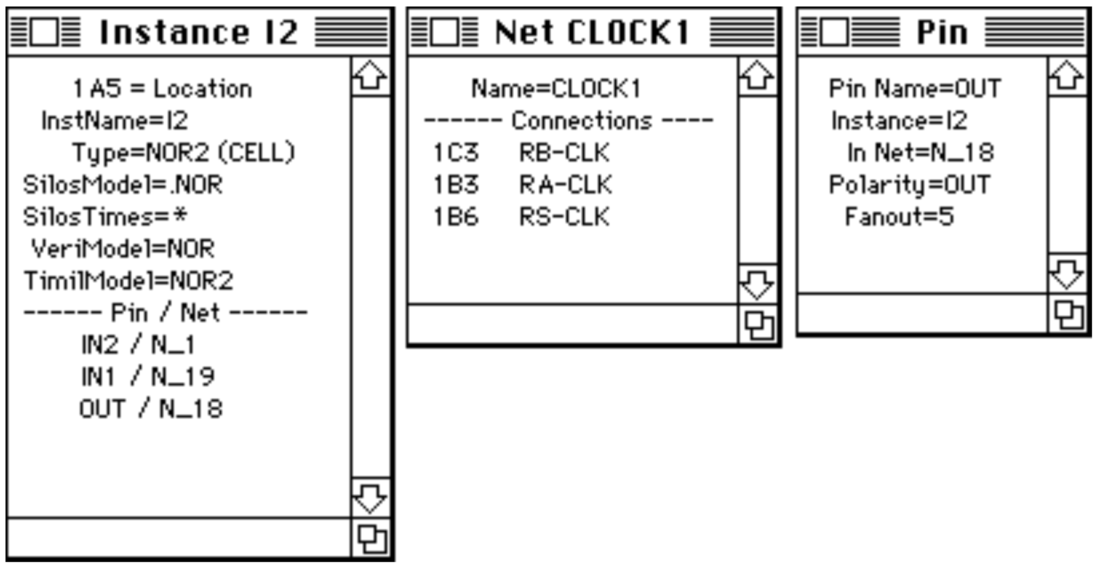


Figure 4-24 ***Misc->Query*** *Command in the Tool Palette*

The requested information is shown in a small window that pops up when the first element is selected. Figure 4-25 shows representative query windows. **Misc->Query** can also be used to trace nets through a schematic because it highlights the items selected.



```
Instance I2
     1 A5 = Location
  InstName=I2
     Type=NOR2 (CELL)
SilosModel=.NOR
SilosTimes=*
 VeriModel=NOR
TimilModel=NOR2
------ Pin / Net ------
      IN2 / N_1
      IN1 / N_19
      OUT / N_18
```

```
Net CLOCK1
    Name=CLOCK1
------ Connections ----
 1C3    RB-CLK
 1B3    RA-CLK
 1B6    RS-CLK
```

```
Pin
Pin Name=OUT
Instance=I2
  In Net=N_18
Polarity=OUT
  Fanout=5
```

Figure 4-25 Query Reports for Symbol, Net, and Pin

Query returns the following information for the indicated circuit elements.

Pins

- pin name
- pin attached to which instance
- polarity

Nets

- net name
- connections
- reference location on the schematic in terms of sheet number
- vertical border reference
- horizontal border reference

Symbols

- the name (for example, NAND2, NOR3) and type (gate, component, block, cell, master, pin) of symbol
- full path showing location of symbol file in the file structure
- instance name
- reference designator
- other symbol attributes
- reference location on the schematic (sheet number, vertical border reference, horizontal border reference, for example, 2A6)
- gate section (A, B, C, etc) on gate symbols
- pin/net connections

Once the query report is visible, you can click the mouse on the various *Connections* entries in the query report to cause the schematic display to zoom to the particular connection selected in the report. This allows you to rapidly examine all the connections made to a net.

For more information, see **Misc->Query** in the *Command Reference* section.

Check

The **Misc->Check** command of the Schematic Editor is used to find errors in finished schematics. The error report is displayed in a pop-up text window. This command is useful for detecting the type of errors that are only meaningful when a schematic is completely specified.

For schematics, the following types of errors are detected.

- Bus taps must be named.
- There must not be any isolated Net Name Flags.
- If the *ShowSymbolPins* option is enabled, each pin on the symbol representing this schematic must be connected to a net.
- If the *MarkOpenEnds* option is enabled, there must not be any wire ends that are not connected to something.
- Only ordered buses can be connected to a bus pin on a symbol.
- Only ordered buses can be marked with I/O markers.
- Nets cannot be marked with more than one I/O marker.
- A bus tap and its bus cannot both be marked with an I/O marker.

If there is a symbol for this schematic, the following errors are detected by the schematic checker.

- Each pin must have a corresponding net with an I/O marker whose direction matches the *Polarity* of the pin.
- Each net marked with an I/O marker must correspond to a pin.

The error report from the Schematic Editor lists the sheet number and zone in front of each error. Clicking the mouse on the individual errors causes the system to zoom to that area of the schematic containing the error. This greatly speeds tracking down errors.

The schematic check function may be told to ignore intentionally unconnected pins by setting an attribute on the pin. The attribute is #9 of the Pin Attributes and is named OpenOK. The attribute should be added to the **ecs.ini** file and set to the '+' option, freely assign.

The attribute may be set in either the Schematic or the Symbol Editor. If set in the Symbol Editor, the pin is never flagged as unconnected. The attribute can also be selectively set in the Schematic Editor to disable the message on specific pins, but not on all instances of the symbol.

Any value (non-NULL) may be used to inhibit the message. Use an attribute value of *yes*, *OK*, *true* as appropriate.

Output the Schematic

This section describes saving the schematic to a file and printing the schematic on a printer. Information on netlisting is also presented.

Saving the Schematic

Schematics are saved with the **File->Save** command. If it is a new schematic, you are prompted for a name under which to save the new schematic. If you are editing a previously defined schematic, the changes you have made are saved to the already existing file.

Schematic files are named by adding the extension *.sch* to the name of the schematic. This is done automatically by the system when you perform a **Save**.

You can also save a schematic with another file name. This is useful when you are making many schematics that are very similar. You can complete the first one, then use **File->Save As** to save the schematic under different names.

The schematic definition is maintained as an independent file of data. A schematic can be stored in a project library for use in many designs, or it can be kept in the design directory for use in a specific design. Updating a schematic causes the new version of the schematic to replace the old version wherever it was used.

Printing the Schematic

Schematics are printed with the **File->Print** command. This automatically sends a print file to the print spooler. The **File->Page Setup** command allows you to change the orientation (portrait or landscape) of the page, change the sheet size, scale the plot and choose various bitmap options. Refer to the *Command References* section for more details on printing.

It is also possible to print a portion of a schematic with the **File->Print Window** command. You can enclose part of the schematic inside a rubber band box and print this piece.

Netlisting

The normal method of transferring a design from one step in the design cycle to the next is with netlists. A netlist is a description of all the components placed in a design along with all the connectivity information. You must use the Hierarchy Navigator to generate a netlist of any schematics you enter. Refer to the *Hierarchy Navigator* section for more details.

Creating Symbols in the Schematic Editor

The **File->Create Symbol** command provides a quick and efficient method of creating generic Block symbols while running the Schematic Editor.

The graphics for these symbols consist of a rectangle with pin leads extending outward. The height and width of the symbol are adjusted based on the number of pins and the length of their names. The input pins are lined up on the left side and the output pins are lined up on the right side. The length of the pin leads is based on the value of the *DefaultPinNameOffset* parameter of the ECS Preferences Editor.

The symbol also contains an attribute window for displaying the symbol name near the top of the symbol and another attribute window for displaying the instance name near the bottom of the symbol.

When the **Create Symbol** command is invoked, a dialog box allows you to specify the symbol name, input pins, output pins, and bidirectional pins. The names for the pins are separated by commas.

The button, *This Block*, allows you to create a symbol that represents the current schematic. This button is especially useful when the input nets and output nets are already labelled with I/O markers. Pushing the *This Block* button causes the system to fill in the edit fields with the necessary information. The *Name* field is filled in with the name of the schematic.

Inputs	is filled in with the names of all the nets marked as inputs with I/O markers.
Outputs	is filled in with the names of all the nets marked as outputs with I/O markers.
BiDirs	is filled in with the names of all the nets marked as bidirectional with I/O markers.

To add a bus name to one of these fields, the bus name must be enclosed between equal signs. If a compound name appears in the list of pins and is not surrounded by equal signs, it is expanded to produce individual pins for each name. If the name is enclosed by equal signs, it produces a bus pin. For example:

Name entered into input field	Pins Created
A,B[0;3],C	creates 6 pins A B[0] B[1] B[2] B[3] C
A,=B[0;3]=,C	creates 3 pins A B[0;3] C
=A,B[0;3],C=	creates 1 pin A,B[0;3],C

When all the information is entered into the edit fields, the symbol can be placed or edited. To edit the symbol with the Symbol Editor, push the *Edit* button. To place the symbol without editing, press the *Run* button. This builds the symbol defined and invokes the **Add->Symbol** command with the newly created symbol on the cursor.

The Update Schematic Utility

Symbol files maintain a time and date stamp to keep track of changes. Schematic files keep track of the time and date for each symbol used in that schematic. Whenever a symbol file is changed in the Symbol Editor, any schematic files that reference that symbol become out of date.

The Hierarchy Navigator displays warning messages about schematics that are out of date. Normally, if only minor changes are made to the symbol, the warnings can be ignored. If a significant change has been made to a symbol (such as a new pin, new pin name, or new symbol origin) then any schematics that contain that symbol must be edited to accommodate these new changes.

If the changes to the symbol do not alter any pin placements, the **Utility->Update Schematic** command in the ECS Executive Program can be used to update the schematic.

The **Update Schematic** command loads a schematic, updates any symbols that were out of date, and saves the file.

Hierarchy Navigator

The Hierarchy Navigator forms the bridge between the individual schematics produced in the Schematic Editor, and the production environment, which requires that the whole design be in a single database. The Hierarchy Navigator loads the complete design at once. This includes all schematic sheets and all levels of hierarchy. Having access to the complete design database allows you to trace signals and connectivity between levels of hierarchy.

The Hierarchy Navigator performs three functions:

- Verifies a design's connectivity after the design is entered with the Schematic and Symbol Editors
- Optimizes and analyzes the circuit's performance using simulation and analysis tools
- Prepares the design data for the next step in the design process

The Hierarchy Navigator loads existing schematics. It cannot create new schematics. You can, however, use the Navigator to modify the attributes of any instance in a design. This attribute override information is stored along with the scales and view windows in a Navigator *.tre* file. This file has the extension *.tre*.

Figure 4-26 shows the icon for the Navigator and a *.tre* file created by the Navigator.

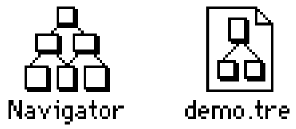


Figure 4-26 Hierarchy Navigator Icon and Icon of Tree File Created by Navigator

The following information explains the basic concepts of the Hierarchy Navigator and then explores the three applications.

Navigating the Design

You can start the Hierarchy Navigator on a design by double-clicking on the design's *.tre* file. This assumes that the design has already been loaded once and you have saved a *.tre* file. Use the following procedure to load the design into the Navigator if a *.tre* file does not exist.

1. Double-click on the Hierarchy Navigator application. A file open list box appears as shown in Figure 4-27, allowing you to select a design to open. The files in the list box default to *.tre* files when the application is first started.
2. Click the mouse on the *New* button in the dialog box. The names of all the schematics in the current directory appear in the list box.

3. Double-click the mouse on the schematic you wish to edit. A Navigator window opens displaying the chosen schematic. All schematics below the chosen one in the hierarchy are also loaded. Use the **File->Save** command to save the *.tre* file for this new design.

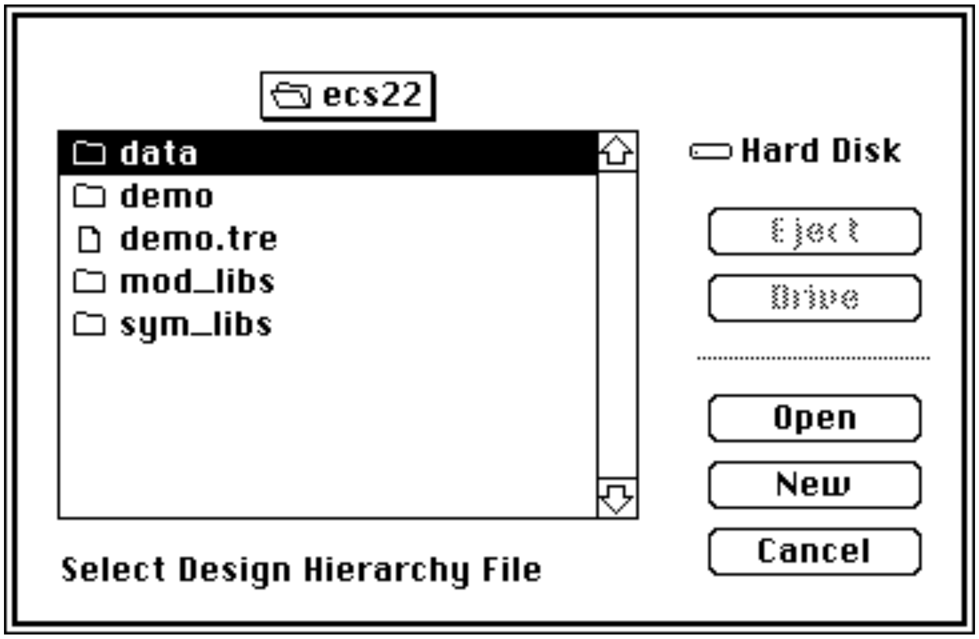


Figure 4-27 File Open List Box in Hierarchy Navigator

The Navigator display looks like the Schematic Editor but you cannot alter any of the symbols or connection wires.

Loading a design in the Hierarchy Navigator reads in all *.tre* files from all underlying designs. This allows attributes to be added to design sub-hierarchies and have those be active when the whole design is loaded.

The **View** commands and the **File->Sheet** command allow you to traverse a design laterally, at one level in the hierarchy. The **Push/Pop** command allows you to traverse the depth of a design by penetrating through the different levels in the hierarchy. After pushing or popping to a new level in the hierarchy, the **View** and **Sheet** commands allow you to examine that level.

Push/Pop

The **Push/Pop** command is used to traverse the design hierarchy. The **Push/Pop** command is selected from the menu or from the tool palette as shown in Figure 4-28.

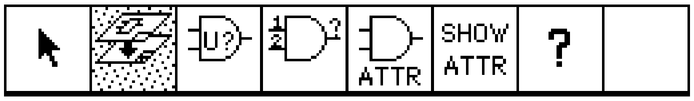


*Figure 4-28 **Push/Pop** Command in the Tool Palette*

The operation of the **Push/Pop** command is described below.

Push moves to a lower (more detailed) level in the hierarchy. To push into a symbol, click the mouse within the symbol boundary. If the symbol represents a lower-level schematic, that schematic replaces the schematic currently on the screen.

Pop moves to a higher level in the hierarchy. To pop back to the parent of the current schematic, click the mouse outside all symbol boundaries. The parent schematic replaces the schematic currently on the screen unless the current schematic is at the highest level (root) of the hierarchy.

Alternatively, you can type the instance name of the destination schematic at the prompt.

The full context of the hierarchy is maintained during **Push** and **Pop** operations. All net names are shown with full hierarchical formats. Symbol instance names are shown in an abbreviated format that replaces the leading portion of an instance name with a dot. The leading part of the name is always displayed on the title bar of the Navigator allowing you to reconstruct the full hierarchical name. For example, the instance .AB.CD.EF is displayed as .EF in the Hierarchy Navigator. The prefix of .AB.CD is displayed on the top of the Navigator window.

Save

Save records the display context for the schematics that are being viewed. This includes nets that are marked, the level of hierarchy being viewed, or a particular view or enlargement of the drawing. It also saves any attributes that have been added to the hierarchy since the beginning of the work session.

When the Hierarchy Navigator is started, it requires the name of the file containing the root (top-level) schematic. The **Save** command derives the saved file name by replacing the *.sch* extension with *.tre*. Subsequent **Save** commands store the new changes in the *.tre* file.

Sheet

The **File->Sheet** command allows you to select which of many sheets to view in a multi-sheet design. Several sheets can be open at the same time. Each successive window or sheet examined is opened on top of the previous with enough offset so that some of the previous sheet can be seen. This allows you to move between sheets by bringing a required sheet to the front.

Up to three views of a single sheet can be opened. This allows you to work with a combination of an overview and two detailed views of the drawing. A total of eight windows can be open at one time in the Navigator.

Sheet windows can be resized and repositioned as needed. A window that is no longer needed can be closed. Many of the **Edit** commands, such as **Edit->Copy**, work across different windows of the same sheet. This allows you to initiate a selection of data in one window and complete the selection in another window. The rubber banding works intelligently across windows.

Print

The printing commands produce a hardcopy of a schematic. **File->Print** prints the schematic currently being viewed. If there is only one sheet defined, the command prints that sheet immediately. If the schematic has more than one sheet, a sheet selector dialog box is presented. Select the required sheet by clicking the mouse on the corresponding line and then clicking the *Print* button. There is also an *All* button that causes all sheets to be printed.

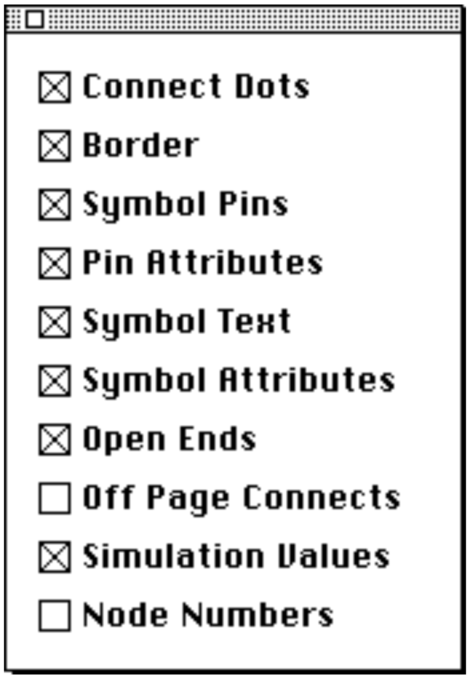
The **File->Page Setup** command allows you to change the orientation (portrait or landscape) of the page, change the sheet size, scale the plot, and choose various bitmap options.

You can use the **File->Print Window** command to print a rectangular portion of a schematic.

Controlling Display Options

The **Misc->Display Options** command sets several of the display switches and choices in the Hierarchy Navigator. The changes made with this command last for the duration of the editing session. The next time a schematic or design is invoked, the default values stored using the ECS Preferences Editor are used.

When the command is invoked, a dialog box as shown in Figure 4-29 displays the following options.



- ☒ Connect Dots
- ☒ Border
- ☒ Symbol Pins
- ☒ Pin Attributes
- ☒ Symbol Text
- ☒ Symbol Attributes
- ☒ Open Ends
- ☐ Off Page Connects
- ☒ Simulation Values
- ☐ Node Numbers

Figure 4-29 Display Options Dialog Box in Hierarchy Navigator

Statistics

The **Misc->Statistics** command summarizes how many elements are placed in the design and how much memory is consumed by different records in the database.

A list of the following primitive elements and the number of each found in the design is generated by the **Misc->Statistics** command:

- types of symbols
- primitive cells
- hierarchical blocks
- instances
- instance pins
- primitive instances
- primitive pins
- nets connected

This is followed by a list of seven types of records used to store information in the database. Each record type is followed by a measure of the capacity used in a particular design. Three of the record types are fixed length and are reported as number consumed, number available, and percentage of the Block that has been consumed.

- instances
- nets
- pins

The remaining four types are attribute records. Because each of these is a variable-length record, only the percentage of the memory block that has been consumed is shown.

- definitions and pins
- net attributes
- instance attributes
- pin attributes

Attributes

You can use the Hierarchical Navigator to override attributes that are assigned in either the Symbol Editor or the Schematic Editor. This allows you to penetrate the hierarchy and assign different values to attributes on different instances.

For example, suppose you have a schematic with two tri-state buffer symbols. One of the buffers needs to be large while the other can be small. You cannot push into the tri-state buffers in the Schematic Editor; however, the Navigator allows you to push into the buffers and customize each instance by changing attributes.

Another reason to change attributes in the Navigator rather than the Schematic or Symbol Editors is convenience. Many design modifications are made based on results obtained from the Navigator's various simulator and other analysis interfaces. It is convenient to make modifications to the circuit from the same environment when the analysis is being performed.

Attributes are described in the *Attribute* section of this manual.

Pin Attributes

Any pin attributes for which values exist in the symbol can be edited in the Navigator. Any attributes that do not have a value assigned on the symbol are not accessible at the schematic or Navigator levels unless the attribute is defined in the ECS Preferences Editor as being assignable in the schematic.

To edit a particular attribute on a given pin:

1. Select **Misc->Edit Attributes** from the menu or from the tool palette as shown in Figure 4-30. This displays the Attribute Editor.
2. Click the mouse on the pin to be edited.
3. Select the desired attribute from the attributes listed.
4. Type the attribute value and press the *Return* key.

The standard pin attributes, their attribute numbers, and a brief explanation are listed in the *System Configuration* section.

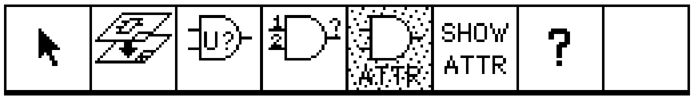


Figure 4-30 ***Misc->Edit Attributes*** *Command in Tool Palette*

Symbol Attributes
Any symbol attributes for which values exist in the symbol can be edited in the Navigator. The Attribute Editor is displayed when you select the **Misc->Edit Attributes** command. The editing procedure is the same as described above for editing pin attributes.

The default set of attribute names, numbers, and window assignments in the ECS Preferences Editor is listed in the *System Configuration* section.

It is possible to change whether an attribute's original value, as defined in the Symbol Editor, can be overridden on the schematic. This is described in the *Attribute* section of this manual.

Attribute Windows
Attribute windows in a symbol are used to display the symbol's attribute values when the symbol is placed in a schematic. Attribute windows are created when the symbol is created. In order to display any attribute values on a schematic, an attribute window must be defined on the symbol. Refer to the *Symbol Editor* and *Attribute* sections to see how attribute windows are added to symbols.

Misc->Show Attributes in the Hierarchy Navigator selects which attributes are displayed in which attribute windows. This is useful if you need to see some attributes that are not currently displayed on the schematic. You can temporarily reassign the attribute window from some other attribute to the attribute you wish to see.

For example, it is common to have attribute windows for the instance name and symbol name. Usually you do not have a window on symbols to display the simulation delay for that symbol. If you need to see all of the symbol delays, you can reassign the attribute window from the attribute *InstName* and have the simulation delays displayed in that spot on the symbol. After you complete your analysis, you can return the attribute windows to their original assignment, or you can save the new assignment.

The command is available from the menu or the tool palette, as shown in Figure 4-31.

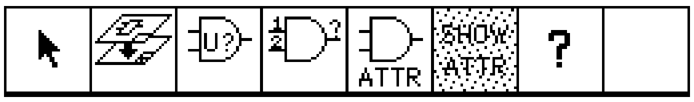


Figure 4-31 ***Misc->Show Attributes*** *Command in the Tool Palette*

Verifying Design Entry

A large percentage of design time is spent correcting errors such as unconnected wires and pins, too much loading on nets, or unwanted connections.

While both the Symbol Editor and Schematic Editor have a number of checks built into them to avoid entry errors, some errors can only be caught in the context of the complete design. For example, the loading on nets can be influenced across many sheets and at many levels in the hierarchy.

The Design Analysis Tools (DAT) is an option available from within the Navigator that identifies these errors that are only present in the context of the complete design. For more information on this module, see the section on *Design Analysis Tools*.

The Hierarchy Navigator also provides several other tools to help you debug the design.

Tracing Signals

Although the problems in a design are usually observed at the very top level — the primary outputs — the problems are usually caused at low levels in the hierarchy. Being able to trace signals from the primary outputs back through the hierarchy can greatly aid circuit debugging.

There are two functions in the Navigator that facilitate this tracing of signals. They are **Misc->Mark** and **Misc->Query**.

Misc->Mark
This command highlights the selected nets or symbols with the appropriate highlighted color. This is used to trace signals from one side of a sheet to the other, across sheets, or through the hierarchy.

Mark is used in some of the simulation interfaces to choose waveforms for display. All items that are marked can be displayed in a list box by typing a question mark (?). See the *Command Reference* section for further details.

Misc->Query
The **Misc->Query** command provides a brief summary of important properties local to the selected element as well as the connections to and from the element. The connectivity information is helpful in tracing the circuit and its connectivity. The local properties of the elements can help you to spot a simulation parameter or loading characteristic that is incorrectly specified.

You can click the mouse on the individual connections listed in the query pop-up window and the system automatically jumps to the area of the schematic containing the selected connections. This allows you to quickly find all connections to a given net or find the Block or Cell that is driving a particular net.

The **Misc->Query** command provides a search facility that displays and highlights elements which you type in at the prompt. Elements which can be searched are: instance names, pin names, and reference designators.

Figure 4-32 shows the tool palette location of the **Misc->Query** command.

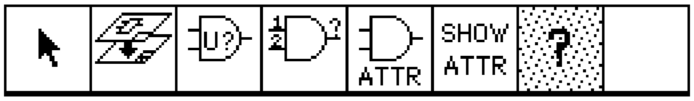


Figure 4-32 ***Misc->Query*** *Command in Tool Palette*

The elements that can be queried are pins, nets, individual symbols, and all symbols of a specified type. Individual symbols can be located based on instance name or reference designator. The information is displayed in a text window. The information available for each of the elements is:

Pins

- pin name
- attached to which instance
- attached to which net
- pin polarity
- fanout

Nets

- net name
- polarity if input or output node
- local net name, applicable only on underlying schematics
- node number in database
- symbol connections

Individual Symbols

- the name (for example, NAND2, NOR3) and type (gate, component, block, cell, master, pin) of symbol
- full path showing location of symbol file in the file structure
- instance name
- reference designator
- other symbol attributes
- reference location on the schematic (sheet number, vertical border reference, horizontal border reference, for example, 2A6)
- gate section (A, B, C, etc) on gate symbols
- instance number in the hierarchical database
- pin/net connections

All Symbols of a Given Type

- internal net count
- instance names and locations

Circuit Checking

The Design Analysis Tools module provides two checking utilities for finding errors in a design. These utilities are accessed through the **Tools** menu of the Hierarchy Navigator. They are described in the *Design Analysis Tools* section.

Tools->ERC Check performs basic connectivity checks and analyzes the current loading or fanout on a completed design. **Tools->PCB Check** performs a packaging check on PCB designs to ensure that the assignment of gates to physical components is correct. These utilities are tightly coupled to the Navigator. For example, the checker generates a list box containing the errors. You can click the mouse on any of the errors listed and the portion of the circuit responsible for the error is highlighted on the schematic.

Circuit Analysis

Before manufacturing a design, it is wise to analyze the circuit to ensure it meets the target specifications. The Navigator directly supports several analysis tools. Many more can be coupled to the Navigator through the **Misc->Processes** or **Misc->Tools** menus. ECS can interface with other analysis tools through netlists.

Simulation

The Navigator supports interfaces for the following simulators:

- SILOS
- Verilog
- X-Sim
- Timemill

This simulator is invoked from the **Simulator** menu. Refer to the *Interfaces* section for more information on the interface to a specific simulator.

Additional simulators are added to the **Processes** menu. Selecting an entry under this menu spawns a task associated with the menu entry. The relationship between the menu sub-entry and the process to be spawned is established using the ECS Preferences Editor. You can add any process or program to the **Processes** menu by editing the **Controls->Processes** section of the ECS Preferences Editor.

You may have to perform some simple netlist translation before running the new simulator. The process called from the menu can be a script that first performs some translation, then calls the simulator. The simulator output can be examined with the same methods used when you run the simulator in stand-alone mode.

A tight integration between an unsupported simulator and the Navigator can be written using the *Processor Interface Kit*. This is a separate module that is available to customize interfaces to the ECS environment.

The Waveform Tool

This is a tool for viewing digital waveforms. It is tightly coupled with the Navigator and allows cross-probing. This is where you select a node in the schematic and the waveform is automatically displayed. Alternatively, a waveform can be selected and the corresponding net on the schematic is highlighted.

The Waveform Tool (WT) is a separate module described in its own section.

ProbeItem

ProbeItem is used to identify the name of a symbol instance or net to the currently running tool. To probe an item, position the cursor over it and click the mouse. The name of the item is sent to the tool. **ProbeItem** does not cause the **Query** box to be displayed, but it uses the **Query** box if it is visible when the command is invoked. This allows you to **Query** several nodes and then switch to the **ProbeItem** command to send the names to the tool.

ProbeItem is often used with the Waveform Tool. You can select nodes with **Tools->ProbeItem** and have the corresponding waveforms added to the waveform display.

FindItem

FindItem can be invoked at any time to find the output pin driving the net that is currently **Queried**. This is useful for tracing signals during the analysis of simulation results.

Some of the tools highlight nets or symbols on the schematic. The **Tools->FindItem** command can be used to search through the hierarchy to find a schematic that contains the output pin that is driving the highlighted net.

Netlisting

The normal method of transferring a design from one step in the design cycle to the next is with netlists. For example, if you have an in-house timing verification system, you can output a netlist of the design from the Navigator and then read this netlist into the timing verification application.

You can write a custom display interface using the *Processor Interface Kit (PIK)* to display the results of your analysis directly on the schematic in the Navigator. Custom netlist interfaces are also possible using the *PIK*.

The Navigator supports several standard or generic netlists. These are:

- SPICE
- HSPICE
- PSpice
- EDIF

- Generic netlist by pin
- Generic netlist by net
- PADS PCB
- Racal Redac netlist format
- Cadnetix
- Timemill
- X-Sim
- Verilog
- VHDL
- Futurenet pin netlist
- Hierarchical SPICE netlist

The netlisters are available under the **Processes** menu in the Navigator and are described in more detail in the *Interfaces* section.

Preparing for Manufacture

The purpose of designing a PCB or IC is to make a circuit that can be manufactured. There are a number of conventions and practices that are used to interface the design to the manufacturing environment. These include such things as reference designators on PCB designs, and netlists for both ICs and PCBs. These manufacturing aids are described in this section.

Pin Numbers

Pin numbers are used in PCB applications to indicate how the pins on a Gate or Component symbol correspond to each physical pin on a physical device. You only need pin numbers if you are doing PCB design.

The **Misc->Pin Numbers** command of the Hierarchy Navigator assigns pin numbers prior to packaging or layout. If the design contains hierarchy, the Hierarchy Navigator must be used to add the pin numbers; otherwise the Schematic Editor can be used. The command is selected from the menu or from the tool palette, as shown in Figure 4-33.

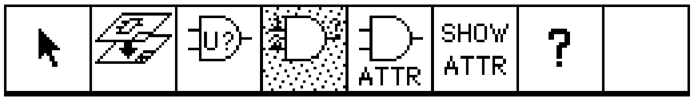


Figure 4-33 ***Misc->Pin Numbers*** *Command in the Tool Palette*

Gate symbols represent one section of a physical device that has more than one section. Component symbols represent a complete physical device where the pins are fixed and cannot be interchanged with different pins of the same functionality in the component. Gate symbols have pin numbers assigned that are actually a group of numbers for each pin function. For instance, the symbol for the 7400 series NAND Gate has pins with the following names and numbers:

Name= IN1	Number= 1 4 9 12
Name= IN2	Number= 2 5 10 13
Name= OUT	Number= 3 6 8 11

This is a Gate symbol (:sym_libs:ttl:7400.sym) that represents one-quarter of a complete 7400 component. The symbol has four sections called *A*, *B*, *C*, and *D*. This is shown in Figure 4-34.

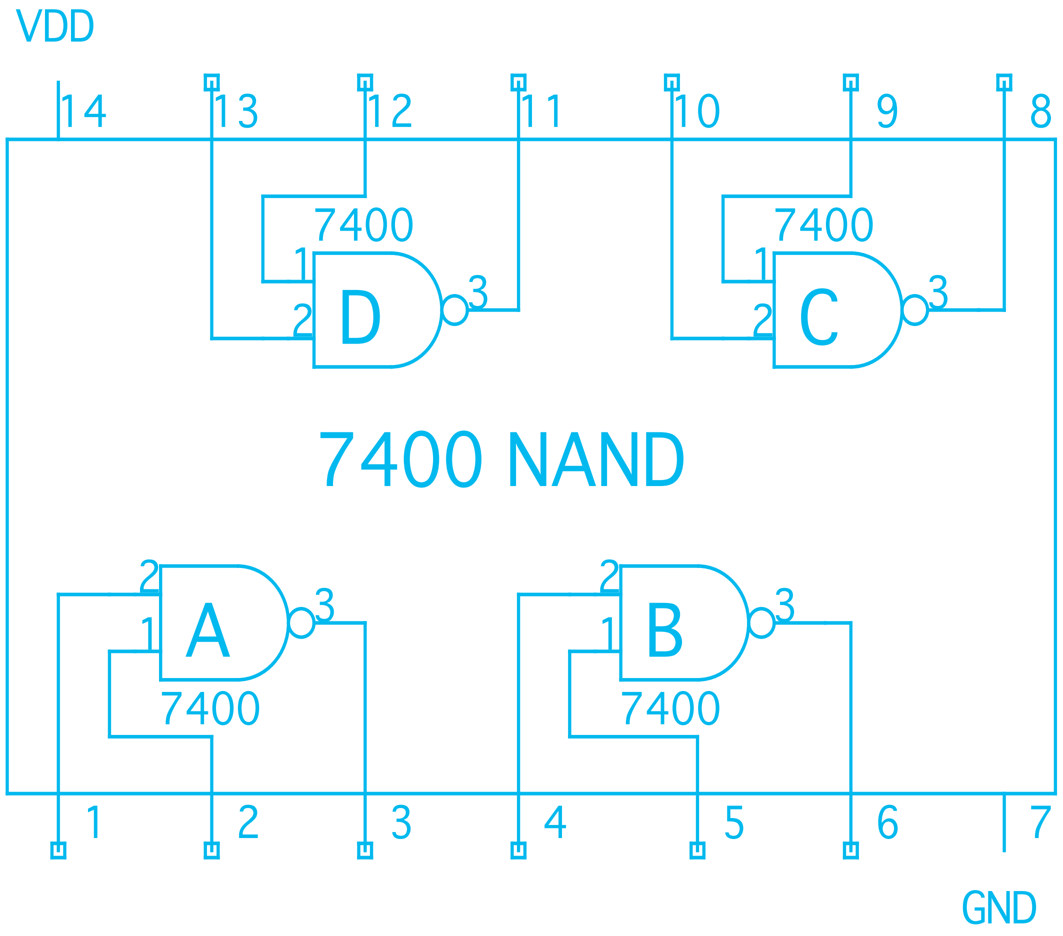


Figure 4-34 Gates in a 7400 Symbol

The numbers on the pins of Gate symbols are initially set to the first section in the physical device. **Misc->Pin Numbers** changes the pin numbers to another section and can reflect a permutation of functionally equivalent pins.

To change the section of a gate, enter the pin number for any unique pin in the section and click the mouse within the outline of the symbol to assign the set of pins for that section. Alternatively, enter a letter corresponding to the required section (*A*, *B*, or *C*) and click the mouse within the outline of the symbol.

To swap two functionally equivalent pins, enter the desired pin number for one of the pins and click the mouse on that pin. The system checks the *Pin Group* attribute. If both pins have the same *Pin Group* (and if the symbol is a Gate, belonging to the same section), the selected pin is swapped with the pin that already has the given pin number.

Some Gates have common pins. For example, the 74175 quadruple D flip flop has a common clock and clear. The pin number attribute in this case has four identical entries as below.

Name= ENAB	Number= 4 4 4 4

These common pins cannot be swapped.

The **Misc->Pin Numbers** command of the Hierarchy Navigator allows you to click on an instance to pick up the current gate letter or to click on a pin to pick up the pin number. This makes it easier to copy the gate or pin from one instance to another.

Reference Designators

Reference designators designate the physical devices represented by the symbols used in PCB applications. You only need reference designators if you are doing PCB design.

The **Misc->Ref Designator** command of the Hierarchy Navigator and Schematic Editor assigns reference designators prior to packaging or layout. If the design contains hierarchy, the Hierarchy Navigator must be used to assign the reference designators; otherwise the Schematic Editor can be used. The command is accessed through the menu or from the tool palette, as shown in Figure 4-35.

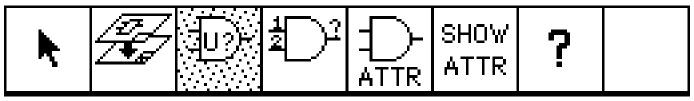


Figure 4-35 ***Misc->Ref Designator*** *Command in the Tool Palette*

The reference designator must be unique for each Component symbol in the schematic. The same reference designator can be assigned to two different Gate symbols if they are the same type of symbol. This permits several symbol instances to represent different sections in the same physical part. The special connector Pin symbols (symbols of type *Pin*) are allowed to have duplicate reference designators. This permits edge connectors to have a single reference designator.

DeMorgan equivalent symbols, such as the 7400 and 7400D symbols, are considered to be the same type. That is, they are permitted to have identical reference designator assignments.

Modify Attributes

Attributes are often modified when the schematic drawings are completed. The Navigator facilitates changing any attributes in the design because it has access to the whole design database.

The results of any analyses performed using simulators or other tools in the Navigator can be used to adjust attribute values in the design. For example, the results of a simulation might show that two instances of an inverter need to be 50% faster for the design to work properly. In this case, you could modify the width and length attributes on the two inverter instances from the Navigator, re-run the simulation, and then verify the effectiveness of the modifications.

Netlisting

A netlist is the normal method of transferring a design to manufacturing. This is the case in transferring an integrated circuit schematic database to a layout system, or if you are transferring a PCB design to a board layout application. The Navigator supports several standard or generic netlists. More are being written and the *Interfaces* section should be consulted for the most up-to-date information. A partial list of netlisters follows:

- SPICE
- HSPICE
- PSpice
- EDIF
- Generic netlist by pin
- Generic netlist by net
- PADS PCB
- Racal Redac netlist format
- Cadnetix
- Timemill
- Verilog
- VHDL
- Futurenet pin netlist
- Hierarchical SPICE netlist

The netlisters are available under the **Processes** menu in the Navigator. Refer to the *Interfaces* section for more information.

You can write your own custom netlist interface using the *Processor Interface Kit*. This is an option that allows access to the database.

Attributes

An attribute in the ECS environment is a characteristic or property belonging to or associated with a symbol, pin, or net. ECS attributes can describe:

- width or length of transistors
- price of a particular chip
- size of a chip or a cell
- number of connections to a block
- delay from input to output
- length of time taken to design a block or cell

In short, attributes can describe anything in an IC or PCB design.

In the ECS environment, an attribute has several parts. An attribute name is perhaps the most obvious and fundamental. *Width*, *Length*, *ReferenceDesignator*, and *PinNumber* are examples of names for attributes in ECS. Another important component of an attribute in the ECS environment is the attribute number. This is something ECS uses to keep track of attributes and you can take advantage of attribute numbers to do some sophisticated things. For example, you can make a derived attribute (more about derived attributes later in this section) that depends on other attributes. The link between the original attribute and the derived attribute is made using the attribute number.

An attribute window is needed in order to display an attribute on a symbol or schematic. It is a port through which you can view attributes. When you define a symbol, you can add these attribute windows to the symbol. Later, when the symbols are placed in a schematic, various attributes can be displayed on the schematic through the attribute windows. Attribute windows, like attributes themselves, have a number associated with them. This assignment of numbers takes place using the ECS Preferences Editor.

This section on attributes explains:

- attribute types
- creating attributes
- assigning values
- using attribute windows

Attribute Types

There are four attribute types in ECS. They are:

global	Global attributes are used for global constants such as technology feature size, supply voltage, or identification codes. These are accessible from every sheet of every schematic at every level of hierarchy in a design.

symbol	These describe various features related to the symbol as a whole. Examples of such features are width and length parameters of transistors, or SPICE model characteristics. Symbol attributes apply only to the symbol on which they appear. They can be passed up or down in the design hierarchy using derived attributes as described later in the section.
pin	These attributes describe features related to individual pins of a symbol. Such things as drive strength or loading effects are typically described using pin attributes. Parameters from pin attributes are accessible at the instance level and can be used to generate net listed information.
net	Net attributes are used to describe characteristics associated with nets. One good example is the capacitance resulting from the routing of a net across a chip.

Creating Attributes

Attributes are created using the ECS Preferences Editor. There is one menu entry for each of the four attributes types:

- global attributes
- symbol attributes
- pin attributes
- net attributes

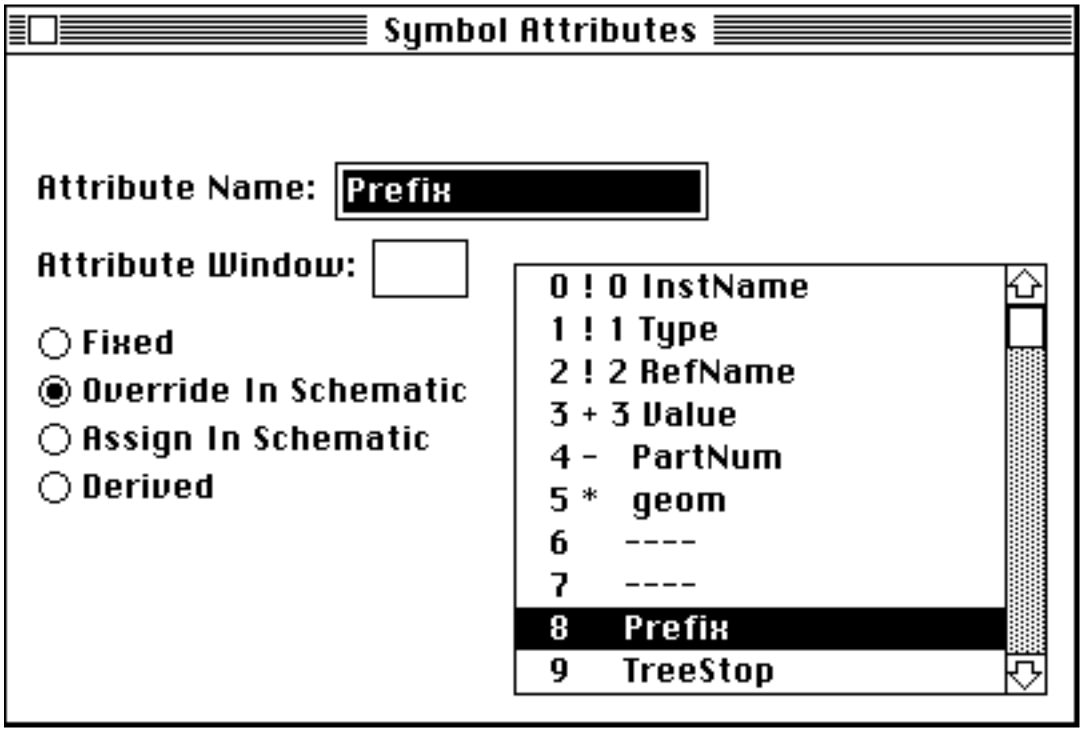
To create an attribute from within the ECS Preferences Editor:

1. Select the **Attribute** menu.
2. Select one of:
   - global attributes
   - symbol attributes
   - pin attributes
   - net attributes

   This opens a window like that shown in Figure 4-36 from which you define the attributes. A list box is displayed on the right containing all the attribute numbers from 0 to 199 listed in order. The next field contains an optional attribute modifier. The third field in the list box is an optional attribute window number. The last field contains the attribute name. Some of the attributes are not assigned and do not have names. Table 4-1 shows some typical attribute entries. A complete list of the default attributes is given in the *System Configuration* section.

   The global attribute edit window is slightly different. It has a column of attribute numbers from 1 to 20 on the left, with the attribute names and values appearing in the middle of the list box.

3. Click the mouse on the attribute number you wish to edit. This enables the edit fields.
4. The attribute name, attribute window, and override modifier can be specified using the edit fields and radio buttons in the left part of the window.



Attribute Name: Prefix

Attribute Window:

( ) Fixed
(●) Override In Schematic
( ) Assign In Schematic
( ) Derived

0 ! 0 InstName
1 ! 1 Type
2 ! 2 RefName
3 + 3 Value
4 - PartNum
5 * geom
6   ----
7   ----
8   Prefix
9   TreeStop

Figure 4-36 Symbol Attribute List Box in ECS Preferences Editor

Attribute Number	Attribute Modifier	Attribute Window	Attribute Name	Comments
0	!	0	InstName	Instance Name
1	!	1	Type	Symbol Name
2	!	2	RefDes	Reference designator for PCBs
3			Value	General value parameter
35	*	8	Width	MOS transistor width
36	*	9	Length	MOS transistor width
102	*	12	Lambda	smallest processing geometry

Table 4-1 Attribute Fields in ECS Preferences Editor

The names of symbol attributes are text strings and can contain any characters except embedded blanks. There is no case sensitivity in the names. You can use upper-case letters to improve readability.

As an example, the attribute, *Type*, shown in Table 4-1, is attribute number 1 and is displayed in attribute window number 1.

You can edit symbol and pin attribute values on the symbol and override the values on any schematic where the symbol appears. The four override modifier characters are used to determine how the attribute values can be changed in the schematic.

!	Edits the attribute value only with special ECS commands such as **Add->Net Name** and **Add->Instance Name**. These are special attributes that ECS uses internally.
-	Edits the attribute value only in the Symbol Editor. It cannot be overridden in the Schematic Editor or Hierarchy Navigator.
+	Edits the attribute value in the Symbol Editor and can be overridden in the Schematic Editor or the Hierarchy Navigator.
*	Specifies a derived attribute and allows you to edit the attribute value in the Symbol Editor. It can be overridden in the Schematic Editor or the Hierarchy Navigator.

Attribute numbers 0 through 99 are reserved for ECS definition. You can change the attribute windows, names, and most values to reflect your local environment, but you cannot change the attribute numbers and purposes associated with those attributes. Attribute numbers 100 through 199 are available for you to define and use for any purpose.

In the example in Table 4-1, the attribute InstName is predefined in ECS and although you can make use of this attribute and even change the name, you cannot change its attribute number (0) or the purpose (storing instance names) for which it is used. The attribute, Lamda, has attribute number 102 and is a user-defined attribute.

Assigning Values to Simple Attributes

When attributes have been created using the ECS Preferences Editor, you can assign various values to the attributes. Many attribute values are assigned during symbol creation. An attribute like *Prefix* — attribute number 8 — provides the SPICE prefix of an element and is used by the SPICE netlister. The value of this attribute is well known at the time of symbol creation as either M (MOS transistor), Q (bipolar transistor), D, C, L, V, I, or R. This value can be assigned in the Symbol Editor by using the **Add->Symbol Attributes** command. This opens up the Attribute Editor window.

If you want to set the *Prefix* value for an MOS transistor symbol:

1. Select **Add->Symbol Attributes**. This displays the Attribute Editor.
2. Select *Prefix* from the list box.
3. Type the value *M* followed by the *Return* key.

Any time this MOS transistor is instantiated, the attribute *Prefix* will have M as its default. Because there would be no reason to ever change the value of this parameter in a schematic, you can add the modifier character - to the *Prefix* attribute using the

ECS Preferences Editor. The - modifier prevents accidental changes to the attribute value in the Schematic Editor or the Navigator.

You can also provide default values of width 5 and length 2 for the MOS transistor. Scroll down through the Attribute Editor, and edit values of the width and length attribute.

When you create a schematic, you can use the transistor symbol and make changes to the width value from within the Schematic Editor. For example, to change the width from 5 to 10:

1. Select **Add->Attributes**. The Attribute Editor dialog box opens.
2. Change the value of 5 under the width attribute to 10.

Because of the normal override or precedence in ECS, the value you just changed in the schematic overrides the default value from the symbol. The ability to edit attribute values is also available in the Hierarchy Navigator. When a design is finished and logically correct, you can go back using the Hierarchy Navigator and adjust device sizes to achieve the necessary timing constraints.

To change the value of an attribute in either the Schematic Editor or the Navigator, a default value must be initially chosen and entered at the symbol level. You can also allow an override of an attribute value without assigning a meaningful default value to the symbol or pin by assigning an asterisk (*) as the default value for the attribute. The asterisk is displayed on instances of the symbol in the schematic. This alerts you to add the required override in the Schematic Editor or Navigator. Do not confuse this asterisk with the one that indicates a derived attribute.

Most of the netlist processors ignore attributes that have a leading asterisk. This feature can be used to eliminate an attribute from a symbol when it is netlisted by overriding the existing symbol attribute with an asterisk in the Schematic Editor. When the asterisk is encountered during netlisting, the attribute is ignored.

Displaying Attribute Values on a Schematic

To display an attribute on a schematic, assign an attribute window number to the attribute using the ECS Preferences Editor. This number is the number of the window where the desired attribute is displayed.

In the Table IV-1 example, the instance name attribute is displayed in attribute window number 0, and the width attribute is displayed in window number 8.

The attribute windows in a symbol are specified by location, text justification, and font size. The justification determines how the attribute value is displayed on a schematic. The text can be left-justified, center-justified, or right-justified. This is adjusted using the **Misc->Graphics Options** command before placing the attribute window. This text is always centered vertically. The text in an attribute window cannot be rotated or mirrored within the symbol. A short line or bar appears both above and below a portion of the attribute window when it is placed on the symbol. This indicates the justification that is used for that window.

Attribute windows are placed on a symbol using **Add->Show Attributes**.

1. Select the **Add->Show Attributes** command. A list box appears containing all the attributes that currently have window numbers defined.
2. Select the attribute from the list box whose attribute window you wish to place. This attribute name is attached to the cursor in the symbol drawing area.
3. Position the cursor over the spot on the symbol where you wish the attribute to be displayed.
4. Click the mouse to place the symbol attribute window on the symbol. In the Symbol Editor, the window contains the attribute name. When the symbol is instantiated in a schematic drawing, the attribute window contains the attribute value for that particular instance rather than the attribute name.

Ensure the attribute is placed so that the attribute value has enough room to be displayed.

If you do not find the particular attribute you want to display listed in the Attribute Window Editor, then you probably need to add a window number to the attribute definition using the ECS Preferences Editor.

Window numbers allow flexibility in working with large designs. For example, suppose you are editing a large design using the Navigator and want to see the values of all the SILOS timing delays associated with the various elements in your design. You can borrow the window associated with an attribute currently displayed, and temporarily reassign that attribute number to the attribute SilosTimes. This is easily done inside the Navigator with the **Misc->Show Attributes** command.

1. Select **Misc->Show Attributes**. A list of all attributes and their associated attribute windows is displayed.
2. Select an attribute, *xyz*, that is currently displayed on the symbol.
3. Remove the window number from attribute *xyz* by using the *Backspace* key.
4. Add that window number to the SilosTimes attribute by clicking the mouse on SilosTimes and entering the window number.

The Silos delay times you wanted to see are now displayed on the schematic in the windows where the attribute *xyz* was displayed.

To return to the original attribute display, either return the attribute windows to the original numbers or exit from the Navigator. The changes to the windows made in the Navigator are temporary and are forgotten as soon as you leave the Navigator.

It is possible to assign one attribute window to several attributes. In this case, the attribute with the lowest attribute number is displayed.

Graphic Symbol Attribute Windows

A special mechanism exists allowing you to modify fields on graphic symbols blocks. The main use of this feature is in constructing title blocks for schematics. The following attribute windows can be placed on graphic symbols and display their contents when placed in a schematic.

#	*name*	*attribute window description*
200	FileName	This displays the name of the schematic. If the schematic has not been saved previously, it displays *untitled.*
201	Date	This displays the date that the file was last written.
202	Sh#	This displays the current sheet number. Valid sheet numbers range from 1 to 99.
203	Sheets	This displays the number of sheets in this schematic.
204	Time	This displays the time that the file was last written, for example 13:59:21.
205	Design	This displays the name of the navigator file (without *.tre* extension).
206	InstName	This displays the instance name of the current block, for example .RB.D1. This is useful in the Hierarchy Navigator to see what the current block is.

In addition to these special attributes, any attributes having attribute numbers between 100 and 199 can be added to graphic symbols.

Using Graphic Symbol Attribute Windows
After opening a graphic or master symbol with the Symbol Editor, you can add the attribute windows described above. Use the following procedure to add any of the graphic symbol attribute windows described above to a graphic symbol.

1. Select **Add->Show Attributes**. A list of available attribute windows is displayed in a list box.
2. Select one of the attribute windows displayed. The selected window is attached to the cursor.
3. Click the mouse to place the attribute window on the symbol.

Derived Attributes *(DAT Only)*

Derived attributes are part of the Design Analysis Tools module and are described in the DAT section of the manual.

Number Notation in Attributes

The ECS has been designed with a flexible system for handling attributes that represent numerical values (e.g., width, length, fanin, impedance). The notational conventions for SPICE have been adopted for these attributes. Numbers may be integers (e.g., 100, -5), floating-point numbers (e.g., 3.14159), either an integer or a floating-point number followed by an integer exponent (e.g., 1E5, 2.54E-3), or any of the preceding types followed by one of the following unit scale factors.

Unit Scale Factor	*Prefix*	*Multiplying Value*
T	tera	1E12
G	giga	1E9
MEG	mega	1E6
K	kilo	1E3
M	milli	1E-3
U	micro	1E-6
N	nano	1E-9
P	pico	1E-12
F	femto	1E-15

Alphabetic characters immediately following a number are ignored unless they represent a scale factor. Therefore, 10, 10V, 10VOLTS and 10HZ all represent the number 10 just as 1000, 1000.0 1000HZ, 1E3 and 1.0E3 all represent the number 1000.

Each attribute has a default scale factor. Whenever an attribute appears without one of the above scale factors, the default scale factor is used. If an attribute has a unit scale factor specified, the value is taken exactly as written (the default scale factor is ignored).

For example, the width attribute represents micrometers when it appears on a transistor symbol. The default scale factor for width is U (micro), so a width of 1000 on a transistor would represent 1000 micrometers or 1 millimeter. However, a width of 1K on a transistor would mean 1 kilometer because the presence of the unit scale factor, K, causes the default scale factor to be overridden.

Examples:

Attribute	*Value*	*Meaning*
Width	1	1 micron
Width	1U	1 micron
Width	1000	1000 microns
Width	1K	1 kilometer

Differences in IC and PCB Design Entry

This section describes some of the differences between integrated circuit (IC) and printed circuit board (PCB) design as it pertains to the ECS.

CAD tools for ICs need the ability to handle hierarchical designs with very large element counts containing relatively few different building-block cells. PCBs typically don't need as much hierarchy, but they have very large libraries of components and must have support for:

- reference designators
- pin numbers
- the ability to swap pins within a package
- DeMorgan logically equivalent symbols

A short example at the end of this section shows how the PCB features are used.

Symbol Types

ECS uses different symbol types for representing primitive elements in ICs and PCBs. The three different primitive elements are described below.

Component	These symbols represent a complete physical package in PCB design. A typical example is a TTL 7400 quad NAND package. The Component symbol has pin numbers that correspond to the physical pin numbers on the package.
Gate	These symbols represent a portion of a physical package in a PCB design. A typical example is a single NAND gate from a TTL 7400 quad NAND package. By representing only a fraction of a package, you have the flexibility to assign gates to packages after the logic design is complete. You can change the pin assignments easily using Gate symbols in ECS. For example, you might find that swapping two gates in a package would lead to a cleaner PCB routing. You can do this by using the **Add->Pin Numbers** command in the Schematic Editor or the **Misc->Pin Numbers** command in the Hierarchy Navigator.
Cell	These symbols represent the lowest-level primitives used in IC design. Cell symbols do not support pin numbers or reference designators.

ECS provides support for both pin numbers and pin names. In ICs you use only pin names, while in PCB design you always use pin numbers and can optionally use pin names.

DeMorgan Equivalent Gates

DeMorgan equivalent logic gates are logically equivalent but have different schematic representations. Often a schematic is easier to understand when a particular type of logic gate is used. For example, a circuit where many inputs are ORed together to form an error flag signal can be implemented with NAND gates. However, the circuit is much more understandable when drawn with OR gates than NAND gates. Even though the logical result is the same, people are conditioned to associate the ORing function with a symbol that looks like an OR gate.

DeMorgan equivalent gates are used when the resulting schematic conveys more meaning than the physical implementation. An example of DeMorgan equivalent gates is shown in Figure 4-37 where a NAND gate is represented by an OR gate that has two inversion bubbles on its inputs.

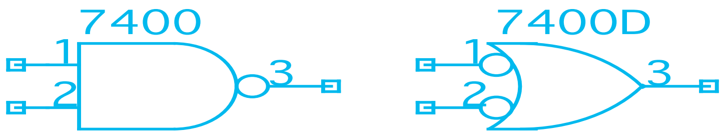


Figure 4-37 DeMorgan Equivalent Gates

Instance Names and Reference Designators

There are several differences between PCB and IC designs. One such difference is the convention used to identify individual components in a design.

The convention used to identify elements in a PCB design is the *reference designator*. A reference designator is a unique code that identifies each physical component in a PCB design. Typical codes are:

- U1, U2, U3… to identify integrated circuit components
- C1, C2, C3… to identify capacitors in the circuit
- R1, R2, R3… to identify resistors in the circuit

This convention works well as long as there are not too many components in the design.

In large hierarchical designs, such as IC designs, it is difficult to provide a unique reference designator to each primitive element. For this reason, and also to make the larger databases more understandable, IC designs use instance names to identify the individual components. Meaningful names can be assigned to low-level components and also the Block symbols that represent hierarchy. Every time a level of hierarchy is encountered, the lowest-level components derive their full hierarchical names by concatenating all of the higher-level block names together with the primitive's name. A delimiting character separates each level in the name, thus allowing you to determine the level of hierarchy by looking at the full hierarchical instance name.

Examples of hierarchical instance names are:

cpu.alu.bit1.carry_lookahead.inverter1
mother_board.disk_controller.lm741

ECS supports both reference designators and instance names. You can select whether to use reference designators or instance names to identify individual elements in a design by setting the *ApplicationMode* parameter in the **Controls->System Controls** section of the ECS Preferences Editor. This toggle parameter can be set to the following values:

IC	Provides instance name support
PC	Provides support for pin numbers and reference designators
BOTH	Provides support for instance names, pin numbers, and reference designators

Assigning Reference Designators

Use the following procedure to assign reference designators:

1. If your design is flat, select the **Add->Reference Designator** command in the Schematic Editor. If your design is hierarchical, select the **Misc->Reference Designator** command from the Hierarchy Navigator.
2. Type the reference designator name. The name is attached to the cursor.
3. Position the cursor over the Component or Gate symbol to be named and click the mouse.

Pin Numbers

PCBs are made from packaged parts that have pins. These pins are soldered to a board to form the electrical connections between the board and the packaged part. One convention in the PCB industry refers to the electrical connections of a packaged IC in terms of the pin numbers.

Pin numbers are first assigned to Gate and Component symbols in the Symbol Editor. Use the following procedure to initialize the pin numbers.

1. Select the **Add->Pin Attributes** command while editing the target symbol.
2. Assign the proper pin numbers to the *PinNumber* attribute for the various pins. Component symbols have only one pin number assigned per pin.

   Gate symbols have one pin number for each gate section in the package. For example, the TTL 74174 hex D flip-flop with clear, has six sections. The Gate symbol representing this device has six *PinNumber* attribute entries for the D input corresponding to the six possible input pins. They are 3 4 6 11 13 14. The *PinNumber* entries for the Q output are 2 5 7 10 12 15. Spaces are used as delimiters to separate the pin numbers.

When Gate symbols are first placed in a schematic, they have the pin assignments from the first section of the package. ECS allows you to change this pin number assignment while editing the schematic or hierarchy. You must do this to properly fill a package with gates.

If there is any hierarchy in the design, use the **Misc->Pin Numbers** command in the Hierarchy Navigator to change pin numbers on your design. The Navigator allows you to traverse the hierarchy and to push into low-level components and reassign the required pin numbers.

You can use the **Add->Pin Numbers** command in the Schematic Editor to change the pin numbers if the design is flat.

There is no equivalent inside an IC to the pin numbers described above.

Gate Assignment

A common objective in PCB routing is to have as few crossovers as possible. You can often minimize the number of wire crossings by careful assignment of gates to packages and careful assignment of gates within a package. ECS allows you to complete the design before doing any gate assignment. After the logic has been verified, you can assign reference designators and distribute the various gates among the different packages.

Swapping functionally equivalent gates can sometimes lead to cleaner PCB routing. Use the following procedure to change the assignment of gate sections within a package. This is only possible with Gate symbols.

1. Select the **Add->Pin Numbers** command in the Schematic Editor for a flat design or the **Misc->Pin Numbers** command in the Hierarchy Navigator in a hierarchical design. The system prompts for a pin number or gate section.
2. Choose the section to which you wish to reassign the target gate. Type either the section letter (D for fourth section) or one pin number from the chosen section.
3. Click the mouse on the gate to be reassigned. The pin numbers on the gate change to reflect those on the new gate section.

Pin Swapping

Swapping functionally equivalent pins can sometimes lead to cleaner PCB routing. Use the following procedure to swap two functionally equivalent pins that belong to the same gate.

1. Ensure that the two pins to be swapped have the same value for the pin attribute, *PinGroup*. This attribute accepts numeric values. If two pins belong to the same gate section and have the same value for *PinGroup*, they can be swapped.
2. Select the **Add->Pin Numbers** command in the Schematic Editor for a flat design or the **Misc->Pin Numbers** command in the Hierarchy Navigator in a hierarchical design. The system prompts for a pin number or gate section.

3. Type the pin number of the first pin to be swapped.
4. Click the mouse on the second pin to be swapped. The two pin numbers are interchanged.

Example PCB Design

The following example builds the latch designed in the *Brief Tutorial* section (shown in Figures 3-17 and 3-19) using a TTL 7400 NAND package. The features described in this example include:

- use of Gate symbols
- DeMorgan equivalent symbols
- pin numbers
- reference designators
- assigning gates within a package
- swapping pins within a gate

Figure 4-38 shows a circuit equivalent to Figure 3-17 that is composed of two-input NAND gates. The OR gate at the output of the latch is a DeMorgan equivalent of a NAND gate. This is indicated by the two bubbles on the inputs of the OR gate. Because the bubbles on the outputs of the NAND gates cancel those on the inputs to the OR gate, the circuit is logically the same as the latch shown in Figure 3-17.

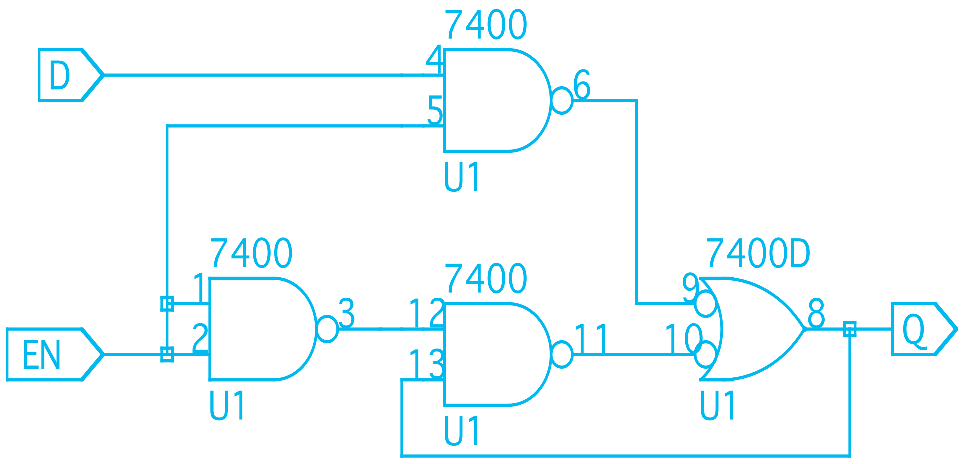


Figure 4-38 Latch Built Using NAND gates

System Configuration

Before creating a PCB design, you must set the *ApplicationMode* parameter in the **Controls->System Controls** section of the ECS Preferences Editor to be either *PCB* or *BOTH*. This enables the reference designator and pin number functions in ECS.

Creating a Gate Symbol

Two symbols are needed for the latch circuit shown in Figure 4-38. One is a standard NAND gate and the other is the DeMorgan equivalent NAND that shows an OR function. The following explanation of how to create these symbols assumes the standard pinout for a 7400 NAND gate as shown in Figure 4-39.

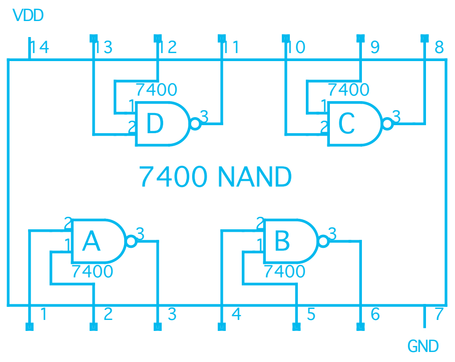


Figure 4-39 Pinout of TTL 7400 Quad NAND Gate

Use the following procedure to create the NAND symbol.

1. Open a new symbol window.
2. Use the **Misc->Change Symbol Type** command to make the symbol of type gate.
3. Draw the NAND symbol with two input pins and one output pin.
4. Use the **Add->Symbol Attributes** command to set the reference designator attribute to *U*. The schematic packager in the DAT uses this default prefix and automatically adds a number to create a unique reference designator. See the *Design Analysis Tools* section for more details.
5. Use the **Add->Show Attributes** command to place attribute windows for:

Type	takes on the value of the symbol file name
RefDes	reference designator

6. Use the **Add->Pin Attributes** command to set the following attributes:

PinName	Choose suitable pin names such as IN1, IN2, OUT.
Polarity	Set the polarity of the input pins to IN and the polarity of the output pin to OUT.
PinNumber	Set one of the input pins to the following list: 1 4 9 12
	This corresponds to the pin numbers of all of the IN1 input pins on the 7400 NAND package.
	Set the other input pin to the following list: 2 5 10 13

This corresponds to the pin numbers of the IN2 input pins on the 7400 package.

Set the output pin to the following list: 3 6 8 11

This corresponds to the pin numbers of the output pins on the 7400 package

By setting the *PinNumber* attribute to a list of values, ECS knows that there are multiple gates in a package. The completed DeMorgan equivalent symbol is shown in Figure 4-40.

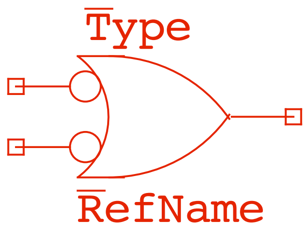


Figure 4-40 Completed DeMorgan Equivalent 7400 NAND Gate

Create the Latch Schematic

Use the following procedure to create the latch schematic as shown in Figure 4-38.

1. Open a new schematic window with the Schematic Editor.
2. Place the four instances as shown in Figure 4-38. All four symbols display pin numbers from the *A* section of the quad NAND gate. That is, they all have pin numbers of 1, 2, and 3.
3. Wire the circuit to look like Figure 4-38.
4. Use the **Add->Reference Designator** command to set *RefDes* to *U1* on all of the symbols. This indicates that all of the gates are from the same package. The system only lets you group symbols of the same name under the same reference designator.

   For example, if you place a three-input NOR symbol on the latch drawing, ECS does not allow you to assign a reference designator of *U1* to the three-input NOR after you have already assigned *U1* to a 7400 NAND symbol.
5. Select the **Add->Pin Number** command. Type *5*, which is a pin in the second gate of the package. Click the mouse on the top NAND gate. The pin numbers of this gate change to reflect those of the second gate in the package — namely, 4, 5, and 6.

   The **Add->Pin Number** command now prompts for a new pin number or gate section. Type *C*, representing the third NAND section in the package. Click the mouse on the DeMorgan equivalent OR gate. Its pin numbers change to reflect the third gate in the package.

Change the pin numbers of the bottom center NAND gate to reflect the fourth gate in the package. The schematic now looks like that shown in Figure 4-41.

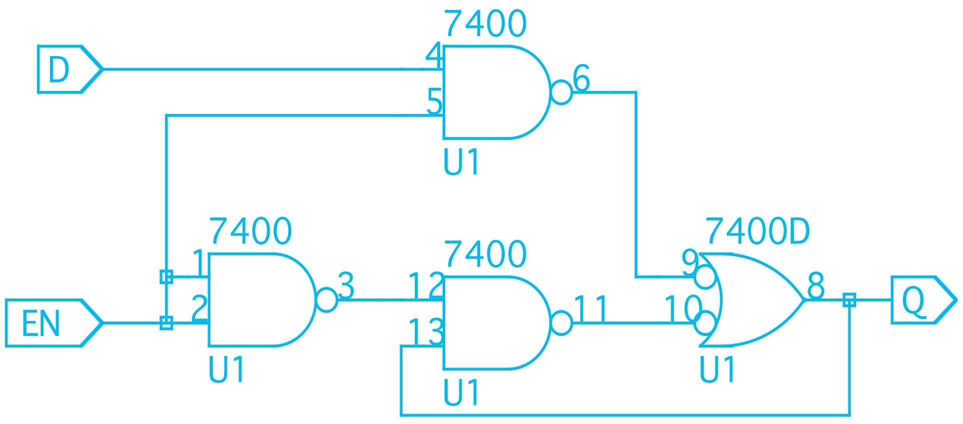


Figure 4-41 Latch With Pins 4 and 5 Crossing in Layout

Figure 4-42 shows the package pinout with connections reflecting the schematic in Figure 4-41. You can see that this pin assignment results in the *D* and *EN* lines crossing to reach pins 4 and 5. This can be avoided by swapping pins four and five. Because the inputs are symmetrical, this does not change the functionality.

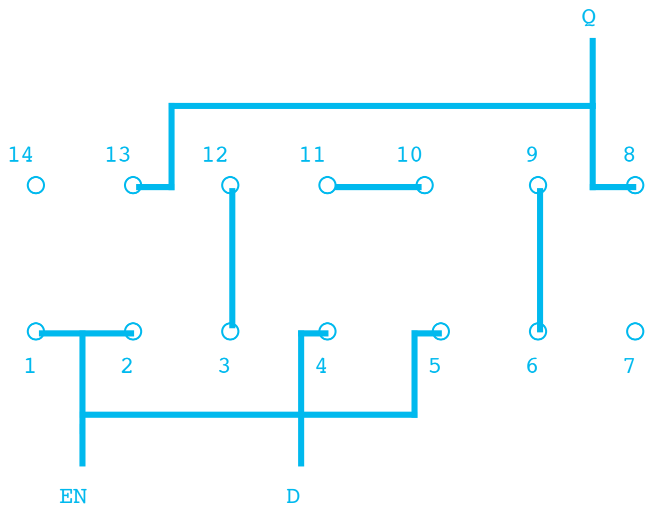


Figure 4-42 Board Routing of NAND Package

Use the following procedure to swap pins four and five.

1. Select the **Add->Pin Numbers** command. A prompt is displayed for a pin number or gate section.
2. Enter 4 or click on Pin 4.

3. Click the mouse on what is currently pin 5 in the schematic. Pins 4 and 5 are swapped.

   This eliminates the crossing of the *D* and *EN* inputs. The ECS checks the *PinGroup* attribute before performing the pin swap. Because pins 4 and 5 belong to the same pin group, they can be swapped.

The final schematic is shown in Figure 4-43.

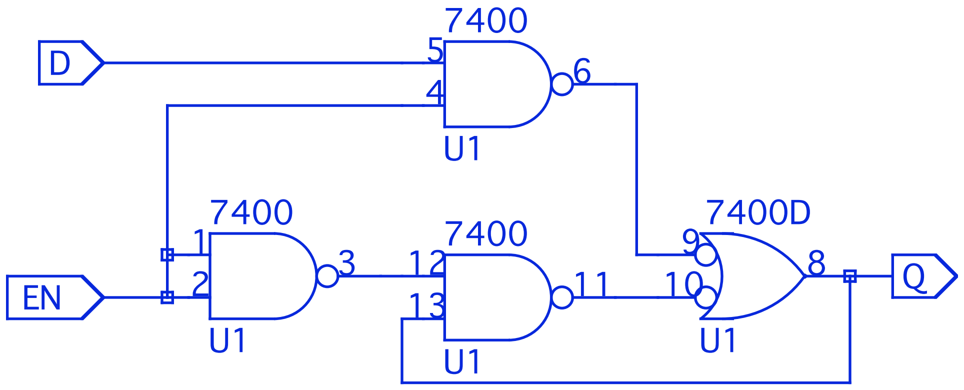


Figure 4-43 Finished Schematic

5

ECS Command Reference

This section explains in detail all of the commands used in the Symbol Editor, Schematic Editor, and Hierarchy Navigator.

Command Reference

This section describes the commands available in the ECS Symbol Editor, Schematic Editor, and Hierarchy Navigator. Commands for the Waveform Tool, the Design Analysis Tools, and the other tools and processes are described in their own sections.

The following conventions are used to describe the commands.

- The commands are listed in alphabetical order.
- ECS commands are shown in boldface type.
- ECS commands are usually shown with the name of the menu where the command is found. For example, the **Line** command is found under the **Draw** menu in both the Symbol Editor and Schematic Editor. It is referred to as **Draw->Line**.
- *Sym*, *Sch*, or *Nav* (or any combination of the three) is printed on the right side of the command line to indicate which ECS applications use the described commands. Occasionally the command appears in a different menu in the different applications. This is noted by including the alternate menu in parentheses beside the application name. For example, **Add->Pin Numbers** is found in the Schematic Editor. It is also found in the **Misc** menu of the Hierarchy Navigator. The right side of the command shows:

  Sch, Nav (Misc)
- The use of list boxes and edit fields is described in the *Before You Begin* section.
- Many commands have a keyboard accelerate key activated by holding down the command key and another key at the same time. The command key is indicated on most computer keyboards with the following symbol (▯).

Quick Command Reference

command name	Keyboard	Application
->About		Sym, Sch, Nav
->Help	⌘?	Sym, Sch, Nav
Add->Attributes		Sch, Nav (Misc)
Add->Bus Tap		Sch
Add->Expand Bus Name		Sch
Add->Instance Name		Sch
Add->I/O Marker	⌘L	Sch
Add->Net Name	⌘F	Sch
Add->Number Pins		Sym
Add->Pin		Sym
Add->Pin Attributes	⌘F	Sym
Add->Pin Numbers		Sch, Nav (Misc)
Add->Ref Designator		Sch, Nav (Misc)
Add->Simulator Net Attributes		Nav
Add->Show Attributes		Sym, Sch
Add->Show Pin Names	⌘L	Sym
Add->Symbol		Sch
Add->Symbol Attributes		Sym, Sch
Add->Table		Sch
Add->Table Data		Sch
Add->Wire		Sch
Draw->Arc		Sym, Sch
Draw->Big Bubble		Sym
Draw->Bubble		Sym
Draw->Circle		Sym, Sch
Draw->Line		Sym, Sch
Draw->Rectangle		Sym, Sch
Draw->Text		Sym, Sch
Edit->Clear	*Delete* key	Sym, Sch, Nav
Edit->Copy	⌘C	Sym, Sch, Nav
Edit->Cut	⌘X	Sym, Sch, Nav
Edit->Drag	⌘T	Sym, Sch
Edit->Duplicate	⌘D	Sym, Sch
Edit->Mirror	⌘M	Sym, Sch
Edit->Move		Sym, Sch
Edit->Paste	⌘V	Sym, Sch, Nav
Edit->Redo		Sym, Sch, Nav
Edit->Rotate	⌘R	Sym, Sch
Edit->Select	⌘E	Sym, Sch
Edit->Undo	⌘Z	Sym, Sch, Nav

File->Back Annotate		Nav
File->Create Symbol		Sch
File->Dump Image		Sym, Sch, Nav
File->Edit Schematic		Sym, Nav
File->Edit Symbol		Sch, Nav
File->Expand Symbol		Sym
File->New	⌘N	Sym, Sch
File->Open	⌘O	Sym, Sch
File->Page Setup		Sym, Sch, Nav
File->Print	⌘P	Sym, Sch, Nav
File->Print Window		Sch, Nav
File->Quit	⌘Q	Sym, Sch, Nav
File->Restart		Nav
File->Revert		Sym, Sch
File->Save	⌘S	Sym, Sch, Nav
File->Save As		Sym, Sch
File->Sheets	⌘Y	Sch, Nav
Misc->Change Symbol Type		Sym
Misc->Display Options		Sch, Nav
Misc->Edit Attributes		Nav, Sch (Add)
Misc->Edit Constants		Nav
Misc->Error Check		Sym, Sch
Misc->Graphic Options	⌘G	Sym, Sch
Misc->HiLite		Sch
Misc->Mark		Nav
Misc->Pin Numbers		Nav, Sch (Add)
Misc->Query	⌘I	Sch, Nav
Misc->Ref Designator		Nav, Sch (Add)
Misc->Set Origin		Sym
Misc->Show Attributes		Sch, Nav
Misc->Statistics		Sch, Nav
Processes		Nav
Push/Pop		Nav
Tools->Find Item	⌘F	Nav
Tools->Probe Item		Nav
View->Full Fit		Sym, Sch, Nav
View->Redraw		Sym, Sch, Nav
View->Reset Views		Nav
View->Window		Sym, Sch, Nav
View->ZoomIn	⌘K	Sym, Sch, Nav
View->ZoomOut	⌘J	Sym, Sch, Nav

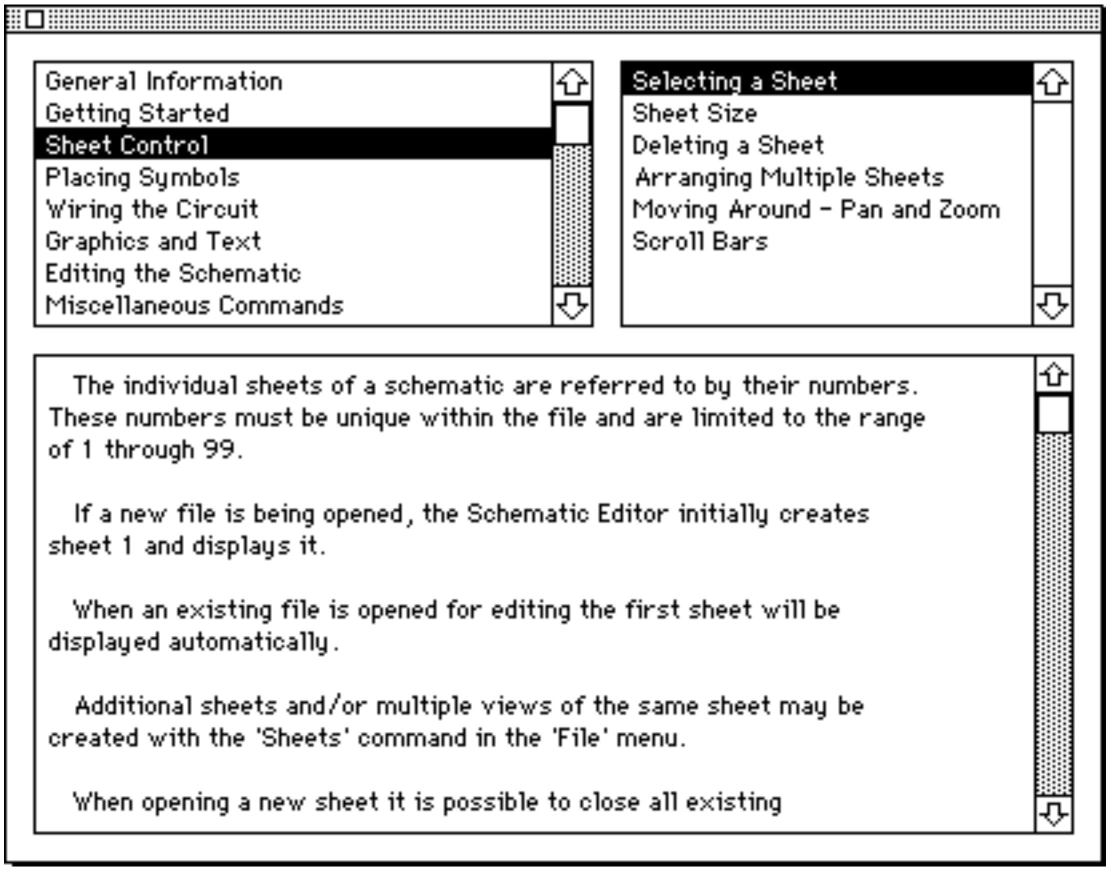
->About ...

Sym, Sch, Nav

The About box under the apple icon lists the software version number. It also lists the serial number of the copy being used.

On-line help is available at any time with the **Help** command. The **Help** facility presents a dialog window with a list of *topics* on the left and a list of *subtopics* on the right. To access the help information:

1. Select the **->Help** command. A Dialog box is displayed similar to that shown in Figure 5-1
2. Select the *topic* using the mouse. The list of *subtopics* is updated.
3. Select a *subtopic* from the list using the mouse. The on-line help information for that particular *subtopic* is displayed in the large bottom window.



The individual sheets of a schematic are referred to by their numbers. These numbers must be unique within the file and are limited to the range of 1 through 99.

If a new file is being opened, the Schematic Editor initially creates sheet 1 and displays it.

When an existing file is opened for editing the first sheet will be displayed automatically.

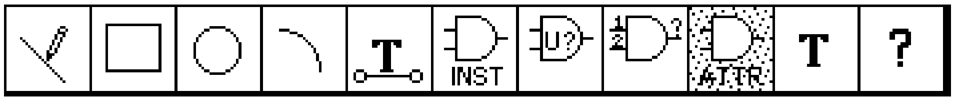
Additional sheets and/or multiple views of the same sheet may be created with the 'Sheets' command in the 'File' menu.

When opening a new sheet it is possible to close all existing

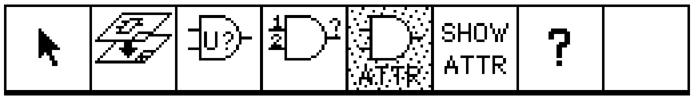
Figure 5-1 Help Dialog Box

Add->Attributes

Sch, Nav (Misc)



Schematic Editor Tool Palette



Navigator Tool Palette

This command is referenced as **Misc->Edit Attributes** in the Navigator. This command assigns values to symbol and pin attributes from the Schematic Editor and Navigator. Net attributes can be assigned only in the Navigator. The command displays a dialog box with an edit field on the top and a list box on the bottom. The list box initially contains the statement:

No Symbol or Pin Selected

Operation on Symbol, Pin or Net

After a symbol, pin, net (Navigator only) is selected, the list box contains a list of the attributes that can be modified. The edit field on top is where the values for each of the attributes are changed.

To operate:

1. Click the mouse on the symbol or pin (or net in the case of the Navigator) whose attribute you wish to change.
2. Click the mouse on the name of the attribute you wish to modify. The attribute name and its current value appear in the edit box.
3. Edit the attribute value.

Operation with Buses

After a bus is selected in the Navigator, a list box is displayed allowing you to select the one element from the bus. Once you have selected a single net form the list of bus elements, a second list box is displayed. The list box contains a list of the net attributes that can be modified. The edit field on top is where the values for each of the attributes are changed.

To operate:

1. Click the mouse on the bus in the Navigator whose attributes you wish to change. A list box displays the elements contained in the selected bus.

2. Click the mouse in the list box on bus element whose attribute you wish to change.
3. Click the mouse on the name of the attribute you wish to modify. The attribute name and its current value appear in the edit box.
4. Edit the attribute value.

These procedures can also be used to delete an attribute value. In this case, the overriding attribute value is removed and the default attribute value from the symbol becomes active. Deleting an attribute value from the Schematic Editor or Navigator removes the override only; it does not remove the default value as originally defined in the symbol.

There are five classes of attributes:

System	The system reserves some attributes for things like net names and instance names. These can only be modified through special system commands such as **Add->Instance Name**. These attributes do not appear in the list box of the **Add->Attribute** command.
Fixed	Values for these attributes are set in the symbol and cannot be overridden in the Schematic Editor or Navigator. A simulation model name is an example of something that is defined once and is not modified at the instance level.
Predefined	These attributes are predefined in the symbol and can be overridden in the Schematic Editor or the Navigator. Examples of where this type of attribute is used are transistor width and length. This type of attribute must have a value defined on the symbol before it can be overridden in the schematic.
Open	These attributes are always accessible at all levels in the design. They are used for things like comments.
Derived	These attributes consist of a constant value or a formula that can depend on other attributes' values. Derived attributes can pass information up and down through the hierarchy. They can be overridden in the Schematic Editor or Navigator.

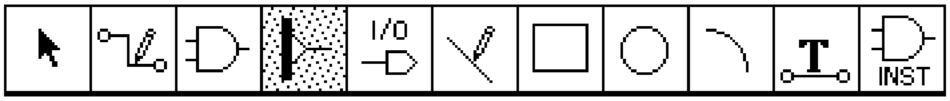
Attributes are defined in the ECS Preferences Editor. Access to the definitions is through the **Attributes** menu of the ECS Preferences Editor.

See Also

- The *Attribute* section in this manual for a detailed description of attributes
- The *System Configuration* section, which details how to modify attribute definitions in the ECS Preferences Editor
- **Add->Symbol Attributes** for a description of adding symbol attributes in the Symbol Editor
- **Add->Pin Attributes** for description of adding pin attributes in the Symbol Editor

- **Add->Show Attributes** allows adding attribute windows in the Symbol Editor
- **Misc->Show Attributes** allows attribute windows to be modified in the Navigator

Add->Bus Tap



Schematic Editor Tool Palette

The **Bus Tap** command creates a bus tap that connects to an existing signal or a signal that can be named at a later time. A bus tap is the point at which a signal enters or leaves a bus. The tap is shown as a small triangle on the side of the bus with a signal wire leaving the triangle in a direction perpendicular to the bus. Each signal tapped from a bus must have a net name flag to identify it.

Bus taps can be made on vertical or horizontal sections of a bus. A tap cannot occur on a diagonal section. To create a tap:

1. Choose the **Bus Tap** command. It is available as an icon from the tool palette or from the main menu.
2. Click the mouse on the bus or wire at the location where you want the tap to originate. If a tap is made from a wire, that wire is promoted to a bus.
3. Click the mouse at the end point of the bus tap. The tap is drawn from the bus to this point.

Once a tap has been added, you can use the **Add->Wire** command to extend the connection to a symbol pin, net name flag, or another net.

The **Add->Net Name** command can be used to create bus taps because it permits the tap, a wire, and the net name flag to be placed simultaneously.

See Also

- **Add->Net Name**
- **Add->Wire**

Add->Expand Bus Name

The **Expand Bus Name** command allows you to break a bus name which is attached to the cursor into its constituent signal names. Each successive element of the bus is separately attached to the cursor so that you can place each element of the bus in succession.

This command is only accessible when the **Add->Net Name** command is active and there is a bus name attached to the cursor. At other times, its menu entry is greyed. After these conditions are met (see **Add->Net Name**) use the following procedure to expand a bus name into its constituent signals.

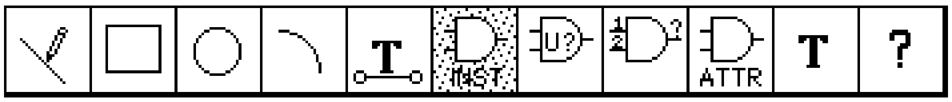
1. Use the **Add->Net Name** command to attach a compound name (bus name) to the cursor.
2. Select the **Add->Expand Bus Name** command from the menu or by typing the escape key. The first element of the bus is now attached to the cursor, ready for placement. See the **Add->Net Name** command for details on net name placement.
3. After placing the first element of the bus, the second element is attached to the cursor ready for placement. As each element is placed in turn, each successive bus element is attached to the cursor. This process continues until all the individual elements of the compound name have been parsed from the name and placed on the schematic.

A special macro command is built into the system permitting rapid connection of name flags to pins. This macro is activated when a name flag is connected to a symbol pin. Subsequent name flags can be placed by clicking the mouse with the cursor over another pin. A segment of the same length as that created by the previous command is created with the name flag attached as above. This is particularly useful when adding many bus taps from a bus to an adjacent symbol.

See Also

- **Add->Net Name**

Add->Instance Name



Schematic Editor Tool Palette

Add->Instance Name allows you to assign meaningful instance names. Each symbol instance must have a unique instance name by which it is referenced. The Schematic Editor automatically assigns names to any symbols that are unnamed when the schematic is saved. The automatically assigned names are of the form *I_nn*, where *nn* is an integer.

Add->Instance Name allows only alphanumeric characters (unless you are naming buses or iterated instances). Names can start with numbers or can be totally composed of numbers. Instance names are not case sensitive and are converted to upper case.

Several options are available.

Full Instance Name
The command prompts you to enter an instance name. You can enter any alphanumeric sequence of fewer than 80 characters. When you press the *Return* key, the name you entered is attached to the cursor. You can attach the name to an instance by clicking the mouse on top of the instance. You are then prompted to enter a new instance name.

This command is terminated by choosing another command.

Click and Increment
This is used when you want a sequence of names, such as INV4, INV5, INV6…

1. Type the prefix, the first sequence number, and a plus sign — all concatenated together. For example, *INV4+*.
2. Click the mouse on the first instance in the sequence. This attaches the instance name to the instance. The instance name consists of the prefix name and the first number in the sequence. For example, *INV4*. If the name is already assigned, the number increments to the next unused number.
3. Click the mouse on each additional instance in the sequence. The next name in the sequence is attached to the cursor after each placement.

To terminate the increment mode sequence, press the *Escape* key. This allows you to enter a new instance name.

Iterated Instances

Add->Instance Name can be used to generate an array of identical instances. To create an array of eight inverters as might be required for an input buffer, make the instance name as follows.

1. Choose the base or prefix name, such as INV.
2. Add a left bracket, [.
3. Add the starting number for the iteration, such as 8.
4. Add a colon ‘:’ as a delimiter.
5. Add the stopping number for iteration, such as 15.
6. Add the matching right bracket to close the expression,].

   The example of INV[8:15] now represents a bank of eight inverters with the names INV[8], INV[9], INV[10]…INV[15]. This is a compact notation for representing bit-slice structures.

   The system places eight instances of inverters even though you only see one displayed. You can push into the individual instantiated inverters using the Hierarchy Navigator and selecting the inverter name to push into. For example, if you are viewing the iterated instance INV[8:15], you can push into INV[9] and view the single inverter instance. This also allows you to change attributes on individual members of an iterated instance.

In all of these methods, the system checks before attaching an instance name to a symbol to make sure the name is not already in use. If it is used elsewhere, an error message alerts you, and you can choose another name.

Only block, cell and component symbol instances can be iterated. The count of the instances is specified by assigning an instance name of the form *XX[i:j]*.

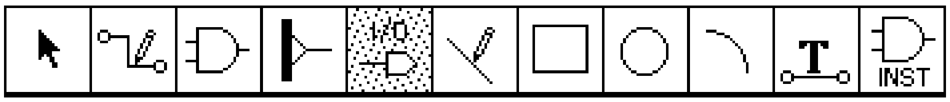
Connections to the nets of iterated instances are made according to the following rules:

- a scalar net net connected to a scalar pin connects to the corresponding pin on each iterated instance.
- a bus connected a scalar pin connects its n’th net to the scalar pin of the n’th instance. The width of the bus must be the same as the number of instances.
- A bus connected to a bus pin connects each successive pin of the bus to each successive pin of the bus pin. The bus and bus pin must have the same net count.

Assigning reference designators to iterated component symbols is accomplished by providing the list of reference designators as the reference designator attribute on the iterated symbol. The list is a comma or blank delimited list of reference designators. A sequence of reference designators can be included by enclosing the numeric range in ()’s or []’s.

For example, instance name *CC[1:20]* specifies 20 iterations of a symbol. Reference designator *C1,C3,C(5:20),C22,C24* assigned to the symbol causes the 20 iterations to be assigned the reference designators: *C1, C3, C5, C6, C7, …,C20, C22, C24*.

Add->I/O Marker



Schematic Editor Tool Palette

An I/O marker is a special indicator that can be attached to a net name flag to identify it as an input, output, or bidirectional signal for the schematic. All primary inputs and outputs must be marked with I/O markers. In addition, all underlying schematics in the hierarchy must have I/O markers on nets that correspond to the symbol pins representing the underlying schematic.

The Schematic Error Checker uses I/O markers to ensure that the polarity of marked signals agrees with the polarity of any attached symbol pins. Any discrepancies are flagged by the Schematic Error Checker. Discrepancies in polarity are also flagged each time you start the Navigator.

To mark a net as input, output, or bidirectional:

1. Select the appropriate polarity from the pop-up dialog box (see Figure 5-2).
2. Place the cursor on the net name flag at the end of a horizontal or vertical wire segment or bus. Click the mouse.

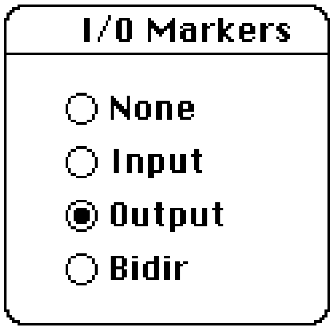


Figure 5-2 I/O Marker Dialog Box

Several markers can be placed at one time by dragging a box around the wire ends that are to receive markers.

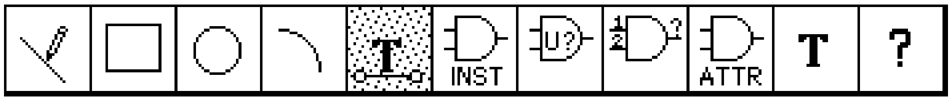
An I/O marker can only be added to net name flags that occur on the end of a horizontal or vertical segment of wire.

A net can only have one I/O marker in the schematic.

See Also

- **Add->Net Name**

Add->Net Name



Schematic Editor Tool Palette

This command allows you to manually name nets.

Nets not named in this fashion receive automatically generated names of the form *N_nn*, where *nn* is an integer. This automatic naming of the nets occurs when you exit or save from the Schematic Editor. These automatically created names are preserved in the schematic database unless you move or delete an unnamed wire, in which case, the unnamed wire receives the next available unnamed net name.

Simple and Compound Names

A simple name is a single name that represents a single signal. Examples of simple names are:

READ
WRITE
MYGOODNESSTHISISALONGSIMPLENAME

A compound name is used to name a bus and consists of a list containing more than one net name. The compound name is formed as a list of names separated by commas. An example is:

READ,WRITE, MYNAME

which represents the three signals READ, WRITE, and MYNAME. All blank characters inside compound names are ignored.

A compound name can also be formed as a sequence of names by inserting a sequence of numbers into the name. The sequence of numbers is specified as a starting number, an ending number, and an optional increment (default is 1). The numbers are positive integers and are delimited by commas, minus signs, or semicolons. The position of the sequence within the name is indicated by enclosing the numbers with [ ]s, { }s, or ()s, and placing them in the desired position. Examples of sequential names are:

DATA[0-7] represents DATA[0] DATA[1] ... DATA[7]
ADDR(0,15,2) represents ADDR(0) ADDR(2) ADDR(4)...ADDR(14)

The number sequence might not include the ending number if the increment is greater than one (second example above).

A compound name can be a list of sequences. For example:

CLOCK,ADDR[0-3],DATA[0-7] represents CLOCK ADDR[0] ADDR[1] ... ADDR[3] DATA[0] ... DATA[7]

Entering the Net Name

The **Add->Net Name** command provides several ways to enter net names:

- A single name can be entered via the keyboard. This name is then assigned to the selected net.
- A name can be copied from an existing net or bus by clicking the mouse on the wire.
- A compound name can be entered using the keyboard. In this case, the complete compound name is attached to a wire selected by clicking the mouse. This creates a bus. The constituent signals of the bus are the individual signals represented by parsing the compound name.

  Bus names entered in this way can be very long. These long names can be wrapped onto several lines by using a backslash (\) as the continuation character in the compound name. Replace a comma (,) in the compound name where you wish the line break to occur. This multi-line naming is only available on net names at right or left ends of buses. The following example shows how the compound name is entered to achieve the multi-line effect.

  ```
  A_NAME\ANOTHER\DATA[0-3]\AND,MORE\LAST
  ```

  This results in the net name below, where the net extends off to the right.

  ```
                     A_NAME  --\
                    ANOTHER  ---\
                  DATA[0-3]  ---->--------------
                   AND,MORE  ---/
                       LAST  --/
  ```
- See the **Add->Expand Bus Name** command to learn how to break a bus name attached to the cursor into its individual components.

Renaming a Net

- A net can be renamed by:
  1. Typing the new net name.
  2. Hold down the *Shift* key while clicking the mouse on the net to be renamed. All name flags for this net on all sheets are renamed.

Attaching the Net Name

Once a name or a sequence of names has been specified, the name (or first name in the sequence) appears attached to the cursor. **Add->Name Net** provides several options for placing the net name flag. In each case, the command first performs tests to ensure that a valid connection will be made.

- The name flag can be placed at an empty point on a sheet. This will be considered an error unless a wire is eventually connected to the name flag.
- The name flag can be attached to a wire by placing the cursor on the wire and clicking the mouse. The name is connected to the wire at the point indicated while ECS adjusts the justification of the name according to the direction of the wire or wires connected at the point. If the flag is placed at an end of the wire, the net name flag extends off the end of the wire. Otherwise, if the net name flag is placed in the middle of a wire segment, the flag sits on top of the wire and is center-justified.

  If the wire was previously named, several possibilities exist.

  - If the wire has only a single net name and the new net name is applied coincident with the old net name, the net is renamed.
  - If the wire has only a single net name and the new net name is not applied coincident with the old net name, an error is flagged.
  - If the net has multiple net names and you try to apply a different net name, an error is flagged.
  - A net can be renamed by holding the *Shift* key down as the mouse is pressed to place the net name. All net flags attached to the target net are updated with the new name.
- The name flag and a single-wire segment can be simultaneously placed by pressing the mouse and dragging the cursor in either a horizontal or vertical direction to define the wire segment. If either end of the segment connects to a perpendicular wire or bus, a bus tap is created at that end of the segment. If the wire was not a bus, it is promoted into one. Placement of the name flag is determined by the ends of the segment.

  If neither end of the wire segment is connected, the name flag is placed on the ending point of the segment. If both ends are connected, the name flag is placed in the middle of the segment. If one end of the segment is connected, the name flag is placed at the unconnected end.
- A special macro command is built into the system permitting rapid connection of name flags to pins. This macro is activated when a name flag is connected to a symbol pin. Subsequent name flags can be placed by clicking the mouse with the cursor over another pin. A segment of the same length as that created by the previous command is created with the name flag attached as above.

In each of these cases, a verification feature is activated when the mouse button is pressed. If the cursor is on a net, the wires of that net are drawn in the highlighted color while the mouse button is pressed.

Once the name has been placed, the **Add->Net Name** command recycles to the input state, awaiting input of the next net name.

Editing an existing Net Name

You can edit the names of nets using the the **Add->Net Name** command. Use the following procedure to do this:

1. Select the **Add->Net Name** command.
2. Hold down the shift key while clicking on the net whose name you like to edit. The name of the selected net is placed on the prompt line.
3. You can now edit the name using arrow and delete keys. Type *Return* when your edits are complete.
4. The edited name can be placed on the schematic in the usual ways.

Global Net Names

Global signals can be accessed across all levels of hierarchy and across all sheets and schematics in a design. For this reason, any names assigned as global signals cannot be used as local signal names in your design. Figure 5-3 shows the global nets dialog box in the ECS Preferences Editor.

You can choose from three different types of global signals. They are:

Unnamed Symbol	You can attach a single name to each of the symbols in the first and third columns (see Figure 5-3). When you specify one of those names in your schematic, the corresponding symbol is attached to the net you are naming. No name is shown. The symbol is the only external indication of the name of the net. You can **Query** the net and find the net name if needed. The selection of *unnamed symbols* is limited to the power and ground symbols shown in the first and third columns.
Named Symbol	You can attach names to the symbols in the second and fourth columns. When you name a net with one of the names associated with these symbols, the corresponding symbol is drawn attached to the net. The *X's* in the right-hand column of symbols are replaced with the actual net name when placed in your schematic. The selection of *named symbols* is limited to the power and ground symbols shown in the second and fourth columns.
No Symbol	You can attach names to the box containing two *X's* at the center top of the editor. These names are global and whenever you attach one of these names to a net, it is global throughout all levels of hierarchy. The name appears attached to the net.

Global ground symbols are only drawn at the bottom of vertical wires while global supply symbols are only drawn at the top of vertical wires. In all other cases, the global signals are represented with the their name inside a box like $\boxed{\text{net\_name}}$.

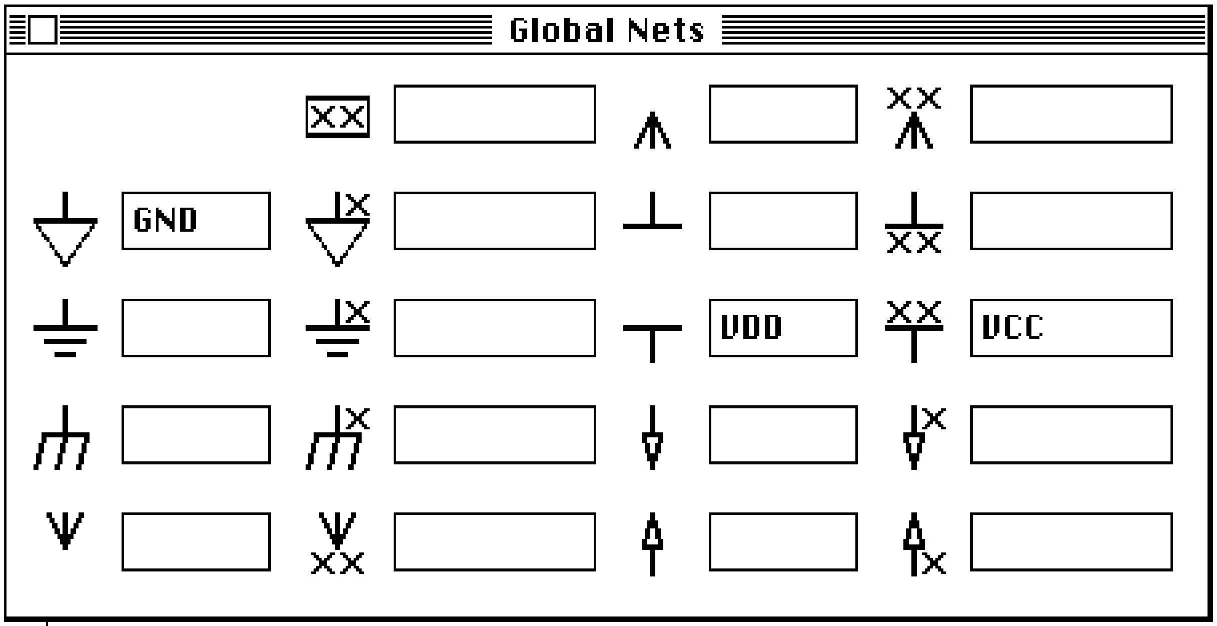


Figure 5-3 Global Nets Dialog Box

You define global nets using the following procedure.

1. Select the appropriate symbol using the mouse. This activates a text field to the right of the selected symbol.
2. Type the name of the global net next to the selected symbol. *Unnamed symbols* in the left column can only have a single name associated with each symbol. *Named symbols* and the *no symbol* icon can have multiple names associated with them with each name separated using spaces.

See Also

- **Add->Expand Bus Name** expands a bus into its constituent signals
- **Add->Bus Tap** adds bus tap to an existing wire or bus

Add->Number Pins

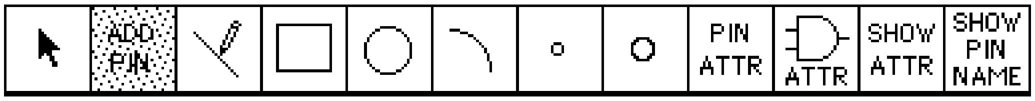
The **Add->Number Pins** command is used in the Symbol Editor to specify the pin numbers of symbol pins.

Pin numbers are used by Printed Circuit Board (PCB) applications to identify physical pins on components. ECS uses pin numbers on symbols of type component, gate and pin. The command is greyed out when you are editing a symbol that is not of type componenet, gate or pin.

Use the following procedure to add pin numbers to a symbol.

1. Select the **Add->Number Pins** command. You are prompted to type a pin number.
2. Type in the number at which the pin numbers are to begin (typically 1). This number is attached to the cursor and you are prompted to select a pin.
3. Select the symbol pin to which you wish to attach the pin number. The number attached to the cursor is automatically incremented, ready for attaching to the next numbered pin.

Add->Pin



Symbol Editor Tool Palette

The **Add->Pin** command is used in the Symbol Editor to specify the locations of the symbol pins. Pins are added to symbols to connect the device represented by the symbol to the rest of the circuit.

Pins are considered electrical elements and are therefore restricted to locations on major grid intersections. Only one pin can exist at any location.

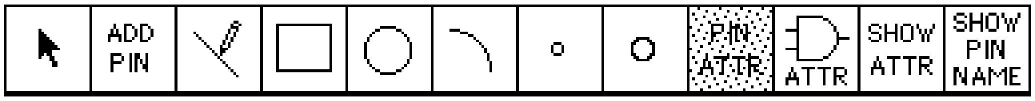
To specify the locations for the pins with this command, move the cursor to the desired location and click the mouse. The command tests for the presence of another pin. If no other pin is located at the selected point, a new pin is created. Pins are displayed in the Symbol Editor as small squares.

You can make special pin symbols to indicate things like test points and edge connectors. These Pin symbols are added to schematics with the **Add->Symbol** command. The use of these pin type symbols is described in the *Schematic Editor* section.

See Also

- *Schematic Editor* section for information on using pin type symbols to indicate test points and edge connectors

Add->Pin Attributes ⌘F Sym



Symbol Editor Tool Palette

Each pin can have a set of attributes. These attributes are assigned during the creation of the symbol and are considered the default attributes for all instances of the symbol's pins.

The **Add->Pin Attributes** command is used to add and edit the attributes associated with symbol pins. When this command is invoked, a dialog box is presented as shown in Figure 5-4. The left side of the dialog box contains a list of the names or numbers for each of the pins. Until names are assigned, this list contains rows of dashes representing the unnamed pins.

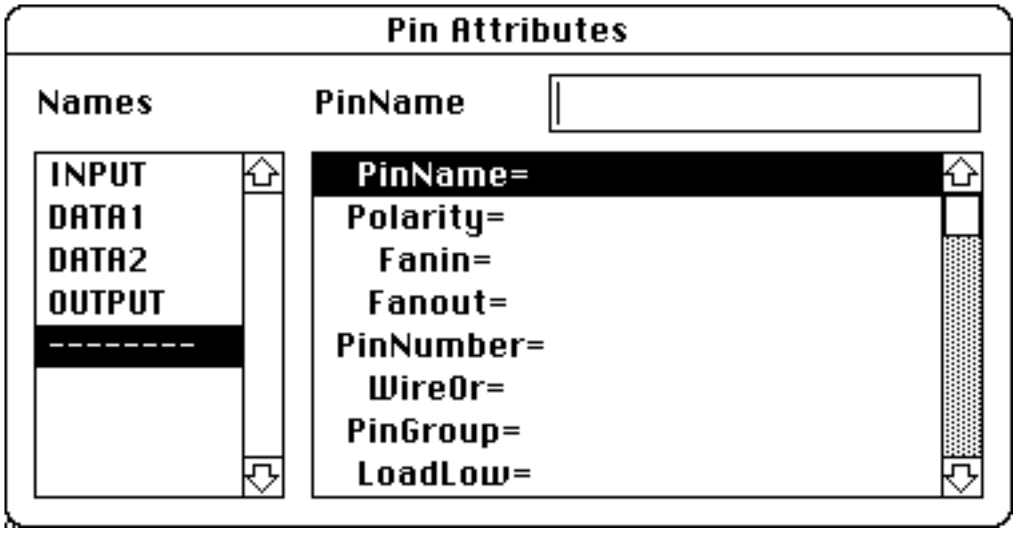


Figure 5-4 Pin Attributes Dialog Box

The right side of the dialog box contains the list of attributes that can be assigned to a symbol pin. If a pin is selected, any attributes that have been assigned to the pin are shown in the format:

name = value

Use the following procedure to assign pin attributes.

1. Select a pin with one of the following methods.
   - Move the cursor to the line in the left list box representing the pin. Click the mouse.
   - Position the cursor over the pin. Click the mouse.

The pin is highlighted, the line in the left list box is drawn in reverse color, and the right list box is updated to show the attributes assigned to the selected pin.

2. Select the attribute to be assigned or modified by moving the cursor to the corresponding line in the right-hand list box and clicking the mouse. The line is shown in reverse color and the attribute name is copied to the top line in the dialog box. If a value has already been assigned, that value appears in the edit field at the top of the dialog box.
3. Change the existing attribute value by typing a new value. (Press the *Return* key to terminate the entry). You can edit the original value by using the *Backspace* and arrow keys.
4. Use the *Up* and *Down* arrow keys to change the attribute selected in the right-hand list box.

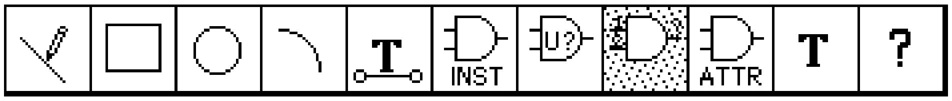
You can select pins and attributes in any order. A typical approach to assigning attributes to pins is to assign the same attribute type to each of the pins. Use the *Return* key to sequence through the pins while keeping the same attribute active.

In most cases, the pin attributes are treated as free-form text until one of the auxiliary processes uses that information to interface with an outside system. A few exceptions are:

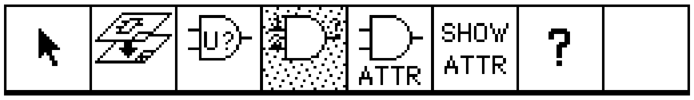
- The *Pin Name* on block type symbols must match the net connected to this pin on the underlying schematic. This is how levels of hierarchy are compiled by the Navigator. The *Pin Name* can represent a single signal or can be a compound name representing a bus (see **Add->Net Name** command). All of the signals represented by a pin must exist on the underlying schematic.
- The *Use* or *Polarity* attribute of a pin has a default value of *Input*. It is only necessary to assign values to this attribute if the *Use* or *Polarity* is other than *Input*. The other "legal" values are *Output* and *Bidir*.
- The *Load* and *Drive* attributes are used in the loading analysis in the Electrical Rules Check Process. These values must be integers and should use consistent units of measure. Input pins use the attributes *LoadLow* and *LoadHigh*. Output pins use the attributes *DriveLow* and *DriveHigh*, while bidirectional pins use both the driving and loading attributes.

Add->Pin Numbers

Sch, Nav (Misc)



Schematic Editor Tool Palette



Navigator Tool Palette

The **Add->Pin Numbers** command of the Schematic Editor and Hierarchy Navigator assigns pin numbers to Gate and Component symbols. If the design contains hierarchy, the Hierarchy Navigator is used to add the pin numbers; otherwise, the Schematic Editor can be used to add pin numbers.

Pin numbers are used in printed circuit applications to indicate which pin on a Gate or Component symbol corresponds to each physical pin on a physical device.

Gate symbols represent one section of a physical device that has more than one section. Component symbols represent a complete physical device. Gate symbols have pin numbers assigned that are actually a group of numbers. For instance, the symbol for the 7400 series NAND Gate has pins with the following names and numbers.

Name= IN1	Number= 1 4 9 12
Name= IN2	Number= 2 5 10 13
Name= OUT	Number= 3 6 8 11

The above symbol is a Gate symbol (:sym_libs:ttl:7400.sym) that represents one-fourth of a complete 7400 component. The symbol has four sections called *A*, *B*, *C*, and *D*.

The numbers on the pins of Gate symbols are initially set to the first section in the physical device. **Add->Pin Number** changes the pin numbers to another section and/or reflects a permutation of functionally equivalent pins.

To change the section of a gate, enter the pin number for any unique pin in the section. Click the mouse within the outline of the symbol and the set of pins for the section will be assigned. Alternatively, enter a letter corresponding to the desired section (e.g., *A*, *B,* etc.). Click the mouse within the outline of the symbol and the set of pins for the section will be assigned.

To swap two functionally equivalent pins, enter the desired pin number for one of the pins and click the mouse on the pin. The system will check the *Pin Group* attribute. If both pins have the same *Pin Group* and belong to the same section, the selected pin will be swapped with the pin that already has the given pin number.

Some Gates have common pins. For example, the 74175 quadruple D flip flop has a common clock and clear. The pin number attribute in this case has four identical entries as below.

Name= ENAB   Number= 4 4 4 4

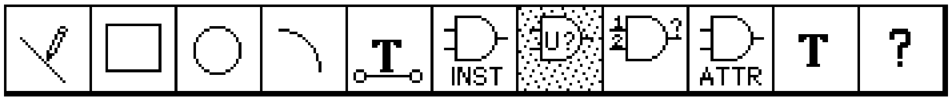
These common pins cannot be swapped.

The **Add->Pin Numbers** command of the Schematic Editor and Hierarchy Navigator allows you to click on an instance to pick up the current gate letter or to click on a pin to pick up the pin number. This makes it easier to copy the gate or pin from one instance to another or to permute pins on a gate.

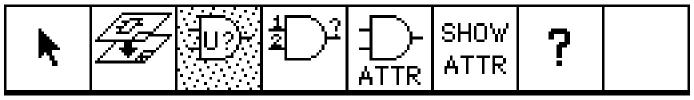
Any auto-packager (including the one in the DAT module) can be used to generate back-annotation information for the pin numbers. This back-annotation information can be added to the design using the Hierarchy Navigator.

Add->Ref Designator

Sch, Nav (Misc)



Schematic Editor Tool Palette



Navigator Tool Palette

The **Add->Ref Designator** command of the Schematic Editor and Hierarchy Navigator assigns reference designators to Gate and Component symbols. If the design contains hierarchy, the Hierarchy Navigator is used to assign the reference designators; otherwise, the Schematic Editor can be used to assign reference designators.

Reference designators designate the physical devices represented by the symbols that are used in printed circuit applications.

The reference designator must be unique for each Component symbol in the schematic. The same Reference Designator can be assigned to two different Gate symbols provided they are the same type of symbol. This permits several symbol instances to represent different sections in the same physical part. The special connector pin symbols (symbols of type *Pin*) are allowed to have duplicate reference designators. This permits edge connectors to have a single reference designator.

DeMorgan equivalent symbols, such as the 7400 and 7400D symbols, are considered to be the same type. That is, they are permitted to have identical reference designator assignments.

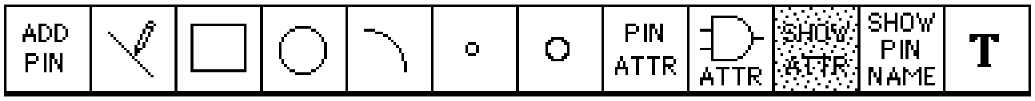
Any auto-packager, such the one described in the DAT section, can generate back-annotation information for the reference designators. This back-annotation information can be added to the design using the Hierarchy Navigator.

Add->Simulator Net Attributes

See description in *Interfaces* section under *Simulation Interface*.

Add->Show Attributes

Sym, Sch



Symbol Editor Tool Palette

Add->Show Attributes in the Symbol Editor defines the attribute windows. **Add->Show Attributes** in the Schematic Editor allows you to move the placement of attribute windows on an instance by instance basis.

In Symbol Editor

When invoked in the Symbol Editor, the **Add->Show Attributes** command presents a dialog box listing the attributes with assigned attribute window identification numbers. These numbers associate the windows with the names of the attributes to be displayed. This association is made using the ECS Preferences Editor. A temporary assignment can be made in the Schematic Editor and Navigator using the **Misc->Show Attributes** command.

The **Add->Show Attributes** command operates as follows:

1. Select an attribute window by pointing at the name of the associated attribute and clicking the mouse. The name of the attribute is attached to the cursor. The justification and text size are controlled by the settings under the **Misc->Graphics Options** command.
2. Move the cursor to the desired location and click the mouse to place the attribute window. The name of the attribute appears in the text window using the selected text size and justification. When the symbol is later placed in a schematic, the value of the attribute replaces the name.

The text string length appearing in the window might not be the same length as the attribute name. Position the window to accommodate text strings of the expected length.

Attribute windows can be reassigned after the symbol and schematic have been created. This reassignment can be done temporarily or permanently.

- Temporary reassignment of windows is accomplished by using the **Misc->Show Attributes** command in either the Schematic Editor or the Navigator.
- Permanent reassignment of attribute windows is accomplished using the ECS Preferences Editor.

In Schematic Editor

When invoked in the Schematic Editor, the **Add->Show Attributes** command operates as follows:

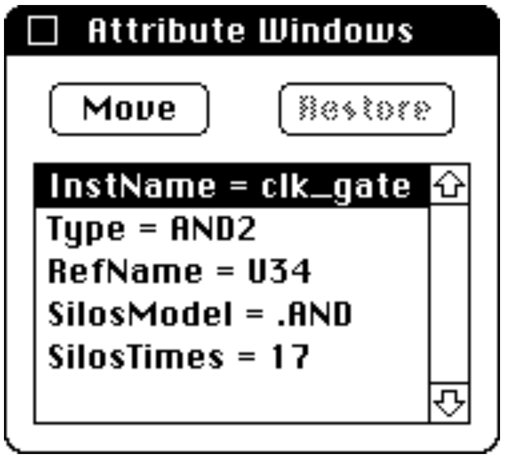
1. Select the **Add->Show Attributes** command from the menu. You are prompted to select a symbol.
2. Select a symbol instance whose attribute windows you wish to modify. A dialog box is displayed as in Figure 5-5 showing the attributes that can currently be displayed on the selected instance.

   For an attribute to appear in the list box, the attribute must have an attribute number assigned to it (assignment is made in the ECS Preferences Editor) and it must have a an attribute value assigned to it.
3. Select an attribute window to modify by selecting the appropriate attribute from the list box.
4. Select the appropriate button to perform the desired action. The two buttons display the following actions depending on the current state of the attribute window.

Add	If the selected attribute from the list box is not currently displayed on the target instance, you can add it by clicking the *Add* button.
Move	If the selected attribute from the list box is currently displayed on the target instance, you can move it to a new location on the target instance by clicking the *Move* button.
Delete	If the attribute is displayed on the selected instance but is not displayed on the symbol definition, you can remove the selected attribute from the display by clicking *Delete*.
Restore	If the attribute displayed on the selected instance has been moved from its original location on the symbol definition, you can restore the attribute display to its position on the symbol definition by clicking *Restore*.

5. In cases where the attribute window is to be newly placed, moved, or have its text size, justification or rotation changed, the attribute name is attached to the cursor. Click the cursor to place the position where the attribute is to be displayed.

   The text size, justification and rotation can be modified by changing the **Misc->Graphic Options** before placement.



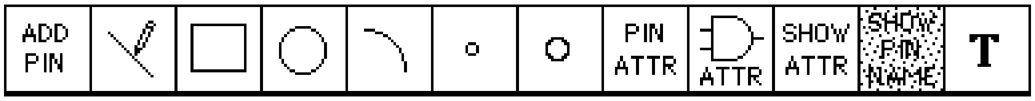
InstName = clk_gate
Type = AND2
RefName = U34
SilosModel = .AND
SilosTimes = 17

Figure 5-5 ***Add->Show Attributes*** *Dialog Box in Schematic Editor*

See Also

- *Attribute* section elsewhere in this manual for a more detailed look at attributes and attribute windows
- **Add->Attributes** to modify attributes
- **Misc->Show Attributes** reassigning attribute windows in the Navigator and Schematic Editor

Add->Show Pin Names ⌘L Sym

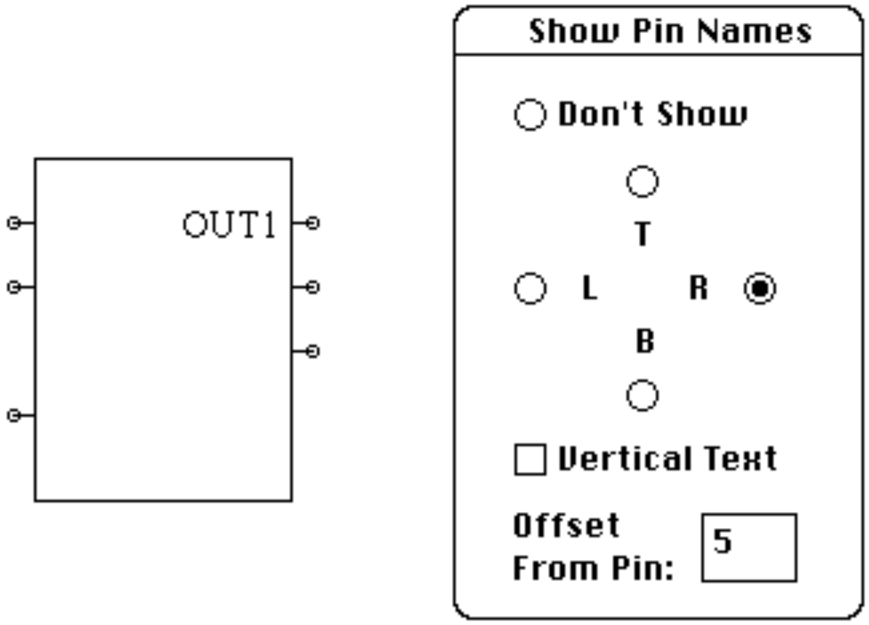


Symbol Editor Tool Palette

This command specifies where to display the pin name. The display of pin names on block and cell symbols can be controlled.

The pin name position is expressed as a *direction* and *Offset* from the pin. The direction is limited to the four primary directions (up, down, right, or left). The distances are limited to a range of 0 through 31 fine grids (quarters of the major grid).

When you select the command, a dialog box is presented. The four buttons on the sides of the box represent positions of pins with respect to a pin name. Figure 5-6 shows the settings required to label the pin *OUT1* with an offset of 5.



Show Pin Names

○ Don't Show

○ T
○ L
[✓ R]
○ B

☐ Vertical Text

Offset From Pin: 5

Figure 5-6 Pin Name Placement on Symbol With Right Offset of 5

Use the following procedure to operate the **Add->Show Pin Name** command.

1. Choose the proper pin name direction.
2. Set the *Offset* value to an integer from 0 to 31. The units are one-quarter of the major grid units.
3. Click the mouse on the symbol pin whose name you wish to display.

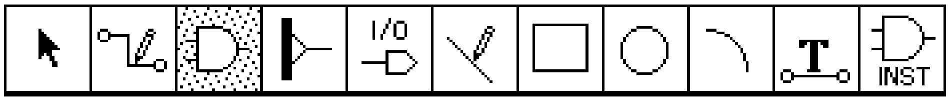
A *Don't Show* button is available to prevent any specific *Pin Names* from being displayed.

A *Vertical* button is available to force the *Pin Name* to be printed with vertical text. This is useful for pins on the top and bottom of a symbol to prevent names from running into each other.

You can specify the location of a group of pins by dragging a box around all of the pins to be specified with the same conditions. This is useful for selecting all the left-hand pins of a symbol and then all the right, top, or bottom pins in separate groups.

The initial setting for the offset is determined by the *PinNameOffset* parameter in the ECS Preferences Editor.

Add->Symbol



Schematic Editor Tool Palette

Add->Symbol is used to place instances of symbols on schematic sheets. The symbols must be previously created with the Symbol Editor. The symbol definition files can be in the local directory or in library directories. Symbol definition files have the file extension *.sym*.

Selecting the Symbol

Use one of the following methods to specify the symbol to be placed.

- Enter the file name of the symbol. Only the base name is entered, not the *.sym* suffix.
- If a symbol of the desired type is already placed on the schematic, click the mouse on one of the instances to select another symbol of that type.
- Use the symbol dialog box as shown in Figure 5-7 to select a symbol from a library. The upper list box contains a list of the library directories in the order in which they are searched for the requested symbol. One of these lines is highlighted, indicating the currently selected directory.

  The lower list box contains a list of the symbol files contained in the selected directory. Select a symbol by clicking the mouse on the desired line.

The selected symbol is the first symbol of that name that is found when the system searches the current directory and then each of the library directories. Once the symbol has been selected, the symbol outline appears attached to the mouse cursor.

Placing the Symbol

Move the symbol to the required location and use the following procedures to place the symbol in standard, rotated, or mirrored orientation.

- Click the mouse for placement in standard orientation.
- Select the **Edit->Rotate** command or use the ⌘R accelerator key. The symbol outline attached to the cursor rotates 90˚ clockwise. Click the mouse to place the rotated symbol.

  When a rotated symbol is placed, all text is adjusted so that it is in the same relative position within the symbol and is oriented horizontally or rotated 90˚ counter-clockwise. See **Edit->Rotate**.
- Select **Edit->Mirror** or the ⌘M accelerator key to display the mirror image of the symbol outline. Click the mouse in the required position to place the mirrored symbol.

All text is adjusted to read correctly and remain in the same relative position within the symbol. See **Edit->Mirror.**

- To replace an existing symbol, move the cursor so the new symbol overlaps the one to be replaced by at least 50%. Click the mouse to delete the old symbol and place the new one. If the overlap is less than 50%, the new symbol is placed overlapping the original symbol. None of the override attributes associated with the previous symbol are transmitted to the new symbol.
- Click the *Escape* key to return to the **Add->Symbol** command.

Before actual placement, ECS performs tests to check for 50% overlap of another symbol and to ensure that the symbol does not extend past the edge of the sheet. In both tests, the rectangle that encloses the symbol is used as the symbol extent box.

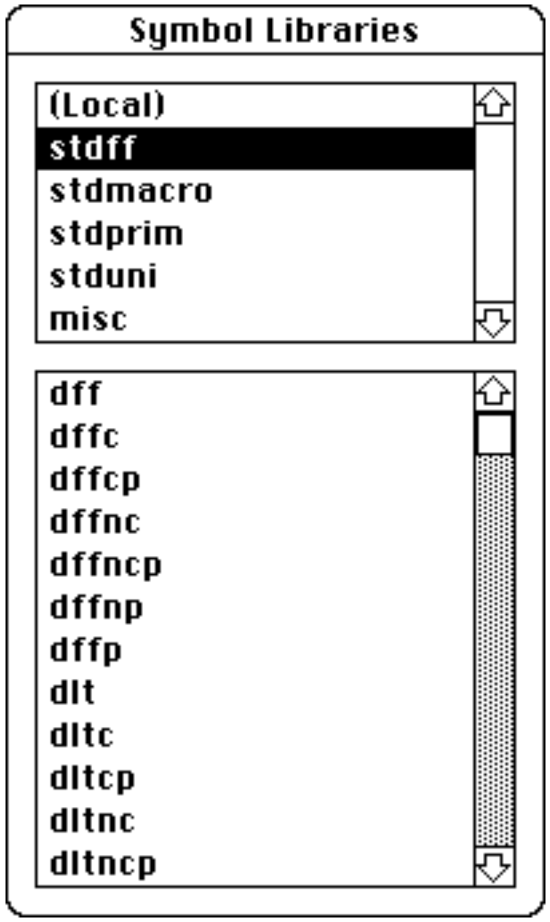


Figure 5-7 Symbol Library Selection Dialog Box

The system also tests each pin to ensure that it does not fall on the intersection of crossing wires. Because this condition results in the connection of two previously unconnected nets and in a connection of four wires on a pin, it causes the command to fail.

If all of the tests pass, an instance of the symbol — rotated and mirrored as specified — appears on the sheet. Each of the symbol pins that touch wires are connected to the respective nets.

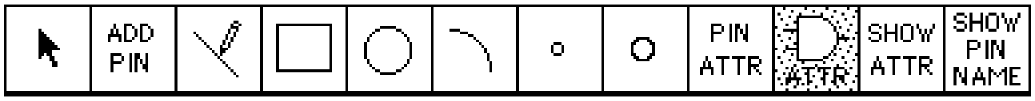
Depending on the display options currently in force, unconnected pins are marked with a flag dot; pins with two or more wires are marked with a connect dot.

Occasionally a symbol is placed so that a pin lands on the wire near, but not on its end. The resulting stub can be obscured by the symbol graphics. The condition is identified by the connect dot on the pin and the hanging end mark on the end of the stub. (Both the *Show Connect Dots* and *Mark Hanging Wires* options must be enabled using **Misc->Graphic Options.**)

See Also

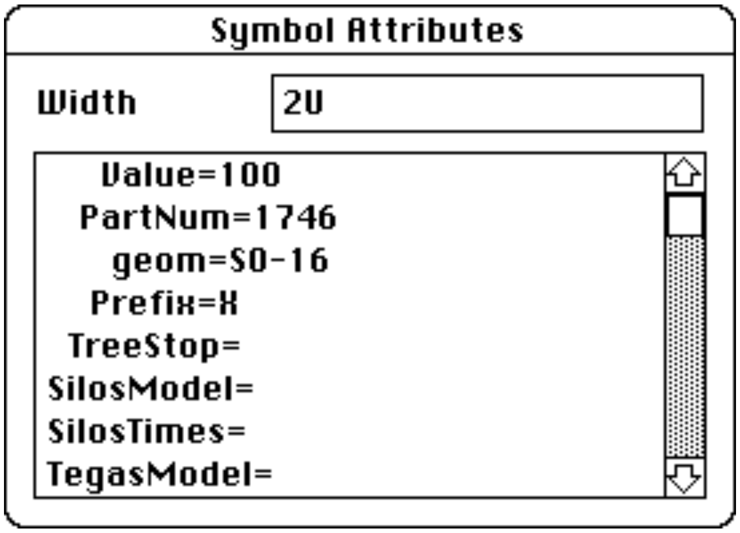
- **Edit->Mirror** to mirror symbols before placement
- **Edit->Rotate** to rotate symbols before placement

Add->Symbol Attributes



Symbol Editor Tool Palette

This command assigns values to symbol attributes in the Symbol Editor. It displays a dialog box as shown in Figure 5-8 with an edit field on the top and a list box on the bottom. The list box contains a list of all the attributes that can be modified. The edit field on top is where the values for each of the attributes are changed.



Symbol Attributes

Width 2U

Value=100
PartNum=1746
geom=SO-16
Prefix=X
TreeStop=
SilosModel=
SilosTimes=
TegasModel=

Figure 5-8 Symbol Attributes Dialog Box

Use the following procedure to set the value of a symbol attribute.

1. Click the mouse on the name of the attribute you wish to modify. The attribute name and its current value appear in the edit box.
2. Edit the attribute's value.

There are five classes of attributes used in ECS.

System	The system reserves some attributes for things like net names and instance names. These can only be modified through special system commands such as **Add->Instance Name**. These attributes do not appear in the list box of the **Add->Symbol Attributes** command.
Fixed	Values for these attributes are set in the symbol and cannot be overridden in the Schematic Editor or Navigator. A simulation model pin name is an example of an attribute that is defined once and is not modified at the instance level.
Predefined	These attributes are predefined in the symbol and can be overridden in the Schematic Editor or the Navigator. Examples of where this type of

	attribute is used are transistor width and length. This type of attribute must have a value defined on the symbol before it can be overridden in the schematic.
Open	These attributes are always accessible at all levels in the design. They are used for things like comments.
Derived	These attributes consist of a constant value or a formula that can depend on other attributes' values. Derived attributes can pass information up and down through the hierarchy. They can be overridden in the Schematic Editor or Navigator.

All possible attributes that can be assigned to a given symbol are visible in the Attribute Editor dialog box when you edit a symbol in the Symbol Editor. These attributes are defined in the ECS Preferences Editor. Access to the definitions is through the **Attribute** menu of the ECS Preferences Editor.

See Also

- The *Attribute* section in this manual for a detailed description of attributes
- The *System Configuration* section, which details how to modify attribute definitions in the ECS Preferences Editor
- **Add->Pin Attributes** for a description of adding pin attributes in the Symbol Editor
- **Add->Show Attributes**, which allows adding attribute windows in the Symbol Editor
- **Misc->Show Attributes**, which allows attribute windows to be modified in the Navigator

Add->Table

The **Add->Table** command in the Schematic Editor allows you to add a table of information which is graphically displayed on the schematic. This can be for purely cosmetic reasons or documentation purposes. Elements in the schematic can reference values stored in these tables.

Graphical Format

The tables display rows and columns of data with lines separating the rows and columns. The first row of the table displays the column labels. The height of the first row can be different from the height of the remaining rows. The text displayed in the first row can have different size and justification from the rest of the table. This allows column labels to be written vertically.

The first column of the table displays the row labels. The width of the first column can be different from the width of the remaining columns. The text displayed in the first column can have different size and justification from the rest of table. The row labels are typically instance names or reference designators of symbol instances in the schematic. The column labels are typically names of attributes or the value of a global constant.

In addition to the above, each table has an optional title which can be written above the top of the table. Whenever text is displayed, the justification of the text determines where the text is written. Left justified text is displayed at the left of the field. Each table has a name which identifies the table to the symbol instances.

The **Add->Table** command brings up a dialog box as shown in Figure 5-9. This dialog box provides independent control of the following parameters:

- number of rows
- number of columns
- height of first row
- height of remaining rows
- width of first column
- width of remaining columns
- font size, justification and rotation of column labels (first row)
- font size, justification and rotation of row labels (first column)
- font size, justification and rotation of remaining data
- font size, justification and rotation of the title
- table title (any text the user wants displayed at the top)
- table name (used to identify table for symbol instances)

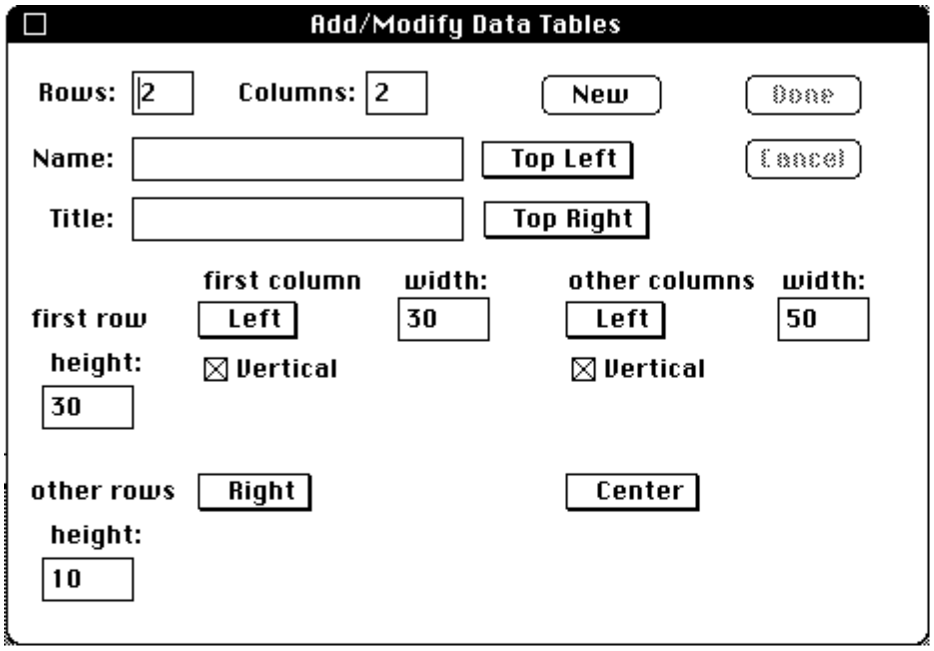

Rows: 2
Columns: 2
Name:
Title:
first row height: 30
first column: Left
width: 30
☒ Vertical
other columns: Left
width: 50
☒ Vertical
other rows height: 10

Figure 5-9 Dialog Box for ***Add->Table*** *Command*

The **Add->Table Data** command is used to enter the data into the table after you have created the table.

Accessing Table Data

Symbol attributes are able to reference values stored in tables by using derived attributes. Since derived attributes are part of the Design Analysis Tools (DAT), you must have the DAT module to use the full power of the table functions.

The symbol instances which need to access data from a table do so using derived attributes. The general information required to specify data from a table is:

get value from table *T*, row *R* column *C*

There are several methods for specifying *R* and *C* to access table data. The simple method for rows is to enter the row number. The more flexible method is to request the row whose label is equal to *value* where *value* may be specified as the value of an attribute (e.g. RefDes). The simple method for specifying which column of the table to use is to enter the column number. A more flexible method is to request the column whose label is equal to *constant* where *constant* can be the value of a global constant.

A useful method of accessing rows is to reference them by instance name. You can create row labels that correspond to instance names in a schematic. When you reference the table from a given instance, a search routine scans the row labels to see if any correspond to the given instance name. This allows you to change the order of rows in the table without destroying the data references.

Syntax:
To access table data using derived attributes use the following syntax:

&table_name[row,column] ...

table_name	is the table's name spelled exactly
row	may either be a number or *=value* or *=#attr*. When row contains an equal sign the table is searched to find the row that has a label equal to *value* or equal to the value of attribute *attr* on the current symbol instance. Row numbers start at one. Row zero is the column label.
column	may either be a number or *=value* or *=$const*. When a column contains an equal sign, the table is searched to find the column that has a label equal to *value* or equal to the value of the global constant *const*. Column numbers start at one. Column zero is the row label.

Examples:

The following examples show how an attribute of a symbol can be specified in order to get the data from a table. In this example, the table *New* contains resistor values that correspond to symbol attribute #3 whose name is *Value*. In this way, the resistor's value is determined from the table rather than from the value attribute directly.

&New[2,3]	gets the data from row 2 and column 3 of the table named *New*.
&New[=R1,=Resistance]	gets the data from the column labelled *Resistance* and the row labelled *R1*.
&New[=#[RefDes],3]	gets the data from column 3 and the row whose label matches the *RefDes* of the current symbol.
&New[=#2,$[country]]	the row whose label matches the value of attribute #2 (RefDes) of the current symbol determines which row the data is taken from. The global constant named *country* determines which column to use.

Since global constants can be used anywhere for string substitution, it is possible to avoid having to type in one of the strings above for the value attribute. By setting a global constant to one of the strings in the above example, the symbol instances which need to reference attributes from the table need only reference the global constant. For example, set the global constant 0 to *&New[=#2,$[country]]*. Then set the Value attribute of the symbol to *$0*. This causes the Value attribute to use the table to derive its data.

See Also

- **Add->Table Data**

Add->Table Data

The **Add->Table Data** command is used to enter or modify data in tables. Tables are first created using the **Add->Table** command.

The data entered can be any keyboard character except a carriage return. Carriage returns are used to terminate the entry of data.

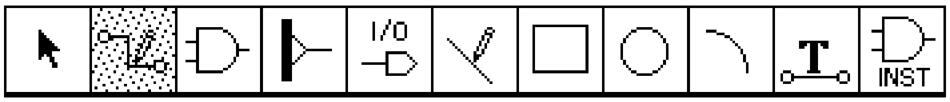
Use the following procedure to modify existing data or enter new data.

1. Select the **Add->Table Data** command. You are prompted to select an entry from an existing table.
2. Select a table entry from an existing table. The selected element in the table is highlighted.
3. Enter the required data by typing. Finish an entry with a carriage return. The next element inthe table is highlighted allowing you to enter data for the next element in the table.

See Also

- **Add->Table**

Add->Wire



Schematic Editor Tool Palette

Add->Wire is the main command for connecting wires between symbol pins. **Add->Net Name** and **Add->Bus Tap** can also be used to add single-wire segments to the design.

The **Add->Wire** command operates in several modes. The mode of the command is established when the first point is entered.

Point-to-Point Wiring

1. Click the mouse on the first point. The first point is established as the starting point for the wire and a rubber band wire is stretched to the new mouse location.
2. Each time the mouse is clicked, another segment of wire is placed. If the new segment ends on a pin, name flag, or other wire, the path is terminated and the command returns to its initial state. The path can also be terminated by clicking a second time on the last point of the wire.

   Wires are normally constrained to the four 90° directions. If the *Shift* key is held down while the mouse is pressed, the wire segment placed immediately after can take diagonal (45°) orientations as well.

   This is the most common mode used for adding wires. It is also the only mode in which diagonal (45°) wires can be added.

Single-Segment Wiring

1. Drag the mouse horizontally between two points. In this mode, a single-wire segment is added, and the command returns to the initial state.

Z and C Connections

1. Drag the mouse between two points *not* on a horizontal line. A three-segment connection is assumed. The first attempt shows a connection consisting of horizontal wires from each of the two points to the vertical position of the cursor and a third vertical wire segment between the ends of the horizontal wires. As the mouse is moved, the connection stretches horizontally to form a C- or Z-shaped pattern.
2. Clicking the mouse attempts to add the wire pattern to the drawing.
3. If the *Escape* key is pressed, the pattern is reversed so that the wires connected to the points are vertical and the connecting segment is horizontal.
4. A second push of the *Escape* key aborts the command and returns to the initial state of the add wire command. Figure 5-10 shows the Z and C connections.

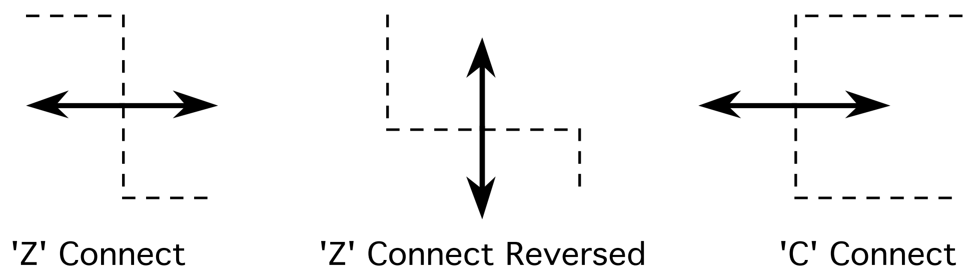
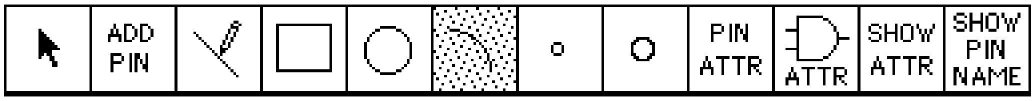


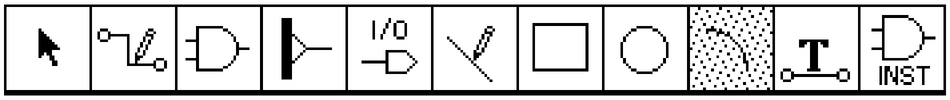
Figure 5-10 Z and C Wire Connections

Crossing wires are not normally connected. It is possible to form a four-way connection on orthogonal wires by first creating a T connection and then adding the fourth wire. This type of connection is automatically marked with a connect dot to distinguish it from unconnected crossing wires.

Draw->Arc



Symbol Editor Tool Palette



Schematic Editor Tool Palette

The **Draw->Arc** command adds arcs to the drawing. These arcs are graphic only and do not have any electrical significance in the schematic or symbol.

The starting and ending points of the arc can be defined using either the click or drag method.

Click	Click the mouse when the cursor is at the starting point of the arc and then move the mouse to the ending point and click the mouse again.
Drag	Press the mouse when the cursor is at the starting point, drag the cursor to the ending point, and then release the button.

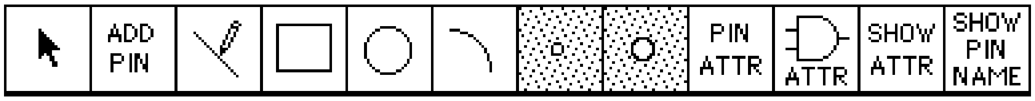
In either case, the command shows an arc that starts and ends on the selected points and passes through the cursor. As the cursor is moved, the shape of the arc changes.

Click the mouse again to fix the third point of the arc. The resulting arc is added to the drawing.

If you move the cursor to positions that specify an arc with too large a radius to be calculated accurately, the arc disappears until you move the cursor back within range.

Press the *Escape* key to discard the selected points and restart the command.

Draw->Bubble / Big Bubble



Symbol Editor Tool Palette

The **Draw->Bubble** command adds negation bubbles to symbols. These bubbles are graphic only and do not have any electrical significance.

The bubbles are circles that are incorporated into special commands for convenience. Once placed, they become circles that can be moved and copied. They can also have their sizes changed by the **Edit->Drag** command.

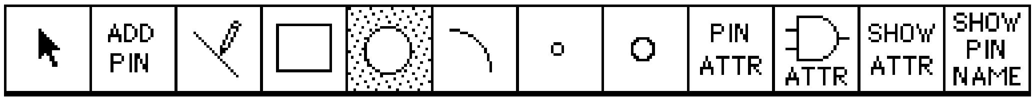
While the **Draw->Bubble** command selects a bubble that is one-half of a major grid in diameter, the **Draw->Big Bubble** command selects a bubble that is one major grid in diameter.

When either command is selected, a bubble of the correct size is attached to the cursor. The cursor is also adjusted to track on a finer grid. The grid selected is the minimum of the radius of the bubble or half the grid that is presently active.

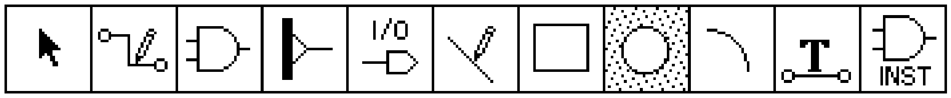
Each time the mouse is clicked, a bubble is added to the drawing.

Draw->Circle

Sym, Sch



Symbol Editor Tool Palette



Schematic Editor Tool Palette

Circles can be placed with the **Draw->Circle** command. These circles are graphic only and do not have any electrical significance in the schematic or symbol. This command operates in both click and drag modes.

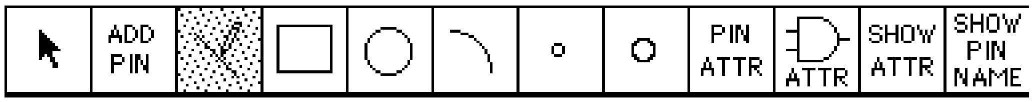
Click

1. Choose the center of the circle by clicking the mouse.
2. Move the cursor to define the size of the radius.
3. Click the mouse again to draw the circle.

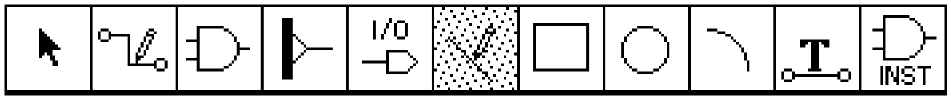
Drag

1. Press the mouse.
2. Drag the cursor. The diameter of the circle equals the distance between the first point and the cursor. The center of the circle is the mid-point of the line joining the first point to the cursor.
3. Release the mouse to draw the circle.

Draw->Line



Symbol Editor Tool Palette



Schematic Editor Tool Palette

The **Draw->Line** command adds individual graphic line segments to the drawing. These line segments are graphic only and have no electrical meaning in the schematic or symbol.

The **Draw->Line** command operates in either the click mode or the drag mode.

Click

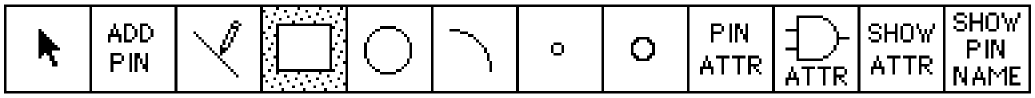
1. Click the mouse at starting point of line. The cursor is then constrained to the eight 45° directions.
2. Each subsequent click of the mouse adds a single line segment to the drawing.
3. The line is terminated by clicking the *Escape* key or clicking twice on the last point of the line. The command automatically restarts after a line has been terminated.

Drag

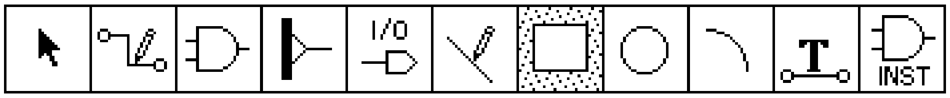
1. Press the mouse. This sets the starting point of the line.
2. Drag the cursor to any point and release. A line is drawn between the two points. The resulting line can be placed at any angle. The command waits for the next point.

Draw->Rectangle

Sym, Sch



Symbol Editor Tool Palette



Schematic Editor Tool Palette

The **Draw->Rectangle** command adds rectangles to the drawing. These rectangles are graphic only and have no electrical meaning in the schematic or symbol.

The **Draw->Rectangle** command operates in either the click mode or the drag mode.

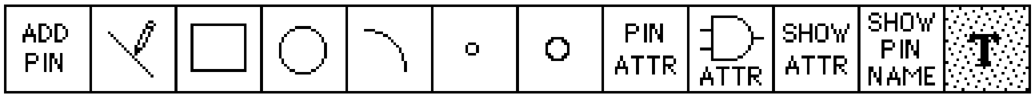
Click

1. Click the mouse at the first corner of the rectangle.
2. Move the cursor to the corner diagonally opposite of the rectangle.
3. Click the mouse to draw the rectangle. No area rectangles are discarded.

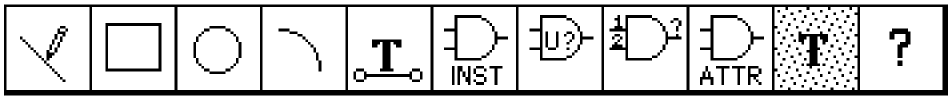
Drag

1. Press the mouse at first corner of rectangle.
2. Drag cursor to the corner diagonally opposite of the rectangle.
3. Release the mouse to draw the rectangle. No area rectangles are discarded.

Draw->Text



Symbol Editor Tool Palette



Schematic Editor Tool Palette

The **Draw->Text** command adds fixed text to the drawing. Text is used for placing notes, creating title blocks, creating symbol graphics, and generally adding information to the symbol or schematic. Text has no electrical significance in the symbol or schematic.

The size and justification of the text is determined by control parameters that are set using the **Misc->Graphic Options** command. Three sizes of text are available. Text can be placed as either left-justified, center-justified, or right- justified.

The **Draw->Text** command operates in either the *Add* or *Edit* modes.

Add Type the required text, then press the *Return* key. Use the **Misc->Graphic Options** command to change the font size and justification while the text is being entered. The text can be fixed to the symbol or schematic by moving the cursor to the required location, then clicking the mouse.

Edit Select an existing string of text by clicking the mouse while the cursor is over the text. The text is removed from the drawing and placed on the edit line where it can be modified. The justification and font size can also be changed. When the modification is complete, the text can be placed back into the drawing at its original position by pressing the *Return* key.

Edit->Clear *Delete* key Sym, Sch, Nav

This command deletes items you have selected from the schematic or symbol drawing. This command does not affect the contents of the clipboard.

Once a set of data is in the selected set (see **Edit->Select** for information on selecting data) it can be deleted from the drawing by choosing the **Edit->Clear** command. A shortcut for the **Edit->Clear** command is to use the *Delete* key. Use this command instead of **Edit->Cut** when the clipboard contains data you don’t want to overwrite.

See Also

- **Edit->Cut**
- **Edit->Copy**

Edit->Copy ⌘C Sym, Sch

The **Edit->Copy** command copies selected items from the symbol or schematic drawing to the clipboard. The items can then be **Pasted** from the clipboard into another schematic or symbol, or even another Macintosh application.

Once a set of data is in the selected set (see **Edit->Select** for information on selecting data), it can be copied to the clipboard by choosing the **Edit->Copy** command. The original data in the schematic or symbol is unaffected by the **Edit->Copy** command.

Use this command instead of **Edit->Duplicate** when you want to transfer parts of your symbol or schematic to another drawing or another Macintosh application.

See Also

- **Edit->Duplicate**
- **Edit->Select**

Edit->Cut ⌘X Sym, Sch

The **Edit->Cut** command deletes selected items from the symbol or schematic drawing and places them on the clipboard. The items can then be **Pasted** from the clipboard into another schematic or symbol, or even another Macintosh application.

Once a set of data is in the selected set (see **Edit->Select** for information on selecting data) it can be copied to the clipboard by choosing the **Edit->Cut** command. Use this command instead of **Edit->Clear** when you want to transfer parts of your symbol or schematic to another drawing or another Macintosh application.

See Also

- **Edit->Clear**
- **Edit->Select**

Edit->Drag

The **Edit->Drag** command is used to move portions of objects while the rest of the object remains stationary. *Stretching* is perhaps the best word to describe what **Edit->Drag** does.

This command is the only **Edit** command that does not act upon previously selected data. To operate the command, you first choose **Drag** from the menu. You then use the mouse to select the data on which to operate. The command works differently in the Symbol and Schematic and Editors, as described below.

Symbol Editor

The **Edit->Drag** command in the Symbol Editor moves one of the defining points of a graphic element. An item is selected by pressing the mouse while the cursor is on or very close to it. The mouse is then dragged, causing the graphic object to be stretched or deformed. The command applies to the following graphic elements in the manner described.

Lines	Moves the end nearest the point at which the line was selected.
Rectangles	If the selection point is near a corner, the rectangle is stretched by moving the corner. If the point is closer to the center of a side, the rectangle is stretched by moving the side.
Circles	The radius of the circle is changed while the center remains fixed.
Arcs	The end of the arc nearest the point where the arc is selected is moved to increase or decrease the angle of the arc. The center and radius of the arc remain fixed.

Schematic Editor

The **Edit->Drag** command in the Schematic Editor allows you to move one or more elements while retaining electrical connectivity. The command also provides the functions for editing the graphic elements that are available in the Symbol Editor.

The command operates in two modes. The first mode is the drag-point mode. The mouse is clicked on an item in the schematic and then the mouse is dragged to reposition the selected item. Three cases can occur:

- If the initial selection point is within the extent box of a symbol, the command switches to the drag-box mode, using the extent box as the area to be dragged.
- If the point is on a wire or net name flag, the command highlights the name flag and/or any wires connected to the point. It then shows the connections as moving lines connected to the cursor. Depending on the pattern of wires being moved, the cursor can be limited in motion.

  The connection can be dragged to its desired location and fixed by releasing the mouse. Assuming that the connections do not violate any of the interconnection constraints, the connections are made.

- If the point is on a graphic element, the drag command operates in the same manner as described for the Symbol Editor.

The second mode is drag-box mode. This is defined by dragging the cursor to define a rectangular area or by clicking the mouse while the cursor is within the extent box of a symbol. In the latter case, the extent box of the symbol is used as the rectangular area.

The command highlights all of the items that are within the border of the area, including wires that are partially enclosed. Symbols must be totally enclosed by the area in order to be included.

The highlighted items are then connected to the cursor. Any connections that spanned the border of the area are maintained. As the mouse is moved, these connections are maintained with three-segment connections from the edge of the area to the original points of connection.

The cursor is constrained to maintain the connections that are broken. When the mouse is clicked, the items are relocated into their new positions on the schematic and the new connections are made.

The command can fail to make all of the connections because of rules violations. If this occurs, the system returns the drawing to its original state.

The command fails if the boundary of the area is on the intersection of two crossing wires. See the **Edit->Select** command for a discussion of this problem.

See Also

- **Edit->Select**, which can be used to drag by holding down the *Option* key during select

Edit->Duplicate D Sym, Sch

The **Edit->Duplicate** command copies selected items from the symbol or schematic drawing to a different location on the same drawing. This command does not affect the contents of the clipboard.

Once a set of data is in the selected set (see **Edit->Select** for information on selecting data) it can be copied to another location on the same drawing by choosing the **Edit->Duplicate** command. A copy of the selected data is attached to the cursor. You can position the data to the desired location on the drawing and then lock it in place by clicking the mouse. The original data in the schematic or symbol are unaffected by the **Edit->Duplicate** command.

Net names and attributes are copied as well as the physical representation of the wires and symbols. Instance names are not duplicated because every instance must have a unique name.

See Also

- **Edit->Copy**, which copies data to the clipboard to be moved to other drawings
- **Edit->Select**, which is required prior to this command

Edit->Mirror

As the name suggests, **Edit->Mirror** reflects selected data through an imaginary vertical line attached to the cursor, each time the command is selected. Selected data to the right of the cursor moves to the left and vice versa. Together with the **Edit->Rotate** command, eight standard orientations of data are possible, as shown in Figure 5-11.

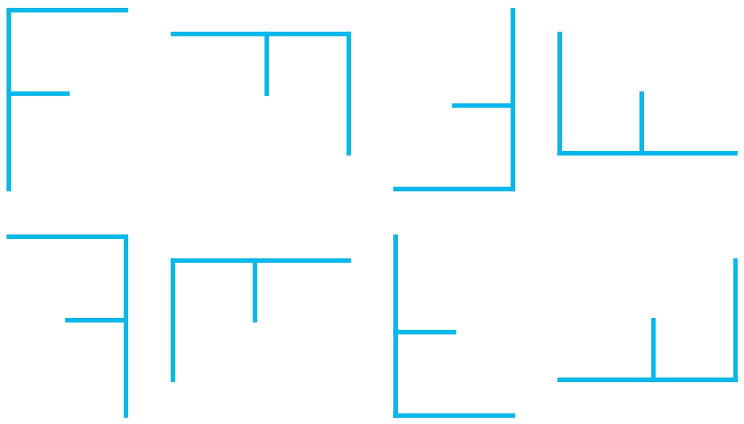


*Figure 5-11 Eight Possible Orientations of Data Using **Mirror** and **Rotate***

Mirror works with data from the **Edit->Duplicate**, **Edit->Move**, **Edit->Paste**, and **Add->Symbol** commands as described below.

Edit->Duplicate
This command allows you to copy symbols and wires. After you choose the **Edit->Duplicate** command, the data are attached to the cursor. At this point, you can mirror the data by using the **Edit->Mirror** command. Each successive **Mirror** toggles the orientation between the original orientation of the data and the mirrored or reflected orientation.

Edit->Move
This command allows you to move symbols and wires. After you choose the **Edit->Move** command, the data is attached to the cursor. At this point, you can mirror the data by using the **Edit->Mirror** command. Each successive application of **Mirror** toggles the orientation between the original orientation of the data and the mirrored or reflected orientation.

Edit->Paste
This command allows you to paste data from the clipboard onto the drawing. After selecting the **Edit->Paste** command, the data from the clipboard is attached to the cursor. At this point, you can mirror the data by using the **Edit->Mirror** command. Each successive **Mirror** toggles the orientation between the original orientation of the data and the mirrored or reflected orientation.

Add->Symbol
When this command is invoked, the symbol is attached to the cursor and is placed when you click the mouse. **Edit->Mirror** is used before the symbol is placed. It causes the symbol to be mirrored or reflected about a vertical axis through the cursor. The symbol is placed by clicking the mouse after the **Edit->Mirror**.

Symbol text is not shown on the outline connected to the cursor. Text that appears as part of a symbol is adjusted so that it is placed and justified in the same relative position within the symbol. It always reads in either a left-to-right or bottom-to-top direction. Figure 5-12 illustrates this more clearly.

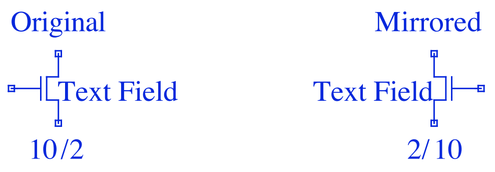


Figure 5-12 Mirrored Transistor, Note Reversal of Two Attributes

Note that to avoid the width and length attributes being displayed in the wrong order, as in the mirrored transistor in Figure 5-12, you must combine the two attributes into a single attribute. Refer to the *Attribute* section for more information.

Edit->Move

Sym, Sch

The **Edit->Move** command moves selected items from a symbol or schematic drawing to a different location on the same drawing. This command does not affect the contents of the clipboard.

Once a set of data is in the selected set (see **Edit->Select** for information on selecting data) it can be moved to another location on the same drawing by choosing the **Edit->Move** command. The selected data is attached to the cursor. You can position the data to the desired location on the drawing and then lock it in place by clicking the mouse.

The **Edit->Move** command is automatically linked with the **Edit->Select** command in two ways. If you have selected a set of data and the last command executed was a select operation, you can move the selected data by dragging it. You do not need to choose the **Edit->Move** command from the menu. It is automatically enabled by the **Edit->Select** command.

A special case of this arises if you are using the single-item select mode. In this case, if you are in select mode (the arrow in the tool palette is highlighted), then you can directly drag the single item to a new location. The select and move operations are combined into a single click-and-drag motion of the mouse.

The **Move** command has the following advantages over a copy-and-paste sequence:

- It is one or two less commands to enter and thus faster to execute.
- It does not affect the contents of the clipboard while **Edit->Copy** overwrites the clipboard.

See Also

- **Edit->Copy**, which copies data to the clipboard to be moved to other drawings
- **Edit->Select**, which is required prior to this command

Edit->Paste

⌘V Sym, Sch, Nav

Edit->Paste is used to move items into the schematic or symbol. The items to be pasted must first be either cut or copied from a schematic or symbol onto the clipboard. Schematic and symbol items are not compatible; you cannot move items between schematics and symbols.

When the **Edit->Paste** command is selected, the items that are on the clipboard are attached to the cursor. They can then be located by moving the mouse to the desired position and clicking the mouse.

The **Edit->Rotate** and **Edit->Mirror** commands can be used to rotate and mirror the data to be pasted before placement.

Edit->Paste fails if any of the items extend beyond the border of the sheet or symbol. In this case, the group remains attached to the cursor and a second attempt can be made.

The command can partially fail if a rules violation occurs when you are placing one or more items. Use the **Edit->Undo** command to erase the remaining items and restore the drawing to its original state.

You can add additional copies of the clipboard items by reissuing the **Edit->Paste** command. The clipboard is not cleared until another **Edit->Cut** or **Edit->Copy** is performed.

Edit->Redo

Sym, Sch, Nav

Edit->Redo allows you to recover from changes caused by the **Undo** command. As the **Undo** command backs up through the session, a log is retained. **Redo** can go forward, playing back the undone events until it reaches the end of the log (the point when the first **Undo** in the sequence was issued).

The **Redo** log is discarded whenever a command (other than **Undo** or **Redo**) causes an entry in the database. Once this log is discarded, **Redo** cannot recover undone commands.

The command does not work with the **File->Sheet** command. That is, if **Undo** is used to restore an original sheet size after changing the sheet size, **Redo** cannot restore the drawing to the changed sheet size.

See Also

- **Edit->Undo**, which allows recovery from **Redo**

Edit->Rotate

▯R Sym, Sch

Edit->Rotate rotates data by 90° each time the mouse is pressed. Together with the **Mirror** command, eight standard orientations of data are possible as was shown in Figure 5-8.

Edit->Rotate works with data from the **Add->Symbol**, **Edit->Duplicate**, **Edit->Paste**, and **Edit->Move** commands as described:

Edit->Duplicate
This command allows you to copy symbols and wires. After you choose the **Edit->Duplicate** command, the data is attached to the cursor. At this point, you can rotate the data clockwise about the cursor by using the **Edit->Rotate** command. Each successive **Rotate** rotates the orientation another 90° in a clockwise direction.

Edit->Move
This command allows you to move symbols and wires. After you choose the **Edit->Move** command, the data is attached to the cursor. At this point, you can rotate the data clockwise about the cursor by using the **Edit->Rotate** command. Each successive **Rotate** rotates the orientation another 90° in a clockwise direction.

Edit->Paste
This command allows you to paste data from the clipboard onto the drawing. After selecting the **Edit->Paste** command, the data from the clipboard is attached to the cursor. At this point, you can rotate the data clockwise about the cursor by using the **Edit->Rotate** command. Each successive **Rotate** rotates the orientation another 90° in a clockwise direction.

Add->Symbol
When this command is invoked, the symbol is attached to the cursor and is placed when you click the mouse. **Edit->Rotate** is used before the symbol is placed. It causes the symbol to be rotated by 90° clockwise about the cursor. The symbol is placed by clicking the mouse after the **Rotate**.

Symbol text is not shown on the outline connected to the cursor. Text that appears as part of a symbol is adjusted so that it is placed and justified in the same relative position within the symbol. It always reads in either a left-to-right or bottom-to-top direction. Figure 5-13 illustrates this more clearly.

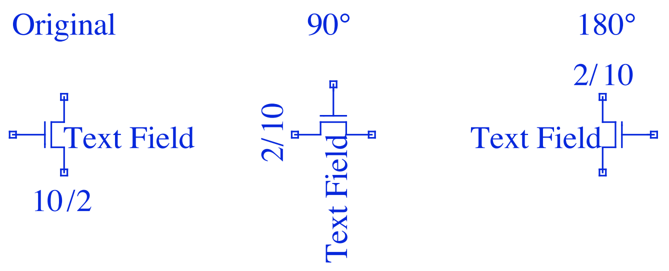
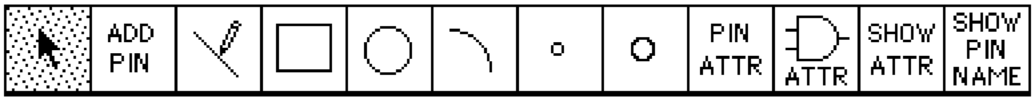
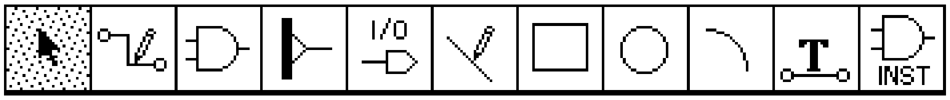


Figure 5-13 Transistors Rotated Through 90° and 180°

Edit->Select



Symbol Editor Tool Palette



Schematic Editor Tool Palette

The **Edit->Select** command identifies those objects in a schematic or symbol drawing on which you wish to operate. There are several ways to identify or select objects in schematics and symbols. Once you have selected one or more objects with the **Edit->Select** command, there are a number of operations you can perform on the selected data. These operations include **Cut**, **Copy**, **Paste**, **Clear**, **Move**, and **Duplicate**.

There are three different ways to select objects: single-item select, box select, and group select. They are described below.

Single-Item Mode
Use the following procedure to select a single object.

1. Choose the **Edit->Select** command from the menu or the tool palette.
2. Move the cursor to the required item and click the mouse. If the cursor is pointing at two or more items when the mouse is clicked, the following priority order is used to determine which item to select.

Schematic Editor Priorities

1. Name Flags	Click on the connect point.
2. Wire Segments	Click anywhere on the wire.
3. Symbols	Click within the symbol extent box.
4. Graphics	Click on the graphic element.
5. Text	Click on the text.

Symbol Editor Priorities

1. Symbol Pins	Click on the pin.
2. Graphics	Click on the graphic element.
3. Text	Click on the text.
4. Attribute Windows	Click within the window text area.

Box-Select Mode

In the box-select mode, the items contained in a rectangular area are selected. Press the mouse in one corner of the target area and then drag the cursor to form a box around the area. When the mouse is released, all of the items totally enclosed by the box are selected. Portions of wire segments enclosed by the box are also selected. Net names and attributes are selected as well as the physical representation of wires and symbols.

If the corner of the box coincides with the crossing point of a vertical and horizontal wire, selection of the two partial segments leaves the remaining segments ending on the original crossing point. Many potential operations would then remove the selected segments, leaving the two remaining segments connected. Because this is not allowed, you cannot select two portions of nets, as shown in Figure 5-14.

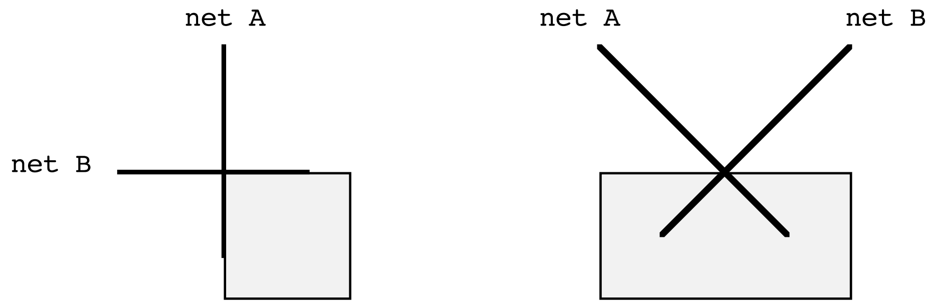


Figure 5-14 Illegal Delete Selections

A similar condition arises when there are crossing diagonal wires that cross on the perimeter of the rectangle. (See Figure 5-14.) The selection of the portions of the wires inside the box also leaves the remaining wires potentially connected following a **clear** or **move** command. For this reason, you are not allowed to select the diagonal wires, as shown in Figure 5-14.

If both of the nets are named (differently), either of the above conditions is an error. If either or both nets were unnamed, the results are technically correct, but probably not intended. If either condition occurs, the **select** command is rejected.

Group-Select Mode

The group-select mode combines the two other modes allowing random collections of items to be selected. To use the group-select mode:

1. Press the *Shift* key prior to selecting the first item or area.
2. Select one or more items and/or areas while holding down the *Shift* key. As each item or area is selected, it is redrawn in the Phantom color to indicate that it has been selected.

3. The group of items is selected and can be used by other **Edit** commands. Because the relative position of the items in a selected group is maintained, it is not possible to group items from different sheets.

Drag Option

There is an option of the **Edit->Select** command that causes it to act like the **Edit->Drag** command. Hold down the *Option* key while performing a select operation to activate this drag option.

Edit->Undo Z Sym, Sch, Nav

The **Edit->Undo** command allows you to back up in the edit process. The **Edit->Undo** command operates on all commands that result in database changes in the Schematic and Symbol Editors. It does not recognize any of the view control commands.

Edit->Undo backs up one event each time it is issued. The command is capable of undoing all of the events back to the last time the file was opened or saved.

If you **Undo** too far back by mistake, you can recover your edits by using **Edit->Redo**.

See Also

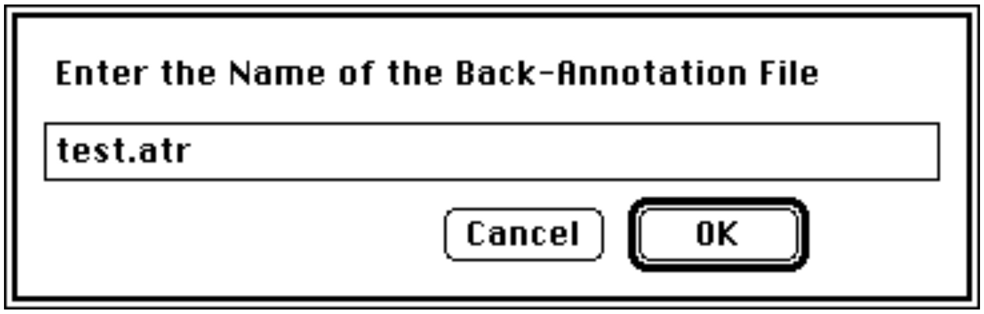
- **Edit->Redo**, which allows recovery from **Undo**

File->Back Annotate

The **File->Back Annotate** command is used to back annotate attribute values to the Hierarchy Navigator. This interface allows you to back annotate values obtained from other CAD tools onto your designs. The typical case in IC design where this is used is to add back onto the design all the loading parasitics determined during the layout process. This allows the schematic to accurately reflect the true state of the design at all times.

Use the following procedure to use the command.

1. Select the **File->Back Annotate** command from the menu. A dialog box similar to that in Figure 5-15 is displayed.
2. Type the file name containing the attribute information to be back annotated.
3. Click the OK button. The attribute information in the back annotation file is read into the Hierarchy Navigator.
4. If you wish to record the back-annotated changes, you **must** perform a **File->Save** command to save the attribute modifications to the *.tre* Navigator file.


Enter the Name of the Back-Annotation File

test.atr

Figure 5-15 ***File->Back Annotation*** *Dialog Box*

The ASCII back annotation file is typically created as output from another program but can also be manually prepared with a text editor. Each line in the file contains a command that adds or changes an attribute value for a specific instance of a net, symbol, or symbol pin.

The syntax for each line in the attribute update file includes four fields separated by one or more blank characters. The last field, the attribute value, can contain embedded blanks. The four fields are:

Attribute Type
This first field contains a single letter to indicate the type of attribute that is being described by the remainder of the line. The choices are:

I	Symbol attribute, applied to a symbol instance
N	Net attribute, applied to a signal in the design

P	Pin attribute, applied to the instance of a symbol pin

Attribute Name
The name of the attribute is specified in the second field. The current attribute names are listed in the **Attributes->Symbol Attributes**, **Attributes->Pin Attributes**, and **Attributes->Net Attributes** sections in the ECS Preferences Editor. Attribute names must be listed in these sections if they are to be used in the attribute update file.

The attribute numbers can also be used in place of the attribute name.

Instance Name
The third field specifies the name of the specific instance of a net, symbol, or symbol pin that receives the attribute.

The hierarchical instance name of a symbol and of the schematic it represents are equivalent. The name is formed by concatenating the local instance name of the symbol to the hierarchical instance name of the schematic in which the symbol appears. The names are delimited by periods.

A net typically appears in several blocks of a design. When naming a net, the instance name of highest level schematic in which the net appears is used. A dash followed by the local name of the net in that schematic is added to the instance name to form the hierarchical net name.

If the net is a global net or appears in the root schematic, the net name is created by preceding its local net name with a period and dash. For example, net DATA0 referenced in the top level schematic is referred to as *.-DATA0.*

The pin instances are named by appending a dash and the pin name to the symbol instance name in which the pin appears.

Attribute Value
The value of the attribute is contained in the last field of the line. The Hierarchy Navigator does not check this value for correctness as it adds it to the hierarchy.

Example Attribute Update File

In the following example, the root schematic contains symbol instance AA1. The schematic represented by this symbol instance contains symbol instance BB1. A set of entries in the *.atr* file to set the Value attribute of symbol AA1.BB1 and the Load attributes of its input pins follows:

```
I       Value     .AA1.BB1         3456
P       Load      .AA1.BB1-IN1     2.2
P       Load      .AA1.BB1-IN2     2.4
```

The entries to set the capacitance attribute of the global net CLOCK, the net XX in the root schematic and YY in the schematic instance .AA1 follow:

N	Cap	.-CLOCK	0.7
N	Cap	.-XX	0.6
N	Cap	.AA1-YY	0.7

See Also

- pcbback in the *Interfaces* section

File->Create Symbol

Symbols that represent hierarchical blocks are typically drawn as rectangles with the input pins on the left and the output pins on the right. The **File->Create Symbol** command provides a quick and efficient method of creating these generic symbols while running the Schematic Editor.

The symbol graphics for these symbols consists of a rectangle with pin leads extending outward. The input pins are lined up on the left side and the output pins are lined up on the right side. The length of the pin leads is based on the value of the *DefaultPinNameOffset* parameter from the **Controls->System Controls** section of the ECS Preferences Editor. The symbol also contains one attribute window for displaying the name near the top of the symbol and another attribute window for displaying the instance name near the bottom of the symbol. The height and width of the symbol are adjusted based on the number of pins and the length of their names.

When the **File->Create Symbol** command is invoked, a dialog box is presented that allows specification of the symbol name, input pins, output pins, and bidirectional pins. The names for the pins must be separated by commas.

For the special case of creating a symbol that represents the current schematic, a button labelled *This Block* is provided. This button is especially useful when the input nets and output nets are already labelled with I/O markers. Pushing the *This Block* button causes the system to fill in the edit fields with all of the necessary information. The following fields are filled with the indicated information.

Name	The name of the schematic
Inputs	The names of all the nets marked as inputs with an I/O marker
Outputs	The names of all the nets marked as outputs with an I/O marker
BiDirs	The names of all the nets marked as bidirectional with an I/O marker

This information is used to name the block itself and its pins. A bus pin name must be enclosed between equal signs. If a compound name appears in the list of pins and it is not surrounded by equal signs, it is expanded to produce individual pins for each name. If the name is enclosed in equal signs, it produces a bus pin. The following examples illustrate three possible ways to enter pin names into the *Inputs* field.

A,B[0;3],C	creates 6 pins: A B[0] B[1] B[2] B[3] C
A,=B[0;3]=,C	creates 3 pins: A B[0;3] C
=A,B[0;3],C=	creates 1 pin: A,B[0;3],C

Once all the information has been entered into the edit fields, the symbol can be placed or edited. To edit the symbol with the Symbol Editor, push the *Edit* button. To place the symbol without editing, push the *Run* button. This builds the symbol defined and invokes the **Add->Symbol** command.

File->Dump Image

Sym, Sch, Nav

Dump Image allows you to copy a rectangular portion of a schematic or symbol to the clipboard. You can then paste the contents of the clipboard into another Macintosh application. This is useful for documenting a design using a word processor.

The following procedure describes the **File->Dump Image** command.

1. Choose the **File->Dump Image** command. You are prompted to select the first corner of a window.
2. Click and move the mouse to create a rubber band box around the area to be copied to the clipboard. A second click defines the area to be copied.

   You can also perform the selection by clicking and then dragging the mouse to form the rectangular area. A rubber band box follows the mouse. Releasing the mouse causes the area inside the box to be copied to the clipboard.

See Also

- **File->Print Window**, which prints a rectangular area of a drawing

File->Edit Schematic

Sym, Nav

The **File->Edit Schematic** command provides a convenient way to invoke the Schematic Editor while you are working in the Hierarchy Navigator. When spawned from the Symbol Editor, **File->Edit Schematic** provides a convenient way to edit the schematic for the symbol being edited.

The **File->Edit Schematic** command immediately spawns a new task with the Schematic Editor and the schematic that is currently in the Navigator's window. The new task is independent of the Navigator. The Navigator is still available and can be activated without terminating the editor task.

If you use the **File->Edit Schematic** command from the Navigator and make changes to a schematic, you can update the Navigator with the new changes by executing the **File->Restart** command from the Navigator.

File->Edit Symbol

Sch, Nav

The **File->Edit Symbol** command provides a convenient way to invoke the Symbol Editor while you are working in the Hierarchy Navigator or Schematic Editor.

The **File->Edit Symbol** command requires that you to select a symbol to edit. Place the cursor within the area of the symbol and click the mouse. A new task is spawned with the Symbol Editor and the selected symbol. The new task is independent of the Navigator.

The **Edit->Symbol** command is invoked using the following procedure.

1. Choose the **File->Edit Symbol** command from the menu. You are prompted to select a symbol.
2. Click the mouse inside the target symbol. The Symbol Editor is invoked on the target symbol.

If you use the **File->Edit Symbol** command from the Navigator and make changes to a symbol, you can update the Navigator with the new changes by executing the **File->Restart** command from the Navigator.

File->Expand Symbol

The **File->Expand Symbol** command increases the size of the work area provided for the symbol. The additional area is added to the right and bottom sides of the symbol. If more space is needed on the left or top of the drawing, use the **Edit->Move** command to shift the entire symbol.

The initial work area created by the Symbol Editor is 40 x 40 grids. After the initial creation, the Symbol Editor provides a work area that is approximately twice the height and width of an existing symbol.

When the size of a symbol increases so that it no longer fits in the area provided, **Expand Symbol** is used to increase the size of the work area. The **File->Expand Symbol** command increases the initial size to 80 x 80 grids. Subsequent applications of **File->Expand Symbol** increase the size in increments of 20 grids up to a maximum of 400 x 400 grids.

File->New ▯N Sym, Sch

File->New is available in the Schematic and Symbol Editors to start a new drawing without exiting from the editor.

File->New checks the current drawing for modifications made since the file was last saved. If the drawing was modified, the command offers three options.

Save	Checks for an existing file. This file can be the original file or one created during a previous **File->Save**. If the file exists, the **File->Save** is performed. If no file had been specified (the drawing was just created), the command presents a dialog box with an edit field for entering the name of the file. The proper file extension of *.sym* for the Symbol Editor or *.sch* for the Schematic Editor is automatically supplied. If you try to specify a file extension, it is ignored. The command then saves the drawing data in the requested file. **File->New** then clears the current drawing from the window and starts a new schematic or symbol drawing. The new drawing is untitled until the first **File->Save**.
Discard	Discards any changes made to the current drawing, clears the current drawing from the window, and starts a new schematic or symbol drawing. The new drawing is untitled until the first **Save**.
Cancel	Returns to the current drawing.

File->Open ⌘O Sym, Sch

The **File->Open** command allows you to change the drawing without exiting from the Symbol and Schematic Editors.

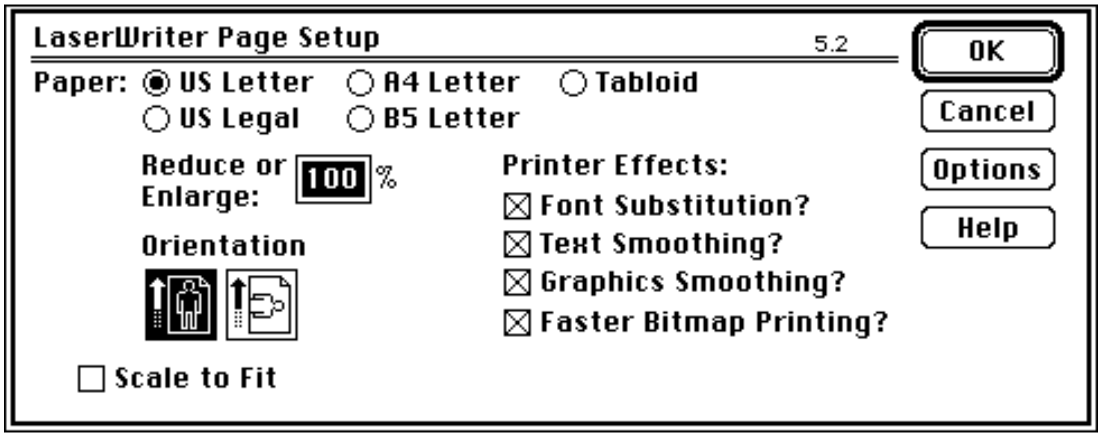
File->Open checks the current drawing for modifications that were made since the file was last saved. If the drawing was modified, the command offers three options:

Save	Checks for an existing file. This file can be the original file or one can be created during a previous **File->Save**. If the file exists, the **File->Save** is performed. If no file had been specified (the drawing was just created), the command presents a dialog box with an edit field for entering the name of the file. The proper file extension of *.sym* for the Symbol Editor and *.sch* for the Schematic Editor is automatically supplied. If you try to specify a file extension, it is ignored. The command then saves the drawing data in the requested file.
Discard	Discards any changes made to the current drawing since the last **File->Save**.
Cancel	Aborts the **File->Open** command and returns to the current drawing.

Once the current file has been saved or discarded, the **File->Open** command presents a dialog box containing a list of the drawings in the current directory. This list only includes those drawings that are the correct type for the editor being used. Select a new drawing from the list and press the *Open* button.

File->Page Setup

The **File->Page Setup** command invokes the dialog box shown in Figure 5-16. The **File->Page Setup** command allows you to change many of the characteristics of the printed page.



LaserWriter Page Setup 5.2

Paper: [✓ US Letter] A4 Letter Tabloid US Legal B5 Letter

Reduce or Enlarge: 100 %

Orientation

Printer Effects:
☒ Font Substitution?
☒ Text Smoothing?
☒ Graphics Smoothing?
☒ Faster Bitmap Printing?

☐ Scale to Fit

Figure 5-16 Page Setup Dialog Box

The various features in the **Page Setup** dialog box are described below.

Paper
Select the size of paper in your LaserWriter cassette tray.

Reduce or Enlarge
Scales the printed page by the amount selected. This is measured relative to the nominal size determined by the grid size and grid units defined using the ECS Preferences Editor.

Orientation
The choices are landscape (longest edge forms bottom of page) and portrait. Usually schematics are drawn in landscape mode.

Scale to Fit
This scales the complete drawing to fit one printed page.

Font Substitution
If selected, the LaserWriter substitutes Times for New York, Helvetica for Geneva, and Courier for Monaco.

Text Smoothing and Graphics Smoothing
This is a fine-tuning feature that makes some text and/or graphic bitmaps look better.

Faster Bitmap Printing
Text and graphic bitmaps print faster with this option selected.

Precision Bitmap Alignment
This option eliminates the minor distortion you may find in printed graphic bitmaps. It also prints faster. However, the entire page is reduced by 4%.

Larger Print Area
This option uses the extra memory in the LaserWriter to increase the print area instead of storing more fonts.

Unlimited Downloadable Fonts
This option allows an unlimited number of fonts in a document. The document will take longer to print. Because you can only have a single font in a schematic or symbol drawing, this option is not typically useful for ECS.

See Also

- **File->Print Window** to print a portion of the current drawing
- **File->Print** to print the current page

File->Print

⌘P Sym, Sch, Nav

File->Print produces a hardcopy of a schematic or symbol.

In the Symbol Editor, the command immediately begins to output a hardcopy of the symbol currently being edited.

In the Schematic Editor or Hierarchy Navigator, the command checks for multiple sheets. If there is only one sheet defined, the command proceeds to print that sheet immediately. If the schematic has more than one sheet, a sheet selector dialog box is presented. The sheet is selected by clicking the mouse on the corresponding line and then clicking the *Print* button. There is also an *All* button that causes all sheets to be printed.

See Also

- **File->Print Window** to print a portion of the current drawing
- **File->Page Setup** to configure the printed page

File->Print Window

Sch, Nav

File->Print Window prints a rectangular portion of the drawing you are currently viewing. **File->Print Window** is found in the Schematic Editor and Navigator.

This command prompts you to select a window corner. Press the mouse at one corner of the rectangular area you wish to have printed. Drag the cursor to the opposite corner and release. A rubber band box indicates the area to be printed. This rectangular area is sent to the printer as soon as you release the mouse.

See Also

- **File->Print**, which prints the current page
- **File->Page Setup**

File->Quit ⌘Q Sym, Sch, Nav

File->Quit terminates the execution of the current application program.

File->Quit checks the current drawing for modifications that were made since the file was last saved. If the drawing was modified, the command offers three options.

Save	Checks for an existing file. This file can be the original file or one created during a previous **Save**. If the file exists, the **Save** is performed. If no file had been specified (the drawing was just created), the command presents a dialog box with an edit field for entering the name of the file. The proper file extension of *.sym* for the Symbol Editor or *.sch* for the Schematic Editor is automatically supplied. If you try to specify a file extension, it is ignored. The command then saves the drawing data in the requested file and exits the application.
Discard	Restores the drawing to the state before the editing session and exits from the application.
Cancel	Returns to the application program.

Once the file has been saved or discarded, the command closes all of the application's windows and terminates the application.

File->Restart

File->Restart causes the current design tree loaded in the Hierarchy Navigator to be rebuilt. This is useful if you have modified any of the underlying schematics or symbols of a design while the design is loaded in the Navigator. Rebuilding the design causes the new symbols and new schematics to be incorporated into the Hierarchy Navigator's design database.

If you did not use the **File->Restart** command after modifying symbols or schematics in a design, any netlists or other output generated would not reflect your most recent changes.

Another case when the **File->Restart** command is used is to incorporate back-annotation data into a design. If you had a design loaded in the Navigator and either modified a back annotation file or created a completely new back-annotation file, you could incorporate the new effects by executing the **File->Restart** command. See the *Interfaces* section for more information on back annotation.

File->Revert

File->Revert causes the file currently being edited to revert to the same state it was in when the file was last saved. This has the effect of forgetting all the changes made to the file since it was last saved.

You can use this command if the changes to the current drawing are not working out and you wish to return to a known state. You can also use the **Edit->Undo** command to get back to a know state, but the **File->Revert** command is faster if major changes have taken place.

See Also

- **Edit->Undo**, which restores the drawing to its previous state before the last command

File->Save

⌘S Sym, Sch, Nav

File->Save is used to save the current results of a work session to an existing file. In the case of the Symbol and Schematic Editors, the results saved consist of any wires, symbols, attributes, or graphic material added to the drawing. In the case of the Hierarchy Navigator, they consist of the context and assigned attributes of the navigation.

Schematic and Symbol Editors
File->Save checks for an existing file. This file can be the original file or can be created during a previous **Save**. If the file exists, the **Save** is performed. If no file had been specified (this is the first save for a new drawing), the command performs like the **File->Save As** command.

Hierarchy Navigator
File->Save in the Hierarchy Editor saves the display context for the schematics that are being viewed. This includes things like nets that are marked, viewing a certain level of hierarchy, or being Zoomed In to a certain place on the drawing. It also saves any attributes that have been added to the hierarchy since the beginning of the work session.

When the Hierarchy Navigator is started, it requires the name of the file containing the root (top-level) schematic. The **File->Save** command derives the saved file name by replacing the *.sch* extension with an extension of *.tre*. Subsequent **Save** commands store the new changes to the *.tre* file.

The Schematic Editor automatically deletes any blank sheets when a **File->Save** or **File->Quit** is performed.

See Also

- **File->Save As**, which saves a design to a differently named file

File->Save As

Sym, Sch

File->Save As is used to save the current results of a work session to a new file name. The results saved consist of any wires, symbols, attributes, or graphic material added to the schematic or symbol drawings.

File->Save As presents a dialog box with an edit field for entering the name of a file. The proper file extension of *.sym* for the Symbol Editor or *.sch* for the Schematic Editor is automatically supplied. If you try to specify a file extension, it is ignored. The command then saves the drawing data in the requested file.

The Schematic Editor automatically deletes any blank sheets when a **File->Save** or **File->Quit** is performed.

See Also

- **File->Save**, which saves a design to the currently named file

## File->Sheets 					Y 					Sch, Nav

The Schematic Editor is capable of creating multiple sheet drawings. It is also able to display several views of the same sheet. The **File->Sheets** command provides the needed control for these features. The command operates slightly differently in the Schematic Editor and Hierarchy Navigator.

Schematic Editor

When the Sheet command is invoked, it presents a dialog box containing a list of the sheets already defined, an *Open* button, a *Replace* button, a *Modify* button, and a *Cancel* button. See Figure 5-17.

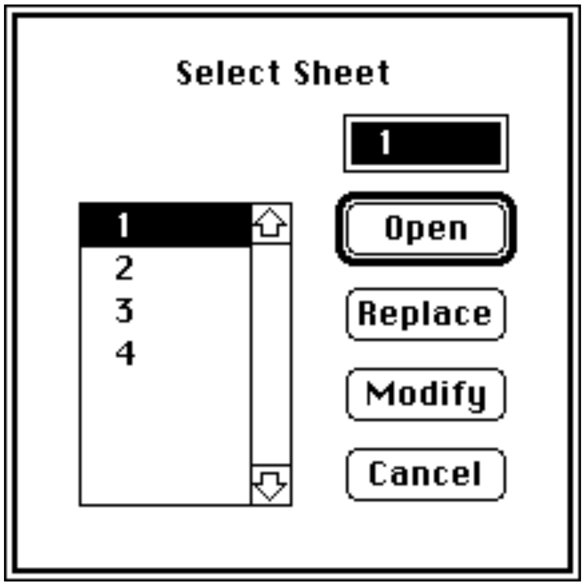


Figure 5-17 Select Sheet Dialog Box

The buttons provide the following functions:

Open	Opens the sheet that is currently selected in the edit box. The sheet being edited previously is also open.
Replace	Closes the sheet you were previously editing and then opens the sheet specified in the list box.
Modify	Allows you to modify the sheet size and sheet number. Pressing the *Modify* button displays a list box as shown in Figure 5-17 with the available sheet sizes and an edit field for changing the sheet number. To change the sheet size, select a new size from the list box and click on the *OK* button. To change the sheet number, type the new number in the edit field and click on the *OK* button. **Undo** returns the sheet to the previous state and truncates the **Undo** buffer. **Redo** is not available after changing sheet size.

Cancel Exits the current task of looking at sheets and returns to the drawing you were editing previously.

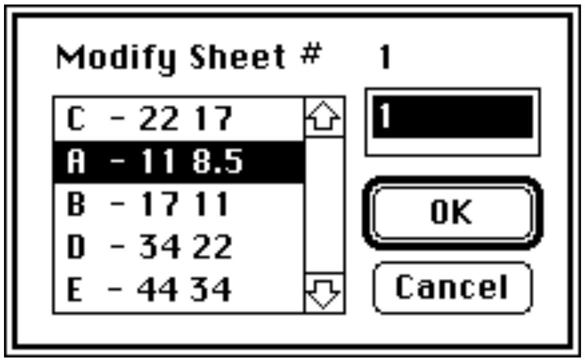


Figure 5-18 Modify Sheet Dialog Box

A sheet can be selected by moving the cursor to the sheet number in the list box and clicking the mouse. A new sheet can be started by typing the new sheet number at the keyboard and pressing the *Return* key. A maximum of eight windows can be open at a time, including up to three copies of the same sheet. Having multiple copies of the same sheet allows both detailed and global views at the same time.

When the desired sheet has been selected, push the *Open* or *Replace* button to activate the command.

If the designated sheet does not exist, the command creates it using the default sheet size.

The Schematic Editor automatically deletes any blank sheets when a **File->Save** or **File->Quit** is performed. To delete a sheet from your data file, use the **Edit->Clear** command to clear a sheet and then perform a **File->Save**.

Hierarchy Navigator

The Hierarchy Navigator can view the sheets of a multiple sheet schematic. The sheet command in the Hierarchy Navigator is similar to the command in the Schematic Editor except it cannot create new sheets.

General Operation

The command creates a new window for the sheet on the display. Each successive window or sheet examined is opened on top of the previous with enough offset so that some of the previous sheet can be seen. This allows you to move between sheets by bringing the required sheet to the front.

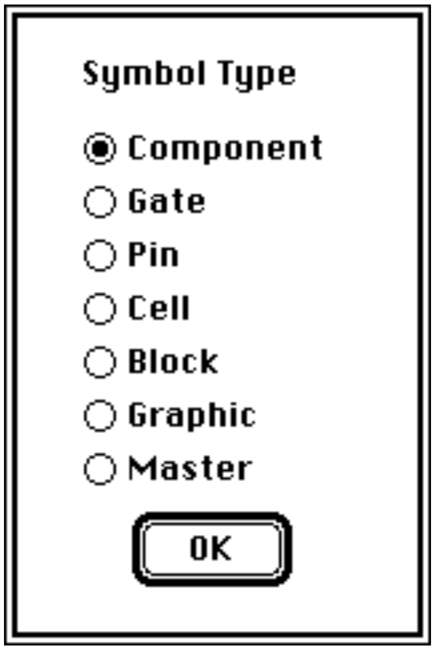
Windows can be resized and repositioned as needed. A window that is no longer needed can be closed.

Up to three views of a sheet can be opened. This permits working with a combination of an overview and two detail views of the drawing. A total of eight windows can be open at one time.

Misc->Change Symbol Type

This command allows you to change the symbol type from within the Symbol Editor.

A dialog box as shown in Figure 5-19 is displayed when you select this command. Choose the button corersponding to the desired symbol type and click on the OK button. This immediately changes the symbol type to thaat selected. You cannot **Undo** this command. You must apply the command a second time to change the symbol back to the original type.


Symbol Type

[✓ Component]
Gate
Pin
Cell
Block
Graphic
Master

Figure 5-19 Dialog Box Displayed for ***Misc->Change Symbol Type***

Misc->Display Options

Sch, Nav

This command is used to set several of the display switches and choices in the Schematic Editor and Hierarchy Navigator. Unless otherwise noted, the options are effective in both the Schematic Editor and the Navigator. The next time a schematic or design is invoked, the default values from the ECS Preferences Editor are used.

When the **Misc->Display Options** command is invoked, a dialog box is displayed that offers control over the following options.

Connect Dots	Three wires connected to a symbol pin or the junction of four wires are always drawn with a connect dot. This control determines whether the connect dot should also be displayed when two wires are connected to a symbol pin or at the junction of three wires.
Border	The sheet border and the zones can be toggled on and off with this control.
Symbol Pins	Unconnected pins can be drawn with an error dot to highlight them. This control turns these dots on and off.
Pin Attributes	The display of pin numbers or names can be turned off to decrease the repaint time and to reduce the clutter on the screen.
Symbol Text	The display of fixed text in a symbol can be turned on and off to decrease repaint time and reduce screen clutter.
Symbol Attributes	The display of text for symbol attribute values can be turned on and off with this control.
Open Ends	Wires not terminating on a net name flag, symbol pin, or another wire are considered errors in a schematic drawing. This control turns on an error dot at the end of such wires. It also places a dot on any net name flag that is not connected to a wire.
Off Page Connects	When creating multiple sheet schematics, it is often desirable to show references to other sheets with nets that connect across more than one sheet. This control enables the cross-reference display on wire segments that have their name at the end of the wire.
Simulation Values	This refers to the Navigator only. It is possible to display values from a simulation directly on the schematic using the Hierarchy Navigator. This command enables the display of these simulation values on the schematic.
Node Numbers	This places the node numbers used for SPICE beside the node names. This is only available in the Navigator.

Misc->Edit Attributes

Nav, Sch (Add)

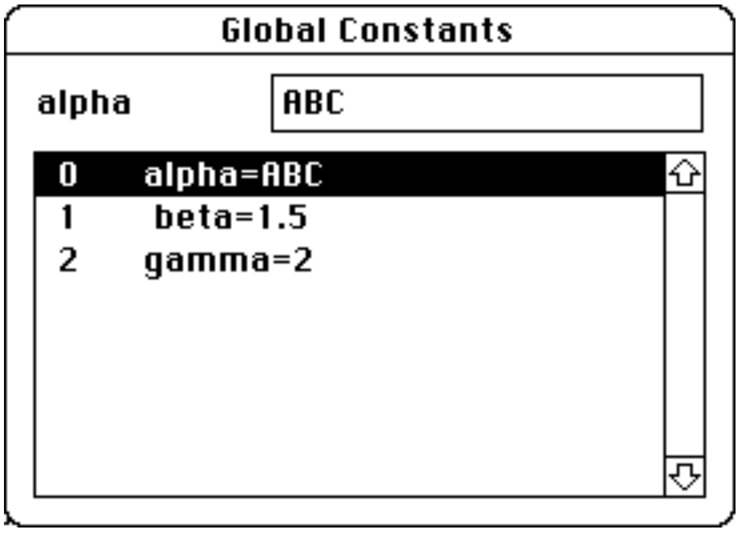
Refer to **Add->Attributes**.

Misc->Edit Constants

The **Misc->Edit Constants** command allows you to change global constants in the Hierarchy Navigator. The changes to the global constants are temporary and last only for the duration of the session.

Use the following procedure to temporarily change the value of a global constant.

1. Select the **Edit->Edit Constants** command from the menu. A list box is displayed as shown in Figure 5-20.
2. Choose the constant you wish to edit from the list. The selected constant is highlighted and its value is displayed on the edit line.
3. Change the value of the constant on the edit line. Type a carriage return to enter the new value.



```
0    alpha=ABC
1     beta=1.5
2    gamma=2
```

Figure 5-20 List Box for Editing Global Constants

Misc->Error Check

The **Misc->Error Check** command of the Schematic and Symbol Editors finds errors in finished schematics and symbols. The error report is displayed in a pop-up text window and written to a file *design_name.err* for later use. This command is useful for detecting the type of errors that are only meaningful when a schematic or symbol is completely specified.

For symbols, the following types of errors are detected.

- Block symbols should have a schematic with the same name present in the current directory.
- Symbols of a type other than Block are primitives and should not have a schematic of the same name.
- Each pin must have a *Name* in a Block or Cell and a *PinNumber* in a Gate or Component.
- Every pin in a Gate must have the same number of sections. This means the *PinNumber* attributes must all have the same number of numbers.
- Pins in Component and Pin symbols can only have one *PinNumber*.
- Pins in the same group of a Gate must all have the same *Polarity*, *Load*, and *Drive*.
- Pins on Blocks should not have *Load* or *Drive* specifications.
- Pins on symbols that are not Blocks should have *Load* or *Drive*. Input pins should have *Load* but not *Drive*. Output pins should have *Drive*, but no *Load* unless the pin is Tri-state, in which case it should have a *Load* that represents the load in the High-Z state. Bidirectional pins should have both *Load* and *Drive*.

For schematics, the following types of errors are detected.

- Bus taps should be named.
- There should not be any isolated net name flags.
- If the *ShowSymbolPins* option is enabled, each pin on the symbol representing this schematic should be connected.
- If the *MarkOpenEnds* option is enabled, there should not be any wire ends that are not connected to something.
- Only ordered buses can be connected to a bus pin on a symbol.
- Only ordered buses can be marked with I/O markers.
- Nets cannot be marked with more than one I/O marker.
- A bus tap and its bus cannot both be marked with an I/O marker.

If there is a symbol for this schematic, the following errors are detected by the schematic checker.

- Each pin should have a corresponding net with an I/O marker whose direction matches the *Polarity* of the pin.
- Each net marked with an I/O marker should correspond to a pin.

The error report from the Schematic Editor lists the sheet number and zone in front of each error. This information can help you find the offending element.

See Also

- *Design Analysis Tools* section for details on further checking available in the Navigator

Misc->Graphic Options ⌘G Sym, Sch

The **Misc->Graphic Options** command sets the parameters controlling the **Draw** commands in the Symbol and Schematic Editors. When the command is invoked, a dialog box is presented that offers control over the following parameters.

Text Font	This control selects the size that is to be used. The available choices are small, medium, and large. They correspond to 5, 7, and 9 fine grid units in height.
Justify	Text can be left-, center-, or right-justified. This parameter applies to fixed graphic text and symbol attribute windows. Text is always vertically centered.
Grid	All electrical elements of the schematics and symbols are drawn on the main working grid as specified in the ECS Preferences Editor. Graphic elements and text can optionally use a finer grid for their locations. This control permits the main working grid to be subdivided by 2 or 4 to provide this higher resolution.
Grid Display	This controls whether or not the primary grid is displayed. If you display the grid, it appears as a dotted line, with one dot at every grid intersection. Every 10th grid point is larger to aid in counting grids. As you zoom out in scale and the grids get closer together, some grid dots are not displayed
Full Cursor	The system has a choice of a small plus cursor (the default) and a full-screen cursor. The full-screen cursor makes it easier to align objects. The *Full Cursor* control selects between the normal cursor and the full-screen cursor.
Wide Lines	There are two line weights available for drawing lines and rectangles in the symbol and schematic. The *Wide Line* control selects between normal and heavy line weights. The heavy lines have the same weight as buses on schematics.
Rotate Text	Text and Attribute windows can be entered as horizontal or vertical text. The *Rotate Text* control selects between the two orientations.

Misc->HiLite

HiLite is used as an aid in tracing nets in a schematic. Highlighting a net causes the net to change color (or become a dashed line on a monochrome monitor). If the net is a signal in a bus, the bus is also highlighted.

The net to be highlighted is selected by placing the cursor on a wire in the net and then clicking the mouse. More than one net can be highlighted at a time.

To remove the highlight from a net, place the cursor on a wire of the highlighted net and click the mouse. The net returns to its original color. If a bus containing this net was highlighted, it too returns to its original color if none of its other signals are still highlighted.

To remove the highlights from all nets in the schematic, press the *Escape* key.

Misc->Mark

Misc->Mark in the Hierarchy Navigator assigns one or more nets or symbols to a special group. All nets in this group are highlighted by being drawn in the Net Highlight color (or dashed lines on monochrome monitors). Symbols in this group are highlighted by being drawn in the HighlightSymbol color (gray on monochrome monitors). Marked symbols are redrawn on top of a colored or shaded background.

This feature is useful for tracing nets through the hierarchy. By marking a net and then pushing into each sub-block, the entire net can be viewed. Buses are considered to be local to the schematic on which they occur. For this reason, a marked bus is only highlighted on the local schematic, not throughout the hierarchy.

The net marking feature can also be used by some of the post-processors to identify special nets and symbols. This feature is used in the Simulator interface to identify the nets to be monitored or graphed.

A net or symbol can be marked by placing the cursor on a wire or symbol and by clicking the mouse. Nets or symbols can also be marked by typing full hierarchical names. All nets and symbols can be unmarked simultaneously by typing *-ALL.*

Typing a question mark (?) brings up a text window with a list of all the marked nets and symbol instances.

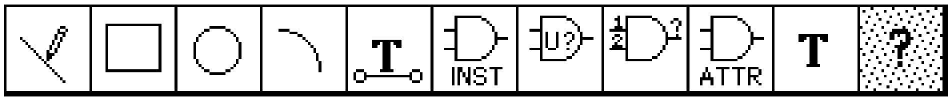
A marked net or symbol can be unmarked by placing the cursor on the marked item and clicking the mouse. Nets cannot be unmarked by name.

Marked nets and symbol instances are saved when the hierarchy is saved.

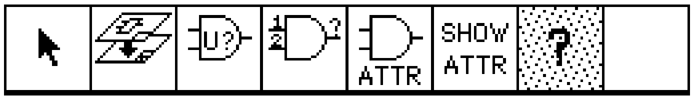
Misc->Pin Numbers

Nav, Sch (Add)

Refer to **Add->Pin Numbers**.



Schematic Editor Tool Palette



Navigator Tool Palette

Misc->Query is used to request additional information about the circuit elements. The information is shown in a text box that pops up when the first element is selected and is updated as each new element is selected.

Circuit elements can be selected by name from the keyboard or by clicking the mouse while the cursor is over the element. There are four types of query selections that can be made. The types, in order of selection priority, along with the information displayed, are pins, nets, symbols, and type.

Pins
Symbol pins can be queried in the Hierarchy Navigator and can only be selected by position. If you select a bus pin, a dialog box allows you to specify which single pin to query. The information displayed for a pin query is:

- pin name
- attached to which instance
- attached to which net
- pin polarity
- fanout
- other pin attributes

Nets
A net can be interrogated by selecting it with the cursor or by entering its name via the keyboard. When entering net names in the Hierarchy Navigator, use the full hierarchical name of the net as it would appear on a net name flag.

When the net that is interrogated is visible, it is highlighted by changing its color (or drawing with dashed lines on monochrome displays). The information displayed is:

- net name
- polarity if input or output node (determined from I/O Marker on net)
- other net attributes
- local net name, applicable only on underlying schematics

- node number in database
- symbol connections

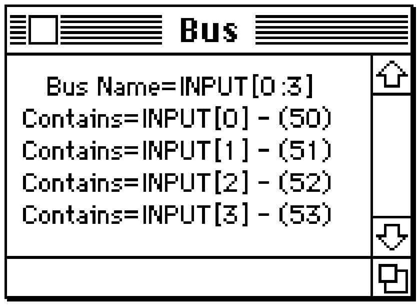
You can click on a particular symbol connection in the list box and the schematic pans to display the selected connection.

Buses
A bus can be interrogated by selecting it with the cursor or by entering its name via the keyboard. When entering bus names in the Hierarchy Navigator, use the full hierarchical name of the bus as it would appear on a net name flag.

When the bus that is interrogated is visible, it is highlighted by changing its color (or drawing with dashed lines on monochrome displays). The information displayed is:

- bus name
- individual nets contained in the bus
- symbol connections (in schematic only)

Figure 5-21 shows the bus list box displayed after querying a bus.



```
  Bus Name=INPUT[0:3]
Contains=INPUT[0] - (50)
Contains=INPUT[1] - (51)
Contains=INPUT[2] - (52)
Contains=INPUT[3] - (53)
```

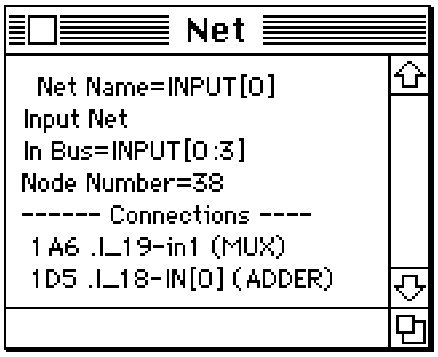
Figure 5-21 Bus List Box Displayed from Bus Query

You can click on individual nets in the bus list box and a net list box as shown in Figure 5-22 is displayed containing all the information for the selected net. The behavior of this second list box is the same as that described for querying nets above with the following exception.

Clicking on the line

In Bus=

causes the net list box to toggle back to the bus list box. This allows you to toggle between the bus list box, showing all the elements contained in a bus and the net list box, which shows more detailed information about one element of the bus.



```
 Net Name=INPUT[0]
Input Net
In Bus=INPUT[0:3]
Node Number=38
------ Connections ----
 1A6 .I_19-in1 (MUX)
 1D5 .I_18-IN[0] (ADDER)
```

Figure 5-22 Net List Box Displayed from Bus Query

Symbols
The instance of a symbol can be queried by selecting it with the cursor or by entering its name via the keyboard. When entering the instance name in the Hierarchy Navigator, use the full hierarchical name of the symbol instance as it would appear in the instance name text window. When querying an iterated instance in the Hierarchy Navigator, a dialog box allows you to choose which instance to query.

To resolve a conflict between an instance name and a net with the same name, the instance name can be preceded with *I=* to force its selection.

The selected symbol is highlighted with a colored (or shaded) background. The field of view is automatically changed to display a symbol which has been queried but which is not currently visible in the window.

The information displayed by the **Query** is:

- the name (for example, NAND2, NOR3) and type (gate, component, block, cell, master, pin) of symbol
- full path showing location of symbol file in the file structure
- instance name
- reference designator
- other symbol attributes
- reference location on the schematic (sheet number, vertical border reference, horizontal border reference, for example, 2A6)
- gate section (A, B, C, etc) on gate symbols
- instance number in the hierarchical database (Navigator only)
- pin/net connections

Type
All instances of a symbol type can be queried by selecting a representative symbol with the cursor or by typing the name of the symbol. To select the symbol type with the cursor, move the cursor to a symbol of the desired type. Then, press the *Shift* key and click the mouse.

All visible instances of the selected symbol type are highlighted with a colored (or shaded) background.

If there is a net or symbol instance with the same name as the symbol type, the conflict can be resolved by prefixing the type name with *T=*.

The information displayed by **Query** in this case is:

- internal net count
- instance names and locations (sheet/vertical zone/horizontal zone)

Reference Designator
A symbol with a specified reference designator can be queried by typing the name of the reference designator at the query prompt. In the case of gate symbols, the first section with a specified reference designator is queried.

The symbol whose reference designator is specified is highlighted with a colored (or shaded) background and a plus cursor is centered over the symbol.

In all cases, the queried element remains highlighted until a new element is selected, until the *Escape* key is pressed, or until a new command is selected. The information display remains visible as long as there is an element selected for interrogation.

For Gate symbols the Query box displays the letter of the gate as *A* for the first gate, *B* for the second, etc.

To find a particular gate, type in the reference designator followed by a forward slash followed by the letter of the gate. For example, type *U1/B* to find the symbol whose reference designator is *U1* and whose gate is the second gate of the package.

Misc->Query in the Hierarchy Navigator allows querying of nodes, instances, or reference designators by name or number.

The Query box remains on the screen when the **Push/Pop** command is used to traverse the Hierarchy.

Misc->Ref Designator

Refer to **Add->Ref Designator**.

Misc->Set Origin

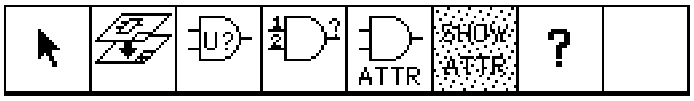
The placement of symbols on a schematic is determined relative to a fixed point on the symbol called the origin. When a symbol is created for the first time, it has no origin. You can assign an origin or change the current origin with the **Misc->Set Origin** command in the Symbol Editor.

When an origin is assigned, its location is marked with long tick marks in the pin color along the border of the symbol window. When a symbol with no origin is saved for the first time, the origin defaults to the upper-left corner of the symbol.

To set the origin, click the mouse over the desired location.

Misc->Show Attributes

Sch, Nav

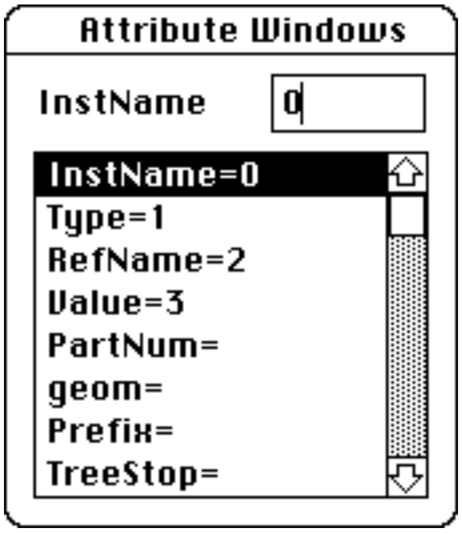


Hierarchy Navigator Tool Palette

Misc->Show Attributes in the Schematic Editor and Hierarchy Navigator can be used to select which attributes are displayed in which attribute windows. This provides a temporary override to the attribute window definitions in the ECS Preferences Editor.

To display a given attribute in a window:

1. Select **Show Attributes** from the **Misc** menu. A dialog box opens as shown in Figure 5-23.
2. Select the attribute from the list by clicking the mouse while the cursor is on the line displaying the desired attribute.
3. Type the window number into the edit field. If more than one attribute is assigned to the same window, the attribute with the lowest attribute number is displayed. Lower attribute numbers are indicated by being closer to the beginning of the list.



Attribute Windows

InstName 0

```
InstName=0
Type=1
RefName=2
Value=3
PartNum=
geom=
Prefix=
TreeStop=
```

Figure 5-23 Attribute Window Dialog Box

See Also

- **Add->Attribute** to set values on attributes in Schematic Editor and Navigator
- *Attribute* section of the manual
- **Add->Show Attributes**, which defines placement of attribute windows on symbol

Misc->Statistics

Sch, Nav

The **Misc->Statistics** command is used to interrogate the Schematic Editor or Hierarchy Navigator about the amount of memory that has been consumed. In each case, the report is given in the pop-up text window. The command gives different displays in the Schematic Editor and Hierarchy Navigator.

Schematic Editor
The report in the Schematic Editor lists each type of element that makes up the schematic database. In most cases, the report includes the number of bytes consumed, the number available, and the percentage of the block that has been consumed.

Attributes are the exception because they are stored as variable-length records. The only meaningful statistic for the attribute block is the percentage of the block that has been consumed.

On a multiple-sheet schematic, there is a data block for the points on each sheet. In this case, there is a report for each sheet's statistics. The sheet numbers are at the extreme right of the line; you may need to expand the text window width to see them.

Hierarchy Navigator
The report for the Hierarchy Navigator lists the number of each of the following found in the design.

Types of symbols	Number of different distinct symbols used in the design. This is equal to the sum of *Primitive cells* and *Hierarchical blocks*.
Primitive cells	Number of different distinct symbols used in the lowest level of hierarchy. Only Gate, Component, and Cell symbols can be counted in this group.
Hierarchical blocks	Number of different distinct Block symbols used in the design.
Instances	Total number of symbol instances found in the design.
Instance pins	Total number of pin instances found in the design.
Primitive instances	Total number of symbol instances found in the lowest level of hierarchy. These symbols are Gate, Component, and Cell instances.
Primitive pins	Total number of pin instances found in the lowest level of hierarchy.
Nets connected	Total number of nets in the design.

This information gives an indication of the size of your design.

This is followed by a list of seven types of records used to store information in the database. Each record type is followed by a measure of the capacity used in this particular design. Three of the record types are fixed length and are reported as number consumed, number available, and percentage of the block that has been consumed.

- instances
- nets
- pins

The remaining four types are attribute records. Because each of these records is a variable length, only the percentage of the memory block that has been consumed is shown.

- definitions and pins
- net attributes
- instance attributes
- pin attributes

Definitions and pins refers to the total number of different distinct symbols and the pins of those symbols.

Processes

The **Processes** menu is available in the Hierarchy Navigator. Selecting an entry under this menu spawns a task associated with the menu entry. The relationship between the menu sub-entry and the process to be spawned upon selecting the menu entry is established in the ECS Preferences Editor. You can add any process or program to the **Processes** menu by editing the **Tools->Navigator Processes** section of the ECS Preferences Editor.

There are several functions in ECS that are provided as auxiliary processes. This method of implementation allows the individual process to be attached as needed. Processes that are never used can be deleted from the system to conserve memory.

The functions that are implemented in this manner have access to, but are prevented from modifying any of the internal data structures of the main modules. Typically, this form of implementation is suitable to post-processors and database verification modules.

Net List Extraction is one class of function that is implemented in this manner. Each net list program produces a data file as its only output. There are several different formats available. Each format is implemented as a separate module. In a typical installation, there are only a few formats that are needed to interface to other CAD tools.

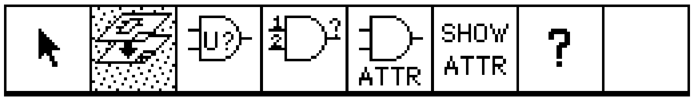
When a command is invoked from the Processes menu, the Navigator passes a copy of the hierarchical database to the selected process. The process then performs its task.

The system is unavailable while the process is executing. Any files that were created by the process are saved on the current directory. In some cases, the process immediately displays its files.

See Also

- *Interfaces* section describing available processes
- **Tools->Processes** in the *System Configuration* section

Push/Pop



Hierarchy Navigator Tool Palette

The **Push/Pop** command is used to traverse the design hierarchy.

Push moves to a lower (more detailed) level in the hierarchy. To push into a symbol, click the mouse within the symbol boundary. If the symbol represents a lower-level schematic, that schematic replaces the schematic currently on the screen.

Pop moves to a higher level in the hierarchy. To pop back to the parent of the current schematic, click the mouse outside of all symbol boundaries. The parent schematic replaces the schematic currently on the screen unless the current schematic is at the highest level (root) of the hierarchy.

Alternatively, you can type the instance name of the destination schematic at the prompt.

The full context of the hierarchy is maintained during the **Push** and **Pop** operations. All net names are shown in their full hierarchical representations. Symbol instance names are shown in an abbreviated format that replaces the leading portion of an instance name with a dot (the instance .AB.CD.EF is displayed as .EF).

Tools->FindItem

Tools->FindItem can be invoked at any time to find the output pin driving the net that is currently queried. This is useful for back-tracing signals during the analysis of simulation results.

The Design Analysis Tools and the Waveform Tool operate from the **Tools** menu of the Hierarchy Navigator. Several of these programs allow interactive identification of nets and symbols on the schematic.

Some of the tools highlight nets or symbols on the schematic. The **Tools->FindItem** command can be used to search through the hierarchy and find a schematic that contains an output pin that is driving the highlighted net.

Tools->ProbeItem

Tools->ProbeItem is used to identify the name of a symbol instance or net to the currently running tool. To probe an item, position the cursor over it and click the mouse. The name of the item is sent to the tool. **Tools->ProbeItem** does not cause the query box to be displayed, but it uses the query box if it is already visible at the time the command is invoked. This allows you to query several nodes without sending the names to the tool and then switch to the **Tools->ProbeItem** command to send more names to the tool.

The Design Analysis Tools and the Waveform Tool operate from the **Tools** menu of the Hierarchy Navigator. Several of these programs allow interactive identification of nets and symbols on the schematic.

Tools->ProbeItem is initially disabled. When a tool is running that allows use of **Tools->ProbeItem**, it is enabled.

View->Full Fit

Sym, Sch, Nav

View->Full Fit changes the magnification of the current drawing to just fill the screen within the magnification limits explained below. The command is executed when it is selected from the menu.

There is a minimum magnification factor where text can be drawn. When the magnification drops below this value, all text strings are replaced by their outline boxes. The minimum magnification scale where text can be displayed is used as the base magnification by ECS. The allowable magnifications in terms of this base scale are 1/8x, 1/4x, 1/2x, 1x, 2x, 3x, and 4x.

View->Redraw

Sym, Sch, Nav

The **View->Redraw** command causes the current window to be repainted. Some operations such as **Edit->Clear** can modify the drawing and not repaint the area properly. In these cases, you can use the **View->Redraw** command to repaint the whole window. The repainting of the window takes place immediately after selecting the command.

View->Reset Views

The **View->Reset Views** command causes the default window size and view to be reset in all schematics across all levels of hierarchy in a design.

View->Window

Sym, Sch, Nav

View->Window increases the current drawing magnification, showing a portion of the current drawing in more detail. When the **View->Window** command is invoked, the cursor changes to a thick plus sign.

There is a minimum magnification factor where text can be drawn. When the magnification drops below this value, all text strings are replaced by their outline boxes. The minimum magnification scale where text can be displayed is used as the base magnification by ECS. The allowable magnifications in terms of this base scale are 1/8x, 1/4x, 1/2x, 1x, 2x, 3x, and 4x.

There are two modes for **View->Window**, a click mode and a drag mode.

Click	Move the cursor to the point on the drawing to be centered in the window. The centering is automatically adjusted when the point is too near an edge of the drawing. When the mouse is clicked, the command repaints the drawing with the next higher magnification and new center point.
Drag	Press the mouse at one corner of a rectangular area you would like to view. Drag the mouse to the diagonally opposite corner of the desired area and release the mouse. ECS centers the rectangular area inside the current window and adjusts the magnification to the largest allowable value that still permits the complete rectangle to be viewed.

View->ZoomIn ⌘K Sym, Sch, Nav

View->ZoomIn increases the current drawing magnification, showing a portion of the current drawing in more detail. The center of the window before the **ZoomIn** stays at the center after the command is executed. The command is executed when it is selected from the menu.

There is a minimum magnification factor where text can be drawn. When the magnification drops below this value, all text strings are replaced by their outline boxes. The minimum magnification scale where text can be displayed is used as the base magnification by ECS. The allowable magnifications in terms of this base scale are 1/8x, 1/4x, 1/2x, 1x, 2x, 3x, and 4x.

View->ZoomOut ▯J Sym, Sch, Nav

View->ZoomOut decreases the current drawing magnification and displays more of the current drawing in less detail.

The center of the window remains constant after the magnification change. The centering is automatically adjusted when the center of the window is too near an edge of the drawing. When the command is selected, the command repaints the drawing with the new magnification.

There is a minimum magnification factor where text can be drawn. When the magnification drops below this value, all text strings are replaced by their outline boxes. The minimum magnification factor where text can be displayed is used as the base magnification by ECS. The allowable magnifications in terms of this base scale are 1/8x, 1/4x, 1/2x, 1x, 2x, 3x, and 4x.

6

Interfaces

This section explains the interfaces the are currently supported for the Engineering Capture System (ECS). This includes netlisters, simulation environments, and generic data transfer interfaces.

The interfaces covered in this section are organized in alphabetical order:

- Archive utility
- ASCII interface
- Back Annotation
- EDIF
- Generic Netlistts
- PCB back annotation interfaces
- PCB netlisters (PADS, RINF, Cadnetix)
- SILOS
- Simulation Environment
- SPICE
- Timemill
- Verilog
- VHDL

Archive

Tools->Archive is a utility that gathers all the symbols and schematics and symbols needed in a design and places them all in a single directory. It is typically used for archiving designs or for transferring designs from one place to another.

Archive is run as a tool from the **Tools** menu of the Hierarchy Navigator and must be added to the tools menu using the ecs.ini Editor. There are currently two options for the command as described below:

archive	creates a list of all files needed for the current design. This list is written to file *design_name.lst*.
archive -save=*path*	saves all symbol and schematic files necessary to build the design. These files are all saved in the folder referenced by *path*. The disk drive name and any folders referenced on the *path* must not contain spaces.

The command currently records symbols of type cell, block, component, gate, and pin. It does not record master or graphic symbols.

ASCII Interface

The ASCII interface converts data stored in the ECS binary database to an ASCII format that you can edit with any text editor. The interface also converts an ASCII file in the appropriate format to the ECS binary database. You can interface to any CAE/CAD system using the ECS ASCII interface.

Once a database is in the ASCII format, you can modify it and then translate it back into the ECS database. This allows various filtering and processing programs to be used in the ECS environment.

There are four ASCII interface programs as shown in Figure ASCII-1. These applications are typically stored in the main ECS folder.

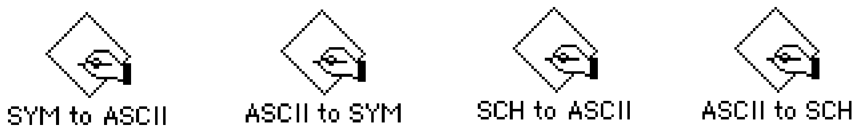


Figure ASCII-1 The ASCII Interface Applications

Operation of ASCII Interfaces

Use the following procedure to invoke one of the ASCII interface programs.

1. Double-click on the desired application in the main ECS folder. This invokes the application and pops up a dialog box as shown in Figure ASCII-2.
2. Navigate the file structure to view the contents of the target folder. Only those files with the appropriate extension are displayed.
   - .sym for *SYM to ASCII*
   - .sch for *SCH to ASCII*
   - .asy for *ASCII to SYM*
   - .asc for *ASCII to SCH*
3. Select a single file to convert by double clicking on the file name. Click the *ALL* button to convert all listed files.

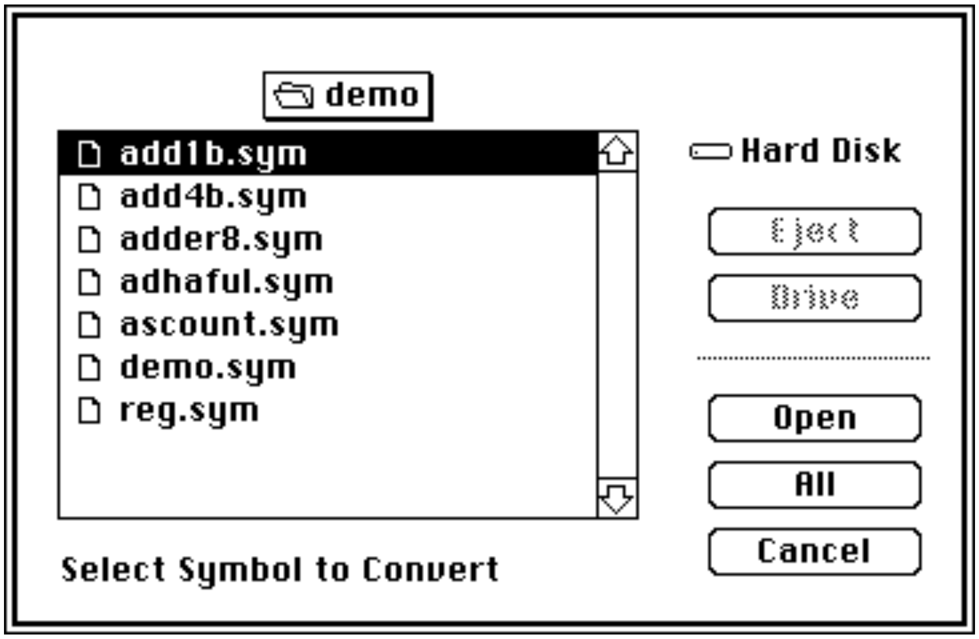


Figure ASCII-2 Dialog Box for SYM to ASCII

ASCII Symbol Interface

An ECS symbol database contains the information required to describe a symbol and all of its attributes. The database entries include graphic elements, fixed text, symbol attributes, attribute display windows, pins, and pin attributes.

The system calculates an extent box for each symbol. This box is a rectangle that encloses all of the points that are used to define the various elements of the symbol. The upper left corner of this box is used as the default origin for the symbol.

The ECS uses a grid system for all coordinates. The size of the grid is defined by the *Grid* and *Units* parameters in the **Controls->Sheet Layout** section of the ECS Preferences Editor. This grid is divided into a secondary grid which has four points in each major grid interval.

The secondary grid is used for all coordinates in the symbol database. When the Symbol Editor stores the symbol data base it recalculates the extent box and adjusts the coordinates so that all coordinates are positive integers. The Symbol Editor also limits the size of a symbol to 400 major (or 1600 secondary) grids in each dimension.

The interface uses a line oriented ASCII text file format. Each element in the symbol is represented as a single line in the text file. Each line consists of:

- A keyword begins the line and identifies the type of element described.
- A list of parameters describing the element follows the keyword.

The following conventions apply within each line:

- The parameters are delimited by white space.

- Text strings in lines describing attributes or fixed text appear at the end of the line and can contain embedded blanks.
- Character case is not significant in any keyword.
- Coordinate locations are given as pairs of numbers. Increasing x coordinates are to the right and increasing y coordinates are toward the bottom of the symbol. All coordinate locations and distances are given in units of the secondary grid.

In the following description of the format, keywords are shown in bold type and parameters are shown in italics.

Version *number*

The ECS symbol database version number is specified on the first line of the file.

SymbolType *type*

The SymbolType should be the second line of the file. The *type* parameter is one of: COMPONENT, GATE, CELL, BLOCK, PIN, MASTER, or GRAPHIC. If this parameter is not specified, the **SymbolType** defaults to BLOCK.

Each of the following formats used for graphic elements contains a parameter that specifies the weight of the lines used to draw the element. This parameter can have the value of NORMAL or WIDE. These keywords can be abbreviated as N or W.

Each of the following formats used for graphic elements contains a parameter that specifies the style of the lines used to draw the element. This parameter can have the value of Dash, Dot, DashDot or if the parameter is not present at all, the line is drawn solid. This style is only implemented on UNIX workstations.

Line *width x1 y1 x2 y2 style*

The coordinates are the line's end points.

Rectangle *width x1 y1 x2 y2 style*

The coordinates are two opposite corners of the rectangle.

Circle *width x1 y1 x2 y2 style*

The coordinates are the opposite corners of a square that encloses the circle. If a rectangle is specified, it is converted to a square with the same upper left corner as the rectangle. The sides of the square are the length of the shorter sides of the rectangle.

Arc *width x1 y1 x2 y2 x3 y3 x4 y4 style*

The first two coordinates describe a circle (see above) containing the arc. The third and fourth coordinates are the starting and ending points of the arc. These points are projected onto the circle if they do not lie on it. The arc is drawn counterclockwise from start to end point.

Text *x1 y1 justify font string*

The coordinate is the origin point for drawing the text.

Justify specifies the horizontal (vertical) justification of the text to the point. *Justify* can have values of LEFT, RIGHT, CENTER, VLEFT, VRIGHT or VCENTER. The first two characters can be used as an abbreviation. Horizontal

(Vertical) text is always justified vertically (horizontally) to the center of the character.

Font specifies the size of the text and can be 0, 1, or 2. The resulting character size is five, seven, or nine secondary grid units in height.

String is the character string that is drawn. The string consists of all characters between the first non-blank character following the *font* parameter and the end of the line.

SymAttr *name string*

Name specifies the attribute and must match one of the attribute names listed in the **Attributes->Symbol Attributes** section of the ECS Preferences Editor.

String is the attribute value assigned including any attribute modifiers.

Window *number x1 y1 justify font*

Number specifies which attribute window is to be drawn.

The coordinate is the origin point for drawing the attribute value.

Justify specifies the horizontal (vertical) justification of the text to the point. *Justify* can have values of LEFT, RIGHT, CENTER, VLEFT, VRIGHT or VCENTER. The first two characters can be used as an abbreviation. Horizontal (Vertical) text is always justified vertically (horizontally) to the center of the character.

Font specifies the size of the text and can be 0, 1, or 2. The resulting character size is five, seven, or nine secondary grid units in height.

Pin *x1 y1 justify offset*

The coordinate is the location of the pin. This location must be on a major grid.

The *justify* and *offset* parameters indicate where to draw the name of the pin. *Offset* is the distance from the pin and can range from 0 to 31 secondary grids. *Justify* indicates the direction the pin name is offset from the pin and can be LEFT, RIGHT, BOTTOM, TOP, VLEFT, VRIGHT, VBOTTOM, VTOP, or NONE. The first two characters can be used as an abbreviation. If the first character is V the text appears vertically.

PinAttr *name string*

The pin attributes apply to the previously defined pin. *Name* specifies the attribute and must match one of the attribute names listed in the **Attributes->Pin Attributes** section of the ECS Preferences Editor.

String is the attribute value assigned including any attribute modifiers.

Example ASCII Symbol File

The following example is the result of translating the symbol in Figure ASCII-3 to the ASCII format.

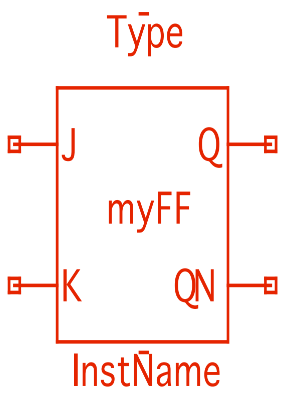


Figure ASCII-3 JK Flip-Flop

```
Version 3
SymbolType COMPONENT
Line N  8 8  0 8
Line N  8 28  0 28
Rectangle N  8 0  40 36
Line N  40 8  48 8
Line N  40 28  48 28
Text 25 17  CENTER 0 myFF
SymAttr SilosModel .JKFF
Pin 0 8  LEFT 9
PinAttr PinName J
PinAttr Polarity IN
Pin 0 28  LEFT 9
PinAttr PinName K
PinAttr Polarity IN
Pin 48 28  RIGHT 9
PinAttr PinName QN
PinAttr Polarity OUT
Pin 48 8  RIGHT 9
PinAttr PinName Q
PinAttr Polarity OUT
Window 0  24 44  CENTER 0
Window 1  24 -8  CENTER 0
```

ASCII Schematic Interface

The schematic database is a compact binary file that is created and modified by the Schematic Editor. The ASCII Schematic Interface allows you to translate the ECS schematic database to and from a format that can be modified by a text editor or other program. This allows conversion of schematics from one system to another.

An ECS schematic database contains the information required to describe a schematic including all of its sheets. The database entries include:

- Wires
- Buses
- Net name flags
- Net attributes
- Bus taps
- Symbol instances (both primitive and hierarchical)
- Symbol attribute overrides
- Attribute window overrides
- Pin attribute overrides
- Cosmetic graphics
- Fixed text
- Tables
- IO Markers
- Table data

A single schematic file has one or more sheets, each representing a separate page of the printed schematic. All of the schematic entities reside on sheets. Each sheet is divided up into grids.

The size of the grid is defined by the *Grid* and *Units* parameters in the **Controls->Sheet Layout** section of the ECS Preferences Editor. The grid is divided into a secondary grid that has four points in each grid interval. All of the connectivity elements (wires, symbols, pins, and attributes) are constrained to lie on the major grid. The cosmetic graphics and fixed text are constrained to lie on the secondary grid.

The secondary grid is used for all coordinates in the schematic database. However, all entities except cosmetic graphics and fixed text have coordinates that are multiples of four.

The size of a sheet in a schematic is determined when the sheet is created. For example, if the width of the sheet is 22 with *Units*=Inches and *Grid*=.1 the sheet is:

22 inches / (.1 inches per grid) = 220 major grids wide

The width of the sheet in the example is 880 secondary grids.

The interface uses a line oriented ASCII text file format. Each element in the schematic is represented as a single line in the text file. Each line consists of:

- A keyword begins the line and identifies the type of element described.
- A list of parameters describing the element follows the keyword.

The following conventions apply within each line:

- The parameters are delimited by white space.
- Text strings in lines describing attributes or fixed text appear at the end of the line and can contain embedded blanks.
- Character case is not significant in any keyword.
- Coordinate locations are given as pairs of numbers. Increasing x coordinates are to the right and increasing y coordinates are toward the bottom of the symbol. All coordinate locations and distances are given in units of the secondary grid.

In the following description of the ASCII format, keywords are shown in bold type and parameters are shown in italics.

Version *number*

The ECS symbol database version number is specified on the first line of the file.

Sheet *number width height*

The sheet entry should appear as the second line of the file. *Number* is the sheet number and can range from 1 to 99. The *width* and *height* represent the coordinate space for the following sheet. All other lines following the sheet represent entities which appear on that sheet. Additional sheet entries represent the beginning of new sheets.

Wire *x1 y1 x2 y2*

The coordinates are the wire's end points. These coordinates must lie on a major grid. These coordinates must also satisfy the interconnection constraints for wires. For example, they must be lines at 45 or 90 degree angles to the edge of the sheet.

Flag *x1 y1 name*

The coordinate is the location of the flag. This coordinate must lie on a major grid.

Name specifies the net name to be displayed. A wire cannot display two different names. The name contains all characters between the first non-blank character after the *y1* parameter and the end of the line. The name must conform to the rules for net names. Spaces are not significant in net names.

IOPin *x1 y1 in_out*

The coordinate is the location of the pin. This coordinate must lie on a major grid. *In_out* specifies the polarity and can have a value of: IN, OUT, or BIDIR. The first character can be used as an abbreviation.

BusTap *x1 y1 x2 y2*

The coordinates are the end points of the wire forming a bus tap. These coordinates must lie on a major grid. The bus tap must be at a 90 degree angle to the bus and can be either horizontal or vertical. It must also satisfy the interconnection constraints.

Symbol *name x1 y1 rot_mir*

The *name* parameter is the name of the symbol. Because this is the name of a symbol database file, it must conform to the file name conventions for ECS.

The coordinate is the location of the symbol origin.

Rot_mir specifies the rotation and mirror operation applied to the symbol definition for this symbol instance. *Rot_mir* can have values of R0, R90, R180, R270, M0, M90, M180, or M270. The first two characters can be used as an abbreviation.

SymAttr *name string*

The symbol instance attributes apply to the previously defined symbol instance.

Name specifies the attribute and must match one of the attribute names listed in the **Attributes->Symbol Attributes** section of the ECS Preferences Editor.

String is the attribute value assigned. The attribute value contains all characters between the first non-blank character following the *name* field and the end of the line.

Window *number x1 y1 justify font*

The attribute window location applies to the previously defined symbol instance. It is an override of the window location defined in the symbol definition.

Number specifies which attribute window is to be drawn.

The coordinate is the origin point for drawing the attribute value.

Justify specifies the horizontal (vertical) justification of the text to the point. *Justify* can have values of LEFT, RIGHT, CENTER, VLEFT, VRIGHT or VCENTER. The first two characters can be used as an abbreviation. Horizontal (Vertical) text is always justified vertically (horizontally) to the center of the character.

Font specifies the size of the text and can be 0, 1, or 2. The resulting character size is five, seven, or nine secondary grid units in height.

PinAttr *x1 y1 name string*

The coordinate is the location of the pin. The specified pin attribute applies to the pin at this location.

Name specifies the attribute and must match one of the attribute names listed in the **Attributes->Pin Attributes** section of the ECS Preferences Editor.

String is the attribute value to be assigned. The attribute value contains all characters between the first non-blank character following the *name* field and the end of the line.

NetAttr *x1 y1 name string*

The coordinate is the location of a point on the net. The specified net attribute applies to the net at this location.

Name specifies the attribute and must match one of the attribute names listed in the **Attributes->Net Attributes** section of the ECS Preferences Editor.

String is the attribute value to be assigned. The attribute value contains all characters between the first non-blank character following the *name* field and the end of the line.

Each of the following formats used for graphic elements contains a parameter that specifies the weight of the lines used to draw the element. This parameter can have the value NORMAL or WIDE. These keywords can be abbreviated as N or W.

Each of the following formats used for graphic elements contains a parameter that specifies the style of the lines used to draw the element. This parameter can have the value of Dash, Dot, DashDot or if the parameter is not present at all, the line is drawn solid. This style is only implemented on UNIX workstations.

Line *width x1 y1 x2 y2 style*
The coordinates are the line's end points.

Rectangle *width x1 y1 x2 y2 style*
The coordinates are two opposite corners of the rectangle.

Circle *width x1 y1 x2 y2 style*
The coordinates are the opposite corners of a square that encloses the circle. If a rectangle is specified, it is converted to a square with the same upper left corner as the rectangle. The sides of the square are the length of the shorter sides of the rectangle.

Arc *width x1 y1 x2 y2 x3 y3 x4 y4 style*
The first two coordinates describe a circle (see above) containing the arc. The third and fourth coordinates are the starting and ending points of the arc. These points are projected onto the circle if they do not lie on it. The arc is drawn counterclockwise from start to end point.

Text *x1 y1 justify font string*
The coordinate is the origin point for drawing the text.

Justify specifies the horizontal (vertical) justification of the text to the point. Justify can have values of LEFT, RIGHT, CENTER, VLEFT, VRIGHT or VCENTER. The first two characters can be used as an abbreviation. Horizontal (Vertical) text is always justified vertically (horizontally) to the center of the character.

Font specifies the size of the text and can be 0, 1, or 2. This corresponds to five, seven, or nine secondary grid units in height.

String is the character string to be drawn. The string contains all characters between the first non-blank character following the font parameter and the end of the line.

Table *x0 y0 rows cols row_height col_width first_row_height first_col_width name_just name_font title_just title_font upper_left_just upper_left_font first_row_just first_row_font first_col_just first_col_font rest_of_table_just rest_of_table_font*
The justification referred to below can take on values of one of the following keywords: BottomLeft, BottomCenter, BottomRight, CenterLeft, CenterCenter, CenterRight, TopLeft, TopCenter, TopRight and None. The *None* option is used if you do not want the title or name the appear on the table. Name and title are drawn outside the table so the *Bottom* options when used to justify the table's name or title prints the table's name above the table. The *Top* options prints the table's name or title below the table.

The font parameters referred to below specifies the size of the text and can take on values of 0, 1, or 2. This corresponds to five, seven, or nine secondary grid units in height.

The coordinate is the upper left hand corner of the table.

rows	specifies the number of rows in the table
cols	specifies the number of columns in the table
row_height	specifies the height in secondary grid units of all rows except the first
col_width	specifies the width in secondary grid units of all columns except the first
first_row_height	specifies the height in secondary grid units of the first row
first_col_width	specifies the width in secondary grid units of the first column
title_just	specifies the justification to be used for the table's title
title_font	specifies the font size to be used for the table's title
name_just	specifies the justification to be used for the table's name
name_font	specifies the font size to be used for the table's name
upper_left_just	specifies the justification to be used for the text placed in the upper left corner cell of the table
upper_left_font	specifies the font size to be used for the text placed in the upper left corner cell of the table
first_row_just	specifies the justification to be used for the table's first row other than the first entry of the first row (covered by *upper_left_just*)
first_row_font	specifies the font size to be used for the table's first row other than the first entry of the first row (covered by *upper_left_font*)
first_col_just	specifies the justification to be used for the table's first column other than the first entry of the first column (covered by *upper_left_just*)
name_font	specifies the font size to be used for the table's first column other than the first entry of the first column (covered by *upper_left_font*)
rest_of_table_just	specifies the justification to be used for all other cells of the table. This includes all the cells to the bottom right.
rest_of_table_font	specifies the font size to be used for all other cells of the table. This includes all the cells to the bottom right.

TableAttr *num value*

The specified table attribute applies to the previously defined table *num* specifies the table attribute number. Currently there two valid choices:

0	specifies the table name attribute
1	specifies the table title attribute

value specifies the value of the table attribute

TableData *row col value*

The specified table data refers to the previously defined table

row col specifies the element in the table for which data is being defined

value specifies the data

The EDIF Interfaces

This section describes the Electronic Design Interchange Format (EDIF) interface for ECS symbols and netlists. EDIF is an emerging standard format (or language) for ASCII data that allows electronic design information to be exchanged among different CAD/CAE systems. The ECS utilizes a subset of this format as an ASCII interface to its symbol libraries.

The EDIF standard is very broad in scope and encompasses many data formats that are not used in the ECS. As a result, not all EDIF files translate correctly into ECS symbol files. The two interface programs used for EDIF interface are shown in Figure EDIF-1. These applications are typically stored in the main ECS folder.

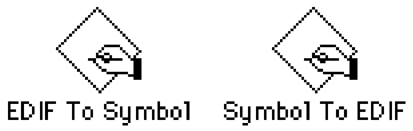


Figure EDIF-1 The EDIF Interface Applications

EDIF Netlist Interface

An EDIF netlist format is available from the **Processes** menu of the Hierarchy Navigator. To add this EDIF netlister to the Hierarchy Navigator menu, use the **Tools->Processes** section of the ECS Preferences Editor and create an entry for the *edifnet* application file. The following command line options are available:

/N	use notepad to view netlist immediately after it is written
/E	use external statement instead of library statement on primitives
/S	suppress whitespace in netlist, makes netlist smaller but less readable

EDIF Schematic Interface

An EDIF interface to ECS schematics is available from a third party vendor. Ask for details from your CAD/CAM representative.

Operation of EDIF Symbol Interfaces

Use the following procedure to invoke one of the EDIF interface programs.

1. Double click on the desired application in the main ECS folder. This invokes the application and pops up a dialog box as shown in Figure EDIF-2.
2. Navigate the file structure to view the contents of the target folder. Only those files with the appropriate extension are displayed.
    - .edf for *EDIF to Symbol*

- .sym for *Symbol to EDIF*

3. Select a single file to convert by double clicking on the file name. Click the *ALL* button to convert all listed files. The original files are left unchanged and new files with the extension of *.sym* or *.edf* as appropriate are created.

Because EDIF files can contain more than one symbol, the file name of each symbol is the name of the *cell* in the EDIF file with the extension *.sym*. If errors or warnings are issued by the EDIF translator they are written to an error file named *symbol_name.err,* where *symbol_name* is the name of the EDIF file being translated. Errors in the error file are flagged with the line number on which the error occurred.

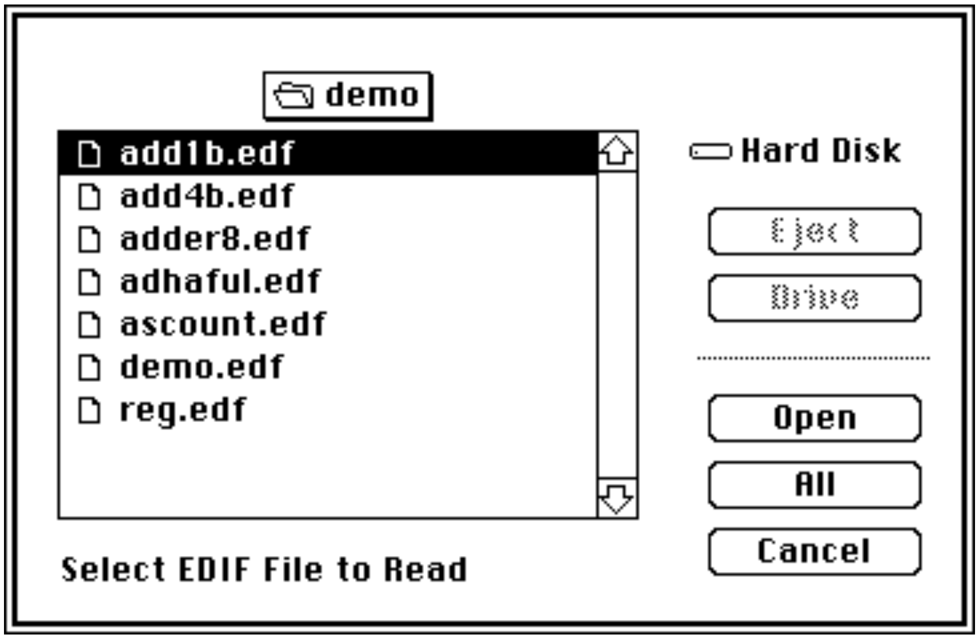


Figure EDIF-2 Dialog Box for SYM to ASCII

EDIF File Format

The EDIF format is briefly described here. For a complete description of the EDIF standard refer to the Electronic Industries Association publication, Electronic Design Interchange Format Version 2 0 0 (EIA Interim Standard No. 44).

The EDIF syntax consists of EDIF statements:

(keywordName { form })

A left parenthesis is followed by a keyword name, followed by one or more *forms*, (a sequence of *identifiers*, *primitive data*, *symbolic constants*, or *EDIF statements*), followed by a right parenthesis.

Keywords

The semantics of EDIF is defined by the keywords. Keywords are the only type of name that can immediately follow a left parenthesis. Case is not significant in keywords.

Identifiers

An identifier is a legal sequence of characters representing the name of an object or group of data. Identifiers are used for name definition, name reference, keywords, and symbolic constants. Valid EDIF identifiers consist of alphanumeric or underscore characters and must be preceded by an ampersand (&) if the first character is not alphabetic. The ampersand is not considered part of the name. The length of an identifier is from 1 to 255 characters and case is not significant.

Numbers

Numbers in EDIF are 32 bit signed integers. Real numbers can be represented by a special EDIF statement called *scaledInteger*. For example, The number 1.4 is represented as *(e 14 -1)*. The ***e*** form requires a mantissa and an exponent. The resulting real number is restricted to the range of plus or minus 1 times 10 to the plus or minus 35 power.

Strings

Valid EDIF strings consist of sequences of ASCII characters enclosed in quotes. Any alphanumeric character is allowed as well as any character from the following string: **!#$&'()*+,- ./:;<=>?@[\]^_`{|}~**. Special characters, such as " and % are entered as escape sequences *%number%* where *number* is the integer value of the ASCII character. For example, "*quote % 34 % character*" is a string with an embedded quote character.

White Space

Blank, tab, line feed, and carriage return characters (white space) are used as delimiters in EDIF. Blank and tab characters are also significant when they appear in strings.

Symbolic Constants

A symbolic constant is a defined EDIF name with a predefined meaning. For example *LOWERLEFT* is used to specify text justification.

Points

The *pt* construct is used to specify coordinate locations in EDIF. The keyword *pt* must be followed by the x-location and the y-location. For example: *(pt 100 200)* is at x=100, y=200.

Scaling

Numbers in EDIF must be scaled to relate them to numbers in the real world. The units of a particular EDIF number are determined according to where the number occurs. Coordinate locations and graphic descriptions such as line width are measured in units of distance and must be related to meters. Each coordinate value can be converted to a number of meters by applying the scale factor. Each EDIF library has a *technology* section which has a required *numberDefinition* construct. The *scale* construct is used within *numberDefinition* to relate numeric values to physical units.

Renaming Object Names

Names are used in an EDIF file to express relationships between objects and also for external reference. The name of a *figureGroup* is only used within the EDIF file as a reference to the EDIF structure which defines the drawing characteristics. Other names, such as *cell* names or *property* names are used by the ECS system. Sometimes the external name of an object is not a valid EDIF identifier. When this happens, the *rename* construct can be used to create a valid EDIF identifier and preserve the external name. For example, the symbol *test$1.sym* could generate the following *cell* construct:

```
(cell (rename test_1 "test$1") ...
```

The above example shows that the EDIF string is used to contain the original name, and a new name, *test_1*, is created as an EDIF identifier.

Key Points For Creating EDIF Files

The transfer from EDIF to ECS symbol files is not always 100% compatible because of differences between the ECS and other CAE systems. Some points to be aware of while performing EDIF translations are:

- The *scale* for *unit distance* must be defined correctly in the *numberDefinition* section of the *technology* block in each library.
- The ECS requires pins, symbols and nets to be on grid and permits graphics objects to be positioned at a resolution of one quarter of the grid. As a result, there can be problems in getting the pins of a symbol to line up on major grids in the ECS.
- The *pathWidth*, *borderWidth*, and *textHeight* should be defined in a *figureGroup* in the *technology* block.
- Because the ECS has only three choices for fonts, text may not appear to have mapped correctly.
- In order to correctly interpret symbol and pin attributes, the SymbolType should be specified as a *property* of the *cell.*
- The *cell* name defines the name of the translated symbol.
- All pins should be defined using the *port* construct in the *interface* section of the *view*.
- The pin attributes must be defined in the **Attributes->Pin Attributes** section of the ECS Preferences Editor.
- Symbol attributes should appear under *interface* before the *symbol* construct if there are any attribute display windows with that attribute.
- Symbol attributes must be defined in the **Attributes->Symbol Attributes** section of the ECS Preferences Editor.

- Arcs should be carefully specified because there is a possibility of rounding errors in the conversion between a three point arc and a center and radius description of the arc.
- The entire world must be limited so that it fits within the ECS world limit for symbols of 400 major grids.
- In displaying pin names, the ECS places pin names from 1 to 15 quarter grids from the pin.
- The ECS uses the SymbolType property to distinguish Gates, Cells, Components, Blocks and Graphic symbols.
- The ECS supports two line widths.
- The ECS allows system wide specification of color for nets and pins.
- The set of attributes which are relevant to the ECS may not be the same set of properties which are relevant to other CAE systems.

Before reading in EDIF files that were produced on another system, the following procedure should be followed.

1. Be sure that the Grid setting in the **Controls->Sheet Layout** section of the ECS Preferences Editor is set to match the grid setting from the source CAE system. Adjusting the Grid setting causes the entire symbol to be scaled up or down.
2. Add Attribute definitions to the **Attributes->Symbol Attributes** and **Attributes->Pin Attributes** sections of the ECS Preferences Editor to correspond to any attributes that were defined in the source CAE system but which are not currently defined by ECS.

   Check the spelling of the attribute names and change the spelling as needed so that it matches the spelling of properties in the EDIF file. For example, *PartNum* in ECS could be spelled *Part_No* in the source CAE system. Once the EDIF files have been converted you can change the spelling back if you wish. The spelling of attribute names is only significant during translation of files to ASCII or EDIF.
3. Determine the desired symbol type. The symbol can be a BLOCK, GATE, COMP, CELL, MASTER, PIN, or GRAPHIC type. The symbol type is normally determined by having a property on the cell which specifies the type. If this property is not present the symbols created during EDIF to ECS translation have the type which was specified by the default *SymbolType* in the **Controls->System Controls** section of the ECS Preferences Editor. If there is no *SymbolType* property in the EDIF file and no *SymbolType* default in the ECS Preferences Editor, the symbol is created as a BLOCK type symbol.

Generic Netlists

Several generic netlisters are supplied with the standard ECS release. The ASCII output of these generic netlists is often in a form that can be easily modified for custom applications.

These netlisters are consolidated into a single executable file called **lister**. This program is accessed through the **Processes** menu of the Hierarchy Navigator. Each separate netlister is specified by using one of the command line options listed below:

-netorder	net list by net
-pinorder	net list by instance and pin
-listmark	list only marked nets and instances
-listinst	list type, location, parent and instance names
-listpart	list block symbols, count of primitive symbols

The syntax in the **Tools->Navigator Processes** section of the ECS Preferences Editor for the netlist by net netlister is

```
lister  -netorder
```

The following options are also available on all the netlisters:

-nohead	suppress printing of header with time and date
-pcb	use reference designator and pin number instead of instance name and pin name
-view	run the currently specified editor to view the netlist file immediately after netlist creation
-ext=.abc	create a netlist file with the specified extension, i.e., *root_design.abc*

Processes->Netlist By Net *(netorder)*

The *-netorder* option generates a flat ASCII netlist from the current design. Only primitive symbols and their pins are included in the listings. Buses are resolved into the individual signals and do not appear as buses in the listings.

The generated netlist is ordered by net. The listing is divided into two sections. The first section of the listing contains a list of the symbol instances showing the symbol type followed by the instance name. Each instance is on a separate line and there is no sorting performed to group instances of the same symbol type. A line of dashes ends this list.

The second section of the listing contains a list of each net followed by a list of the pins that are contained in the net. The first entry in the line is the net name. The following entries are the pins that are connected to the net.

If more than one line is required to list a net, the line to be continued is ended with an ampersand and the following line is indented.

The following example shows the Netlist format for **Processes->Net List By Net**. The circuit in Figure Generic-1 is used as the basis for this example.

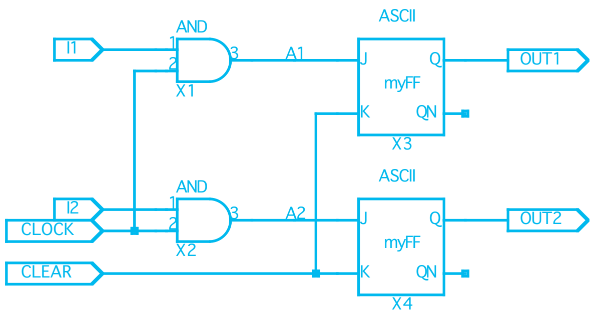


Figure Generic-1 Sample Circuit to Demonstrate Netlist By Net.

JKFF	X3		
JKFF	X4		
AND2	X1		
AND2	X2		

A2		X4-J	X2-OUT
OUT2	X4-Q		
CLEAR	X4-K	X3-K	
CLOCK	X1-IN2	X2-IN2	
I2	X2-IN1		
OUT1	X3-Q		
A1		X3-J	X1-OUT
I1	X1-IN1		

Processes->Netlist By Pin *(pinorder)*

The *-pinorder* option generates a flat ASCII netlist from the current design. Only primitive symbols and their pins are included in the listings. Buses are resolved into the individual signals and do not appear as buses in the listings.

The output netlist is ordered by symbol instances. The two entries are symbol type followed by instance name. The entries which follow this represent the symbol pins. Each pin entry contains the name of the pin followed by the name of the net to which it is connected. Pins that are not connected are listed with the word *Unconnected* in place of the net name.

The symbol and each of its pins are listed on a separate line. The line containing the instance is not indented to differentiate it from the lines containing the pins.

The following example illustrates the Netlist format for **Processes->Net List By Pin**. The circuit in Figure 6-2 is used as the basis for this example.

```
JKFF            X3
        J       A1
        K       CLEAR
        Q       OUT1
        QN      Unconnected
JKFF            X4
        J       A2
        K       CLEAR
        Q               OUT2
        QN              Unconnected
AND2            X1
        IN1     I1
        IN2     CLOCK
        OUT             A1
AND2            X2
        IN1     I2
        IN2     CLOCK
        OUT     A2
```

PCB Back Annotation Interfaces

The Engineering Capture System provides a number of Printed Circuit Board (PCB) back annotation interfaces. The following netlist formats are currently supported:

- PADS PCB
- Redac Interchange Format (RINF)

When additional netlist formats are developed, they are documented in release notes and update notes.

The PCB back annotation interfaces are consolidated into one program called *pcbback*. In order to use a PCB back annotation interface, you must add an entry to your **ecs.ini** file using the **Tools->Processes** menu of the **ecs.ini** Editor as described in the *System Configuration* section. The entry consists of the application name *pcbback* with one of the following command line options:

-pads	read PADS back annotation file *design_name.eco*
-rinf	read Recal Redac RINF format back annotation file *design_name.irp*

PADS Specific Features

The *design_name.eco* file, where *design_name* is the name of your design, must have been produced by PADS. The back annotation interface opens this file automatically and updates attributes in the Navigator to reflect any changes. You must perform a save from the Hierarchy Navigator in order to preserve the back annotated changes. Failure to perform a save from the Navigator results in the back annotated changes being lost.

The PADS statements recognized by the back annotation program are:

REN_PART
SWPPINS
SWPGATES
REM

The error messages created by the pads option are:

Unable to Open Error File ____ For Writing
The pcbback module could not open a file to list errors. This is usually a result of incorrect read/write file or directory permissions.

Missing PADS Back-Annotation File
The file *design_name.eco* does not exist or could not be found.

Insufficient Memory to Process Design
You need more memory in your machine to run this module.

RINF Specific Features

The *design_name.irp* file, where *design_name* is the name of your design, must have been produced by the Racal Redac program. The back annotation interface opens this file automatically and updates attributes in the Navigator to reflect any changes. You must perform a save from the Hierarchy Navigator in order to preserve the back annotated changes. Failure to perform a save from the Navigator results in the back annotated changes being lost.

The RINF statements recognized by the back annotation program are:

.REN_COM
.SWA_PIN
.SWA_GAT
.REM

The error messages created by the rinf option are:

Unable to Open Error File ____ For Writing
The pcbback module could not open a file to list errors. This is usually a result of incorrect read/write file or directory permissions.

Missing RINF Back-Annotation File
The file *design_name.irp* does not exist or could not be found.

Insufficient Memory to Process Design
You need more memory in your machine to run this module.

PCB Netlisters

The Engineering Capture System provides a number of Printed Circuit Board (PCB) netlist interfaces. The following netlist formats are currently supported:

- PADS PCB
- Redac Interchange Format (RINF)
- Cadnetix

When additional netlist formats are developed, they are documented in release notes and update notes.

The PCB netlisters are consolidated into one program called *pcbnet*. In order to use a PCB netlister, you must add an entry to your ECS preferences using the **Tools->Processes** menu of the ECS Preferences Editor as described in the *System Configuration* section. The entry consists of the application name *pcbnet* with one of the following command line options:

-pads	write PADS netlist file
-rinf	write Recal Redac RINF format netlist file
-cadnetix	write Cadnetix netlist file

The remainder of this PCB netlister section is broken into a discussion of features common to all of the PCB netlisters, followed by discussions of the individual netlist features.

Features Common to All PCB Netlisters

PCB netlists share many features. Elements in the netlist are referenced using *reference designators* and *pin numbers*. The different formats generally have a different arrangement of reference designators and pin numbers.

Necessary Attributes

The following attributes must be defined in the Attributes section of the ECS Preferences Editor in order for the PCB interfaces to work properly:

RefName	This is the reference designator attribute used in PCBs.
PinName	The PinName attribute identifies pins for the PADS interface.
Polarity	The Polarity attribute identifies the polarity of pins. The allowed values for this attribute are: INPUT, OUTPUT, and BIDIR. The abbreviations I, O, and B are also allowed.

PinNumber	The PinNumber attribute is used to indicate component pin numbers. For example, the symbol for the 7400 series NAND Gate has pins with the following names and numbers:

PinName= IN1	PinNumber= 1 4 9 12
PinName= IN2	PinNumber= 2 5 10 13
PinName= OUT	PinNumber= 3 6 8 11

PCB_Global	10 symbol attributes are used to represent hidden power and ground pins in component symbols. These are described in more detail later in this section.
PinGroup	The PinGroup attribute indicates which other pins in a component have the same function. In a 7400 series NAND gate, the inputs have the same PinGroup. Pins with the same PinGroup attribute can be swapped in a Gate symbol.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attributes** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. The default values can be overridden for a particular instance of the symbol on the schematic by use of the **Add->Attribute** command of the Schematic Editor.

Please refer to the *Entering Your Design* section in this manual for more details on attributes in the ECS environment.

Preparation of Schematics

The circuit must be fully packaged prior to extracting the netlist. Each symbol should have a Reference Designator assigned. Pin numbers are assigned according to the gate sections within each package. See the *Differences Between IC and PCB Design* section for more details on instance names and reference designators. See the *Design Analysis Tools* section for more information on the schematic autopackager.

Reference Designators and pin numbers are assigned in either the Schematic Editor using the **Add** menu in the Hierarchy Navigator using the **Misc** menu. Assignments in the Navigator override assignments made in the Schematic Editor. If the design contains any replicated hierarchy, the Hierarchy Navigator must be used to assign the reference designators and pin numbers to the gates and components in the replicated portions.

The processor also requires that any symbols representing hierarchical blocks have corresponding schematics linked into the hierarchy. At the lowest level of the hierarchy, all symbols should be of type *Gate* or *Component*.

DeMorgan equivalent symbols are identically named except for a D on the alternate version (7400.SYM and 7400D.SYM). The processor considers the two symbols equivalent and will allow them to share a package.

PCB Power Pins

Most components on printed circuit boards have at least one electrical connection to power supplies (usually VCC and GND). Showing all of these power connections explicitly can clutter the schematic. It is often desirable to hide the power connections in some way. The ECS provides a way to hide these power pins from view and still maintain the proper electrical connectivity.

Symbol attributes numbered 60 through 69 are used by the pcbnet netlister to represent hidden power and ground pins. The name of the attribute corresponds to the name of the global net. The value of the attribute corresponds to the pin number on the package.

For example, the default preferences shipped with the ECS have the following attribute names corresponding to the attribute numbers below:

60	GND
61	VDD
62	VCC

Use the following procedure to apply hidden power and ground pins to a 7400 quad NAND gate where pin 7 is connected to ground and pin 14 is connected to VCC.

1. Edit the symbol of the NAND gate using the Symbol Editor.
2. Use the **Add->Symbol Attribute** command to edit the value of the attribute named GND. Set the value of this attribute to 7, the pin number corresponding to the ground connection on the 14 pin DIP.
3. Use the **Add->Symbol Attribute** command to set the value of the VCC attribute to 14, the pin number corresponding to the power connection on the 14 pin DIP.

Typically you define attributes for GND and VDD. The range of attributes allows up to ten different nets to be defined as potential connections to these hidden pins. The symbol libraries must contain the default connections to the power and ground pins. If the attribute is not assigned or has the value *0* the net is not connected to that symbol.

If a particular pin is sometimes connected to VDD and sometimes connected to VCC, use the following procedure.

1. Edit the attributes on the symbol so that the VDD symbol attribute is the actual pin number and the attribute for VCC is a *0*.
2. Find the instances on the schematic which need to have this pin connected to VCC and set the VDD attribute to *0*.
3. Set the VCC attribute on these identified instances to the actual pin number.

The following command line option enables the use of hidden power pins in the PCB netlisters:

-power	use the hidden power pin option for PCB netlisting

An example of their use in the processes section of the ECS Preferences is:

```
PADS Netlist = pcbnet  -pads  -power
```

Command Line Options

The following command line options are available with the *pcbnet* netlister.

-pads	enables the PADS PCB netlister
-cadnetix	enables the Cadnetix PCB netlister
-rinf	enables the Racal Redac Interchange Format netlister
-power	enables hidden power and ground pins for PCB use
-nohead	suppress printing of header with time and date
-view	run the currently specified editor to view the netlist file immediately after netlist creation
-ext=.abc	create a netlist file with the specified extension, i.e., *root_design.abc*

Error Messages

The following error messages are used by the *pcbnet* netlister.

**** *Missing Pin Number(s) on Symbol Type …*
The pin numbers were not assigned to the symbol. Use the Symbol Editor to add them.

**** *Missing RefDes on Symbol Instance ...* ****
The Reference Designator was not assigned to the symbol.

**** *Missing .SCH file for Block Symbol ...* ****
Symbol ... was created as a Block type symbol. A corresponding schematic was therefore expected, but not found. Correct this by supplying the schematic or by recreating the symbol as a Gate or Component type. To change the symbol type, write the symbol to ASCII, edit the type and read in back in from ASCII.

**** *Encountered CELL Type Symbol ...* ****
Symbol ... was created as a Cell type symbol. This symbol type is intended for IC design and should not be used in Printed Circuit Board Applications. Convert to a Component or Gate type symbol.

**** *Two Instances of Pin ...-.. on Different Nets* ****
It is possible for the same pin to appear on different symbols, as in a common enable pin. This message results if different nets are connected to each of the instances of the common pin.

This message also occurs if two instances of a symbol are assigned the same Reference Designator and pins provided that they are not connected identically. It is possible for a gate to be duplicated in a schematic provided that the instances are connected identically. The resulting netlist is correct.

**** *RefDes ... Assigned to Different Symbol Types* ****
The same Reference Designator is assigned to two different types of symbols.

**** *Missing Power Net ... From Attribute ...* ****
An attribute in the range for global power nets has been defined but this net is not used anywhere on the schematic. The global net must exist on any schematic with symbols that have power pins.

**** *Non Global Power Net ... From Attribute ...* ****
An attribute in the range for global power nets has been defined but this net is not global.

PADS Specific Features

The following features apply only to the PADS netlist.

The PADS netlister supports the PartShape symbol attribute. The examples below show how the PartShape is formatted in the netlist. It follows the part type and is separated from the part type with an @ sign.

```
U1      LS7400@SO-14
U2      AMD386@PCC128
```

The net attribute *width* represents trace widths in PCB applications. This is netlisted with the proper syntax following the signal definition in the PADS netlist. Net attributes can only be added to a design in the Hierarchy Navigator using the **Misc->Attribute** command.

The PADS interface creates a netlist file with the name *design_name.pad*, where *design_name* is your design's name. This file should be copied to the PADS design directory and renamed with a suffix of *.asc*.

The PADS processor combines a checking function with the generation of the PADS compatible netlist. Error reports are inserted in the netlist file. If any errors are encountered, the processor invokes the editor and displays the entire netlist, including error reports, before returning to the Navigator.

If the *-power* command line option is not specified, the PADS netlister assumes you are using the standard PADS library of parts that come with the PADS PCB product. This library already has power and ground pin definitions in it and therefore does not require the ECS hidden power pins.

RINF Specific Features

The Engineering Capture System (ECS) provides an interface to the REDAC Interface Format (RINF). This interface allows you to extract a RINF compatible netlist. This processor creates a netlist file with the name *design_name.frp* where *design_name* is the same as your design's name.

The RINF processor combines a checking function with the generation of the RINF compatible netlist. Error reports are inserted in the netlist file. If any errors are encountered, the processor invokes the editor and displays the entire netlist, including error reports, before returning to the Navigator.

The SILOS Interface

The Engineering Capture System (ECS) provides an interface to the SILOS Logic Simulator. This interface allows you to extract a netlist file for the SILOS simulator. You can view the simulation results using the Waveform Tool (WT). You can highlight signal logic states directly on the schematic using the combination of the WT and Hierarchy Navigator.

The SILOS Netlist Processor is selected from the **Processes** menu of the Hierarchy Navigator. This processor creates a netlist file with the name *design_name.dat*, where *design_name* is your design's name. This file is a topological description of the hierarchical design and is combined with a user-supplied pattern file which specifies the initial values, clock, and pattern definitions for the simulator. The pattern file is typically named *design_name.pat*. Any standard text editor can be used to create the pattern file.

If errors are encountered while writing the netlist, the processor invokes the editor on the netlist file. The editor is used to scroll through the netlist file and identify the errors. Errors are displayed on lines surrounded by asterisks. For example:

```
**** Missing Pin Named: EN ****
```

Required Configuration Information

The application *SilosNet* must be installed under the **Processes** menu using the ECS Preferences Editor. See the *System Configuration* section for more information on installing applications under the **Processes** menu. The following flags are used to modify the execution of the SILOS netlister.

Command Line Options

/A	Automatic mode, do not present dialog box
/M	Write sub-circuit netlist
/P	Use lowest level primitives
/S	Invoke the SILOS simulator
-nohead	suppress printing of header with time and date
-pcb	use reference designator and pin number instead of instance name and pin name
-view	run the currently specified editor to view the netlist file immediately after netlist creation
-ext=.abc	create a netlist file with the specified extension, i.e., *root_design.abc*

The netlister brings up the option dialog box every time it is invoked if the *A* option is not specified. If the *A* option is specified the netlister does not display the dialog box but does use any other options specified on the command line. If other options are specified without the *A* option, the dialog box pops up with the selected options as the defaults.

Necessary Attributes

There are four attributes that are reserved for use with the SILOS interface. These attributes are defined using the ECS Preferences Editor and are used to provide the necessary information for constructing the network description.

SilosModel	This symbol attribute identifies the primitive or macro used to represent the symbol. Primitives are identified by the first three characters after a leading period (.). Macros are specified by name without a leading period. Macros must be defined in one of the library files *SYSTM.LIB* or *MACRO.LIB*. The remainder of the attribute (after the macro or primitive name) is copied into the network description without interpretation. Therefore, if a strength qualifier is desired on a primitive, it can be appended to the primitive name in this attribute.
SilosTimes	This symbol attribute specifies the rise and fall times and multipliers for the gate. The attribute expects two fields. Rise time is the first field and fall time is the second field. If present, this string is inserted into the network description between the SilosModel and the pins.
SilosName	This pin attribute is used for macros and several of the primitives to determine the order that the pins are listed in the netlist. For macros, pins are identified by number and appear in the netlist in the same order as the numbers in the SilosName attribute. For some primitives, the pins are identified by name. If the PinName is different than the name required to identify the pin for SILOS, the SilosName attribute can be used. The netlist processor checks the SilosName before looking at the PinName. If the first character in this attribute is a minus sign (-) the netlister places a leading minus sign in front of the net name connected to this pin. This feature treats the pin as if there is an inverter on the pin. The symbol should show an inverter bubble on this pin to indicate the inversion and avoid confusion.
SilosLoad	This pin attribute indicates the optional numerical load factor used in computing the fanout-dependent delay load for all gates connected to this input pin. If present, this string is inserted into the network description between the SilosModel and the pins.

There are several other attributes which are used by the SILOS interface.

TreeStop	If a symbol has TreeStop=*YES*, it becomes a primitive. This is used to split large hierarchies. Refer to the *Hierarchy Navigator* section for more information on split hierarchies.
Value	The Value attribute is used on *.CAP* symbols to specify the capacitance.

Polarity	The polarity is used by the SILOS netlister to determine which pins are the output pins for several of the primitive types.
PinName	The pin name is used to identify pins for some primitives if the SilosName attribute is not present.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attribute** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. The default values can be overridden for a particular instance of the symbol on the schematic by use of the **Add->Attribute** command of the Schematic Editor.

Please refer to the *Entering Your Design* section in this manual for more details on attributes in the ECS environment.

Creating the Pattern File

The pattern file is one of two files required by SILOS to perform simulations. It provides the stimulus for the circuit such as the clocking signals and any other primary inputs. The pattern file also contains the location of the netlist file. You must create a pattern file for each simulation, typically using a text editor. The pattern file has an extension of *.PAT*.

The following is an example pattern file named *DEMO.PAT* for use with a design called *DEMO.SCH*.

Anything after a dollar sign ($) is a comment.

```
!INPUT DEMO.DAT                      $Causes SILOS to read the netlist
VDD .CLK 0 S1                        $define the global power and ground
GND .CLK 0 S0
RESET  .CLK 0 S1 160 S0              $ Specify all input signals
CLOCK  .CLK 0 S0 100 S1 150 S0    .REP  0
```

Please refer to the *SILOS Reference Manual* for more information about clock and pattern inputs.

Controlling The Netlist Extraction

The netlist file is one of two files required by SILOS. The netlist file contains all the connectivity information for the circuit.

When the SILOS netlist processor is selected from the **Processes** menu of the Hierarchy Navigator a dialog box pops up with several program options.

Netlist	writes the netlist only
Sub Circuit	writes a special version of the netlist for a sub-hierarchy when working with a split hierarchy

Select the required option and invoke the process by clicking the mouse on the *Run* button.

In addition to the buttons, there is a check box labelled *Use Primitives*. Normally when the netlist processor encounters a symbol that has the SilosModel attribute specified, it codes that macro or primitive into the netlist without looking for any lower level schematics. If the *Use Primitives* option is selected, the netlist processor ignores the SilosModel on higher level blocks and codes the netlist using only the lowest level primitives. This feature makes it possible to switch between simulating at the functional level and the gate level without changing anything on the schematic.

For example, a design contains a schematic with a symbol called *NAND2* and the SilosModel attribute of this symbol is *.NAND*. An underlying schematic exists for the symbol *NAND2* containing transistor symbols *NMOS* and *PMOS* (SilosModel= *.NMOS* and *.PMOS*). The normal option codes the netlist with *.NAND* without any reference to the underlying schematic. With the *Use Primitives* option, *NAND2* is coded as a *.MACRO* using statements containing *.NMOS* and *.PMOS*.

SILOS Format Conversions

The SILOS netlister makes the following conversions in the netlist file extracted from the ECS schematic.

- The apostrophe character, which is used for causing names to be written with overbars, is replaced with the underscore (_) character.
- SILOS uses the left parenthesis (as its hierarchical separator rather than the period.

Be aware of these changes when interpreting error messages from SILOS. For example, if the root schematic has a register with instance name I_1 and the register has a net called ADDR, the resulting name for SILOS would be (I_1(ADDR rather than the ECS name of I_1.ADDR.

The SILOS netlister writes a hierarchical netlist. Because of this, any instance-specific attribute overrides, which are added by the Hierarchy Navigator, do not appear in the netlist.

Building SILOS Symbol Libraries

SILOS has many built in primitive functions that are used in simulation. Several attributes in these primitives must take on prescribed values.

The following list gives the attributes needed for various SILOS primitives.

Combinatorial Gates
SilosModel= .INV, .AND, .NAND, .OR, .NOR, .XOR, .XNOR, or, .GATn
Polarity= OUT on the output pin

Tri-State Combinatorial Gates

SilosModel= .TINV, .TAND, .TNAND, .TOR, or .TNOR

Polarity= OUT on the output pin

SilosName= begins with E on the enable pin

Name= begins with E on the enable pin

And-Or-Invert Gates

SilosModel= .AOI, or .OAI

Polarity=OUT on the output pin

SilosName= A, B, ... indicates the groups of pins

Level & Strength Delays

SilosModel= .DLA, or .SDLA

Polarity=OUT on the output pin

Flip-Flops

SilosModel=.DNEFF, .DPEFF, .JKNEFF, .JKPEFF, .SRNEFF, .SRPEFF, .TNEFF, or .TPEFF

SilosName= or Name= one of CLK and Q on all types. CLR, SET and QB are optional on all types. J, K, S, R, and D as appropriate.

Transfer Gates

SilosModel= .NXFR, .PXFR, .NRXFR, .PRXFR, .CXFR, or .CRXFR

SilosName= or Name= IN, OUT, and EN and/or EP on enable(s)

CMOS Transistors

SilosModel= .CMOS

SilosName= or Name= NS, NGP, NGN, or ND

MOS Transistors

SilosModel= one of .NMOS .PMOS

SilosName= or Name= NS, NG, ND

Zycad-compatible D Flip-Flop

SilosModel= .DFF

SilosName= or Name= Q, D, CLK, or R

Zycad-compatible Latch

SilosModel= .LAT

SilosName= or Name= Q, D, EN, R

Resistor

SilosModel= .RES

Polarity=OUT on the output pin

Capacitor

SilosModel= .CAP

Value= capacitance ratio (default is 1)

The capacitor symbol should only have one pin or if it has two pins one of them must be connected to GND or VSS.

Setup & Hold Detectors

SilosModel=.SHNE or .SHPE

SilosName= or Name= CLK or D

Polarity=OUT on the output pin

RAM, ROM, and PLA's

These primitives are not implemented due to the wide range of possible combinations. You should manually code macros for these elements into one of the system library files and set the SilosModel to the name of the macro.

Macros

SilosModel= name of the macro

SilosName= order for pins

Symbols that represent manually coded elements require that the pin output order be specified. This is accomplished by using the SilosName attribute to indicate the numeric order in which the pins are to be output. The macro definitions should be in one of the system library files ("SYSTM.LIB" or "MACRO.LIB").

Examples

Two examples of SILOS elements are presented. A J-K flip flop shows how to use a SILOS primitive. A four-bit inverter shows how to use a SILOS macro to build a non standard function.

Negative Edge-Triggered J-K Flip Flop

The following example is a negative edge-triggered J-K flip flop with active high set and clear. It has CMOS type strengths and a rising delay of 4ns and a falling delay of 2ns. The special attributes necessary are listed.

Symbol Attributes:

SilosModel= .JKNEFF/C

SilosTimes= 4 2

Pin Attributes:

PinName	SilosName	Polarity
J		In
K		In

CLK		In
SET	-SET	In
CLR	-CLR	In
Q		Out
QN	QB	Out

In this example, CMOS drive strength is shown by the */c* in the SilosModel definition. The rise delay of 4ns and the fall delay of 2ns are shown under SilosTimes. The last pin is named *QN* so it is displayed as *QN* on the schematic. Because the SILOS netlister requires the name to be identified as *QB*, the SilosName attribute is used. The *SET* and *CLR* pins are to be treated as active high rather than the default active low. The minus sign on the SilosName causes the net connected to this pin to be listed with a leading minus sign. The *QB* pin does not have to be connected on the schematic.

Four-Bit Inverter

The following example is a macro for a four-bit inverter whose symbol uses bus pins:

Macro Definition

```
.macro INVERT4  DATA0 DATA1 DATA2 DATA3 ENABLE OUT0 OUT1
OUT2 OUT3
OUT3 .TINV ENABLE DATA3
OUT2 .TINV ENABLE DATA2
OUT1 .TINV ENABLE DATA1
OUT0 .TINV ENABLE DATA0
.eom
```

Symbol Attributes

SilosModel= INVERT4

Pin Attributes

PinName	SilosName	Polarity
OUT[0;3]	6	Out
ENABLE	5	In
DATA[0;3]	1	In

If a pin is actually a bus pin (DATA[0;3]) the numbering sequence should treat that pin as a sequence of numbers.

Error Messages

No Model Defined for ...
A primitive symbol did not have the SilosModel defined.

Unknown Primitive Type: ...
The SilosModel attribute is not one of the supported types.

Missing Symbol for ...
The Sub-Circuit option requires a symbol for the root schematic.

Missing Sub Circuit File ...
If a symbol has a SilosModel specified without a leading period (.) or if the symbol has TreeStop=Yes for split hierarchy, there must be a file with the given name and extension *.dat* in the current directory or one of the project or model directories on the search path. Run the Hierarchy Navigator on the subhierarchy and use the Macro option of the Silos netlister to create this file.

Missing Pin Named: ...
A primitive symbol was expected to have a pin with the given Name or SilosName.

SilosName Not Specified for Pin in Symbol ...
Macros require that each pin have its SilosName specified as a number indicating the numerical order for the pin to be listed. This symbol is a Macro because SilosModel specified a name without a leading period.

Incorrect Pin Ordering in ...
This macro has a gap in the sequence of numbers in the SilosName attribute. This symbol is a Macro because SilosModel specified a name without a leading period.

No Output Pin on ...
This primitive requires a pin with Polarity=Out

Unconnected Output: ...
All primitives must have the outputs connected except for the *QB* pin of a Flip-Flop.

Unconnected Capacitor Pin On: ...
The Capacitor pin must be connected.

Capacitance Not to GND or VSS
If a Capacitor symbol has two pins, one of the pins must be connected to GND or VSS.

Missing or Invalid VALUE (Capacitance) for ...
The Value attribute must be set to the relative capacitance for this node.

Simulation Environment

The Simulation Environment is designed as a generic interface for including a simulator in the Engineering Capture System (ECS) environment. The operational model consists of five functions which can be interconnected, as shown in Figure Simulation-1. The ideal environment includes all of the functions. It is also possible to omit one or more of the functions while retaining the interaction among the remaining modules.

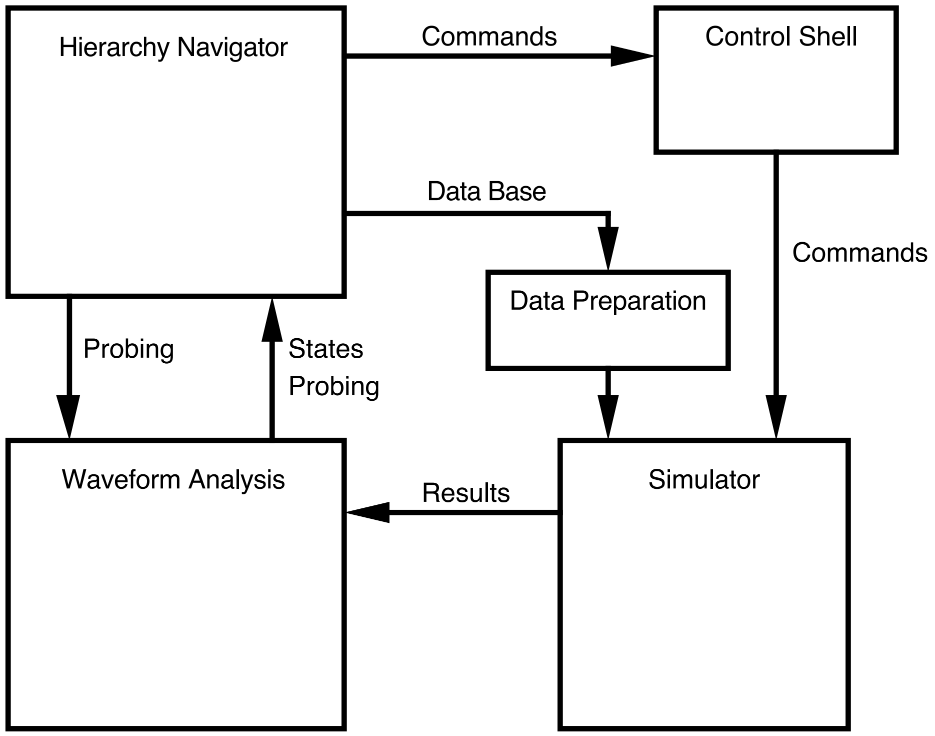


Figure Simulation-1 Operational Model of ECS/Simulation Environment

The environment is constructed to use the Hierarchy Navigator as the parent process. You begin your work session by invoking the Navigator to view your design. The Navigator combines the various symbols and schematics to create the Hierarchical data structure which is represented in the Navigator window. The Navigator also provides a *Simulation* menu which initially offers you the options of preparing the simulation models, compiling the models and of running the simulator.

If you select the **Code Sim Model** option, the Navigator spawns the Data Preparation module to extract the model information from the data base. The simulation models are prepared for the simulator to use.

If you select the **Compile Model** option, the Navigator spawns the process which compiles the models according to the simulator's requirements. This process can be omitted for some simulator environments.

You can then select the **Run Simulator** option in the menu. The Navigator responds by creating a control shell. The control shell consists of up to three buttons for starting and stopping the simulator. It also has a window into which both the Navigator and the user can enter commands.

The Navigator causes the Simulator to be invoked through this shell. If the Waveform Viewer is to be used, the Navigator initiates this tool prior to starting the Simulator. The Navigator modifies its *Simulation* menu to offer options appropriate to the simulation process.

Once the simulation process is invoked, you send commands to the simulator by using the *Simulation* menu in Nav, selecting the control shell's buttons or by typing commands into the control shell's main window. These commands enter the simulator as its standard input (stdin) as if they were typed into a normal operating system command window. Modification of the simulator to accommodate this portion of the interface should not be required.

The Navigator can also invoke the Waveform Analysis Tool - WAVES. When launched, WAVES opens a socket and awaits the connection of the Simulator. The Simulator can then connect to the Tool using a routine which is supplied as part of the Programmer's Interface Kit.

When connected, WAVES displays the simulation results. It also transmits the results (at the query cursor) to the Navigator for display of net state values. All of the other functions of WAVES and the Navigator, such as cross-probing, are also operational.

When all of the modules are operational and linked, the process is controlled through user interactions with the Navigator, the Waveform Analysis Tool and the Control Shell.

A subset of the environment can also be used. Simulators that have their own waveform viewing capabilities can omit the Waveform Analysis Tool. In this case, cross probing and display of simulation results on the schematic would not be available. The remaining functionality would be fully operational. In cases where long simulations are run in a batch environment, the Navigator/Waveform Analysis combination provides a powerful tool for viewing the results of the simulation from a history file.

Setup of the Simulation Environment

The simulation environment is defined within ECS by the use of a configuration file. The lack of any *hard coded* options in the ECS programs results in an interface which can be defined for any simulator without modifying the ECS product programs.

The *ecs.ini* file contains a *Simulator* parameter. This parameter is a text string which is modified using the ecs.ini editor on UNIX platforms and the PC, or using the ECS

Preferences Editor on the Mac. The string (denoted below as *SIM*) contains the base name of the configuration file which defines the simulation environment. Changing this control changes the simulation interface of any subsequent ECS process.

The name of the file controlling the interface is the Simulator name appended with a *.ini* suffix. The simulator name is coerced to lower case on UNIX workstations and case-insensitive on the PC and Mac. The system expects to find this control file in the local directory or in the $ECS_ROOT/data directory.

The *SIM.ini* file consists of several sections. Each section has a header identifying the section. The header is a name surrounded by square brackets, appearing on a single line in the file. All lines following the header, up to but not including the end of the file or another header, are considered part of the section.

The commands in each section are described below.

[SimTitle]

`Title = Simulator Name`
: Causes the name of the simulator to appear on the menu bar of the Navigator. This replaces the default name Simulator that appears if this section is omitted.

`Code = Code Model Name`
: Causes the name of the model coder to appear on the Simulator menu of the navigator. The text replaces the default name *Code SIM Models* that appears if this section is omitted.

`Compile = Compile Models Name`
: Causes the name of the model compiler to appear on the Simulation menu of the navigator. The text replaces the default name *Compile SIM Models* that appears if this section is omitted.

`Setup = Setup Program Name`
: Causes the name of the setup program to appear on the Simulation menu of the navigator. The text replaces the default name *Setup SIM* that appears if this section is omitted.

`Start = Start Simulator Name`
: Causes the name of the simulator to appear on the Simulation menu of the navigator. The text replaces the default name *Start SIM* that appears if this section is omitted.

`Tool3 = Command Line`
: Causes the command line to appear on the Simulation menu of the navigator. It adds an extra entry to the Simulator Menu.

`Tool4 = Command Line`
: Causes the command line to appear on the Simulation menu of the navigator. It adds an extra entry to the Simulator Menu.

[ModelBuilder]

This section specifies the process by which the simulation models (net lists) are extracted from the data base.

```
Method = Navigator
```

Specifies that the model is to be extracted from the hierarchy data base. This is the method used for most of the hierarchical net list and all of the flat net list extraction processes. If this method is selected, the *Process=* control is the only other one used in this section. Typically, a single file representing the entire design is generated.

```
Method = Schematic/Symbol
```

Specifies that the simulation models are to be extracted directly from the Symbol and Schematic data files. This method is used for the VHDL net list extraction process and is typically used for other hierarchical net list extractors that require that the bus structures be preserved.

Individual model files are extracted for each Symbol/Schematic pair. The models can be extracted in the Schematic or Symbol editor using the **Code Sim Model** command in the *Misc* menu of the editor. The Navigator can also be used to create/update all of the models of a design.

```
Process = command_line
```

If the Navigator method is specified, the menu entry **Code SIM Model** is created in the Navigator's Simulation menu. Choosing the **Code SIM Model** menu entry causes the command_line to be executed with the appropriate substitutions as described below.

If the Schematic/Symbol method is specified, **Code SIM Model** is added to the Misc menus of the Symbol and Schematic Editors, as well as the Simulation menu of the Navigator.

A complete design hierarchy is coded when the command is executed from the Navigator while only the current symbol or schematic is coded when the command is executed from the Symbol or Schematic Editors.

When the command is selected, the *command_line* is expanded by replacing any % characters with the path to the schematic. If the character immediately preceding the % is an underscore, _, the preceding string is inserted into the path as a prefix to the file name. i.e.,

```
xyz_proc % abc_%;
```

becomes:

```
xyz_proc /full_path/schem /full_path/abc_schem
```

The resulting command is executed. If you ran the command from either the schematic or symbol editors, the model file is displayed in the text editor.

When the **Code Sim Model** command is selected in the Navigator, the time stamps of the symbol file, the schematic file and the files for each of the symbols used in the schematic are checked against the date on the model file. The only models updated are those whose schematics or symbols have changed since the models were last coded.

The following controls only apply to the Schematic/Symbol method of extraction:

`RootProcess = command_line`

This line is only needed if the root level schematic of the design is netlisted differently than the rest of the design. In VHDL code, for example, it is common to netlist the top level of the design along with a test bench. This ensures that all the top level I/O parts are defined properly for VHDL.

When you execute **Code Sim** from the Navigator, if you have defined an entry for Root Process, the top level schematic is netlisted according to the instructions in *command_line*.

`Extension = file_extension`

This control specifies the file suffix used for the model files. It is used by the Navigator to find the model files.

`Path = file_path`

This control specifies the directory which contains the model files. The path can be specified as an absolute directory (begins with a slash, / in UNIX) or as a relative directory. In the latter case, the directory is relative to the directory containing the symbol file.

`ModelAttribute = Symbol Attribute Number`

If a value is specified for this attribute on a symbol, the system assumes that the model is already available and does not cause a model to be generated. Setting this attribute is effectively a switch preventing ECS from coding a model.

`NetAttribute = Net Attribute Number`

If a value is specified for this attribute on a net, the current netlister can use this information. A common example of using net attributes is for Verilog wire types. This parameter is used by the **Add->Simulator Net Attribute** command.

[ModelCompiler]

Some simulators require that the models be compiled prior to invoking the simulator. This section provides similar controls to the [ModelBuilder] section above.

`Process = command_line`

If the Navigator method is specified, the menu entry **Compile SIM Model** is created in the Navigator's *Simulation* menu. Choosing the **Compile SIM Model** menu entry causes the *command_line* to be executed with the appropriate substitutions as described above.

If the Schematic/Symbol method is specified, **Compile SIM Model** is added to the *Misc* menus of the Symbol and Schematic Editors, as well as the *Simulation* menu of the Navigator.

A complete design hierarchy is coded when the command is executed from the Navigator while only the current symbol or schematic is coded when the command is executed from the Symbol or Schematic Editors.

The following controls only apply to the Schematic/Symbol method of extraction:

`RootProcess = command_line`

This line is only needed if the root level schematic of the design is netlisted differently than the rest of the design. In VHDL code, for example, it is common to netlist the top level of the design along with a test bench. This ensures that all the top level I/O parts are defined properly for VHDL.

When you execute **Compile SIM Model** from the Navigator, if you have defined an entry for Root Process, the top level schematic is compiled according to the instructions in *command_line*.

`Extension = file_extension`

This control specifies the file suffix used for the compiled model files. It is used by the Navigator to find the compiled model files.

`Path = file_path`

This control specifies the directory which contains the compiled model files. The path can be specified as an absolute directory (in UNIX begins with a slash, /) or as a relative directory. In the latter case, the directory is relative to the directory containing the symbol file.

[SimControls]

`Command = command_line`

Adds the **Start Simulator** entry to the Navigator's Simulation menu. If not specified, the entire simulator control function is disabled. See the *[SimTitle]* section to change the text for this menu entry.

The *command_line* is the command which invokes the simulator. Prior to issuing this command, the Navigator scans the *command_line* and makes the following substitutions:

%	base name of root schematic
%%	full path name of root schematic - no suffix.
\n	a new line character - i.e., allows multiple commands
@	the host network id of the local machine (Sun only)
\\	a single backslash
!	the process id of the Navigator (Sun only)
\	do not replace the following character

This command either invokes the simulator directly or invokes a script which then invokes the simulator.

`Waves = command_line`

This control launches the Waveform Viewer prior to invoking the Simulator. The *command_line* is first expanded using the same substitutions as the **Command=** control and then used to launch WAVES.

`FirstMsg = message`

This is the first message that is sent to the Simulator upon launching. Typically this is the message that causes the Simulator to perform its initialization and to connect to the Waveform Viewer. The message is expanded using the same substitutions as the **Command=** control before sending it to the Simulator. This message must contain the terminator character since it is not appended automatically.

`ExitMsg = message`

This is the message that is sent to the Simulator causing it to exit when the Navigator is closed. If no message is specified, the Simulator sends a **SIGKILL** message (on UNIX platforms), to the Navigator.

`Setup = command_line`

Adds the **Setup Simulator** entry to the Navigator's Simulation menu. Typically this spawns a text editor to edit a setup file to be read by the simulator. If the text editor normally expects to have a command tool or shell, **cmdtool** should appear in the string (i.e., **cmdtool vi %.cfg**).

The *command_line* is expanded by replacing the % character with the base name of the root schematic before being executed.

`Tool3 = command_line`

Adds an additional entry similar to **Setup** to the Navigator's Simulation menu. Typically this spawns a simulator in batch mode or WAVES to view history files in binary format.

The *command_line* is expanded by replacing the % character with the base name of the root schematic before being executed.

`Tool4 = command_line`

Adds an additional entry similar to **Setup** to the Navigator's Simulation menu. Typically this spawns a simulator in batch mode or WAVES to view history files in binary format.

The *command_line* is expanded by replacing the % character with the base name of the root schematic before being executed.

`Continue=c`

Specifies the continuation character to use when breaking long commands. The default is not to use a continuation character. A typical option is to precede the new_line with a backslash (\)

`MaxLine=nn`

Specifies the maximum line length that a command can take before inserting a continuation character and a new_line.

`RootIOPrefix=string`

Specifies the prefix to be added to the names of nets that are external to the root schematic. Nets that are external to the top level schematic are primary I/O ports.

`RootPrefix = string`

Specifies the prefix to be added to the names of nets that are local to the root schematic and are not primary I/O ports.

`SubPrefix = string`
: Specifies the prefix to be added to any nets that do not appear in the root schematic (appear as local nets in lower level schematics).

`InstPrefix = string`
: Specifies the prefix to be added to the hierarchical path name which is used to set the scope of the simulator when the schematic is *pushed* or *popped*.

`FlattenBuses = No`
: Normally, buses are flattened to the scalar signals when included in menu commands. If this option is turned off, the name of the bus is placed in the command. The name is the local name and the simulator must understand the hierarchical context. The name is processed to coerce the parentheses, range delimiter and the delimiter between signals in the complex names. (See Concatenate below.)

`HierChar=c`
: Specifies the character denoting hierarchy levels in hierarchical names sent to the simulator. This character replaces the normal period, ., and dash, -, in names within the ECS environment.

`BarChar=B`
: Signals which are complemented have either a single quote or an underscore as the leading character. This control specifies that a prefix of **B** is attached to these signals. This is used for compatibility with the VHDL simulators which do not permit leading underscores in a net name.

`Separator = c`
: Specifies the character used to delimit lists of nets in commands. The default is a blank space. Comma is a typical option.

`Terminator = c`
: Each command sent to the simulator has a terminator character appended before the final new-line. The default is not to append a terminator. A typical option is a semicolon.

`Parentheses=ccc`
: Specifies the type of parentheses that are to be substituted for the parentheses and delimiter in bus names. The first and last character are the parentheses and the middle character is the delimiter. The default condition is to use the parentheses in the schematic and the colon, :, as a delimiter.

`Concatenate=c`
: When buses are not flattened, complex buses (such as *a,b,data[0:3]*) are processed by replacing the commas and backslashes with the concatenate character.

`PushPop=string`
: Specifies the command sent to the simulator when the Navigator performs a push

or pop and when first starting. The command is formed by replacing the first % in this string with the *RootPre=* value and the Navigator's current hierarchical context. If no % is in the string, the path is appended to the string.

[SimMenu]

This section contains the list of menu buttons displayed on the Navigator’s Simulation menu. A limit of 25 entries can be placed in this menu. The syntax of the entries in this section is:

```
menu_item = format_string
```

where `menu_item` is the string that appears on the menu and `format_string` describes the command that is sent to the simulator via the TTY window.

The `format_string` contains the information which defines the operation of the command to NAV. The string is a template of the command which is sent to the simulator. Within the template, controls serve to define the operation of the command to the Navigator.

Simple and interactive commands can be specified. Simple, (immediate), commands are executed immediately when you select the command from the menu. These commands do not have any of the special format indicators described below (%n %*n %i). The `format_string` for immediate commands must not contain any special characters other than a single %. The % character must not be followed by n, i or *. Each time the menu item is selected, the % sign, if any appear in the `format_string`, is replaced by the name of the design file and the resulting message is sent to the simulator as if you typed it.

Interactive commands allow you to select nets or instances by clicking on them. These commands are indicated by the presence of special substitution indicator described below (%n %*n %i). When you select one of these commands from the Simulator menu, the system enters an interactive command mode allowing you to select nets or instances. The Navigator displays a special prompt indicating which command it is executing. Each time you select a net or instance the special substitution characters in the `format_string` are replaced by the names of the selected element(s) and the resulting command is sent to the simulator. One format string can contain only a single substitution indicator. The substitution indicators are shown below.

`%n`	The Navigator allows you to click the mouse on individual nets. Each time a net is selected the name of the net replaces the *%n* and the command is sent to the simulator.
`%*n`	The Navigator allows you to select a group of nets by: • Clicking on the net • Clicking on a symbol instance to select all attached nets • Dragging a box around name flags and pins • Depressing the shift key to accumulate several of the above

The list of nets replaces the *%*n* and the resulting string is sent to the simulator.

`%i`	The Navigator accepts the selection of a symbol instance. The full hierarchical name of the instance is placed in the string and the command is sent to the simulator.
`%{ a bc ... }`	This option defines a set of choices that is displayed in a dialog box when the command is active. Each time a message is issued, the selected choice is substituted into the command. This option is intended for commands which are used to force states in the simulator. A choice of a question mark, ?, causes a type-in field to be included as a choice in the dialog box.

[SimButtons]

There are three buttons that are part of the Simulator Control Window. This section defines the text string which is sent to the simulator each time the button is clicked. If no string is specified, the button does not appear.

`Run = string`

Each time the Run button is clicked, *string*, followed by the terminator character, is sent to the simulator .

`Stop = string`

Each time the Stop button is clicked, *string*, followed by the terminator character, is sent to the simulator . If the first two characters in the string are ^C, they are replaced with a Control-C. This feature does not operate properly if the simulator is run by specifying a script file in the command.

`Delta = string`

A message is formed by replacing the % in the string with the integer value in the type-in field. If no % is found in the string, the value is appended to the string. The message is terminated with the terminator character and a \n.

[NetAttribute]

Entries in this section consist of a button label followed by the *value* corresponding to that button label. For example, the [NetAttribute] section in the verilog.ini file contains the following entries:

WIRE =

WAND = WAND

WOR = WOR

REG = REG

TRI = TRI

TRIAND = TRIAND

TRIOR = TRIOR

TRI1 = TRI1

TRIO = TRIO
TRIREG(S) = TRIREG(S)
TRIREG(M) = TRIREG(M)
TRIREG(L) = TRIREG(L)
SUPPLY0 = SUPPLY0
SUPPLY1 = SUPPLY1

This gives rise to the following dialog box in the schematic editor when you select the **Add->Verilog Net Attributes** command.

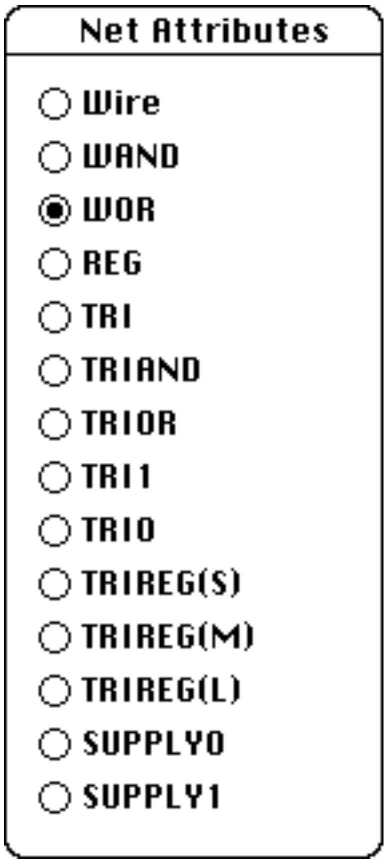


Figure Simulation-2 Dialog Box Displayed for ***Add->Verilog Net Attributes***

The buttons in the dialog box correspond to the names specified in the left hand column. If you select a net attribute by marking a radio button in the dialog box, the value on the right hand side of the equal sign is assigned to any selected nets. In the example shown, any nets selected have the attribute specified in **[ModelBuilder]->NetAttribute** assigned the value *WOR*.

This provides a general means of specifying values to one specific attribute for each simulator.

Launching Waves

The Dynamic Waveform Viewer program is named *waves*. It can be launched as a stand-alone application, from the Navigator's Tools menu or as part of the Navigator's *Simulation Environment*. In all three cases, the program is launched with a combination of command line arguments described below:

-nav	Specifies to the waves program that it is launched under control of the Navigator. This option is explicitly included in the *Waves=* control of the Simulation Environment or the **Controls->Navigator Tools** entry in the ECS Preferences Editor.
-sim	Specifies that the waves program is running dynamically with a simulator. If this control is missing, waves runs in the static mode using a history file for the simulation results. The waves program opens a socket to which the Simulator connects. The default port number that is used is 1703. This can be overridden by following the *-sim* control with the desired port number (i.e.,. -sim1705).
-time	Specifies that waves should accept its data from a history file written in the timewave history file format (described in the Waveform Tool section).
name	Specifies the root name for the display save file (.wav), the timewave history file (.his) if the *-time* option is specified and the binary save file.

The SPICE Interface

The Engineering Capture System provides an interface to several different versions of the SPICE Simulator. This interface extracts a netlist file in either a hierarchical or flattened format for simulation with Spice.

Both the flat and hierarchical netlisters generate netlists that are compatible with Berkeley Spice, HSPICE, PSpice, or LVS (Dracula layout versus schematic). The netlist can be written with the actual node names or with ECS assigned node numbers.

If errors are encountered while writing the netlist, the processor invokes the editor on the netlist file. The editor is used to scroll through the netlist file and identify the errors. Errors are displayed on lines surrounded by asterisks. For example:

**** Unconnected Pin on Transistor: N1 ****

Configuration Required For SPICE

The SPICE netlisters are accessed through the **Processes** menu of the Hierarchy Navigator. If you are using the configuration file provided by the CAD/CAM Group, there are probably already entries in the **Processes** menu for both flat and hierarchical SPICE netlisters. Otherwise, you must use the ECS Preferences Editor to ensure there are entries in the **Processes** menu for each of the SPICE netlisters. Refer to the *System Configuration* section for information on adding new applications to the **Processes** menu. The names of the SPICE netlister applications are:

SpiceNet	flat SPICE netlister application
HspiceNt	hierarchical SPICE netlister application

There are a number of optional flags that can be set in the ECS Preferences Editor for each of the SPICE netlisters. These flags modify the operation of the netlisters to obtain specific results.

The following options are mutually exclusive. Choose only one of the four:

/B use conventions of Berkeley Spice
/P use conventions of PSpice
/H use conventions of HSpice
/L use conventions of LVS

Any of the following options can be specified:

/A	automatic mode, do not present dialog box
/M	write sub circuit netlist (hierarchical netlist only)
/N	use node names instead of numbers
/G	allow use of .GLOBAL instruction (hierarchical netlist only)
/U	do not add *u* to length or width and do not add *p* to areas

/X	ignore Prefix of X on sub-circuits and Use Primitives (hierarchical netlist only)
-ext=.abc	create a netlist file with the specified extension, i.e., *root_design.abc*

If the *A* option is not specified, the netlister brings up the option dialog box every time it is invoked. If the *A* option is specified, the netlister does not display the dialog box but does use any other options specified on the command line. If other options are specified without the *A* option the dialog box pops up with the selected options as the defaults.

Necessary Attributes

There are several attributes which are reserved for use with the SPICE interface. These attributes are defined in the **ecs.ini** file and provide necessary information for constructing the network description.

Prefix	This symbol attribute identifies the type of element represented by the symbol. Only the first character is used to determine the type. Examples are Q, M, D, C, and R.
SpiceModel	This optional symbol attribute specifies the model name listed for this symbol. The model name should match the name on a *.MODEL* statement in the SPICE control file.
SpiceLine	This optional symbol attribute is used to add parameters to the SPICE network description of this symbol. If present, it is copied to the network description without interpretation. It is added to the end of the element instantiation line. SpiceLine can be used on any primitive symbol. It can also be used on instances of blocks in hierarchical netlists allowing parameters to be passed in hierarchical SPICE netlists. The SpiceLine attribute automatically wraps onto another line if it is longer than 80 characters.
SpiceLine2	This attribute is copied to the netlist as part of the subckt header when used on Block symbols. For primitive symbols this attribute is treated in the same way as SpiceLine. SpiceLine2 can be used on any primitive symbol. It can also be used on definitions of blocks in hierarchical netlists. The primary use of the Spiceline2 attribute is to define parameters on subcircuit definitions. The parameters defined using Spiceline2 can then be passed in to individual instances using the SpiceLine attribute. The SpiceLine2 attribute automatically wraps onto another line if it is longer than 80 characters.
Impedance	Characteristic impedance for transmission lines.

Multi	This optional symbol attribute indicates a multiplier factor for multiple devices in parallel. It works for primitives and subcircuits with the exceptions of transmission lines, diodes, voltage sources, and current sources.
AreaS	This represents the source area for transistors. This is defined as a derived attribute allowing you to set its value as a function of other attributes.
AreaD	This represents the drain area for transistors. This is defined as a derived attribute allowing you to set its value as a function of other attributes.
PeriS	This represents the source perimeter for transistors. This is defined as a derived attribute allowing you to set its value as a function of other attributes.
PeriD	This represents the drain perimeter for transistors. This is defined as a derived attribute allowing you to set its value as a function of other attributes.
NRS	Equivalent number of squares of source diffusion in series with a transistor's source. This is used by SPICE to calculate source resistance.
NRD	Equivalent number of squares of drain diffusion in series with a transistor's source. This is used by SPICE to calculate drain resistance.
DefSubstrate	Default net to connect to bulk node on transistor. This is used if a transistor is shown with only three terminals, the fourth terminal, the bulk node, is defined by this attribute. The name referenced by this attribute must be a global net that exists on the schematic.

There are several other attributes used by the SPICE interface.

Value	Several primitive types use this to represent the basic value of the primitive. Elements using this attribute include: capacitor, resister, inductor, voltage source, current source, and transmission line.
TreeStop	If a symbol has TreeStop= *YES* it becomes a primitive. This is used to split large hierarchies. Refer to the *Hierarchy Navigator* section for more information on split hierarchies.
Width	Transistor width.
Length	Transistor length.
PinName	The pin name is used to identify pins for some elements.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attribute** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. The default values can be overridden for a particular instance of the symbol on the schematic by use of the **Add->Attribute** command of the Schematic Editor or Hierarchical Navigator.

Please refer to the *Entering Your Design* section in this manual for more details on attributes in the ECS environment.

Consult your *SPICE Manual* for a detailed explanation of the SPICE format.

Controlling The Netlist Extraction

The Hierarchical SPICE Netlist Processor is selected from the **Processes** menu of the Hierarchy Navigator. This processor creates a netlist file with the name *design_name.spi*, where *design_name* is your design's name. This file is a topological description of the hierarchical design and can be combined with user-supplied files that specify the input stimuli and monitor the output.

The Flat SPICE Netlist Processor can also be used. This processor also creates a netlist file with name *design_name.spi*. The netlist produced by the Flat SPICE netlister is in flattened (non-hierarchical) form. Only primitive symbols and the nets connecting them are listed. This version is useful when the design contains instance specific attribute overrides that must be taken into account.

When the netlist processor is selected from the Process menu of the Hierarchy Navigator a dialog box pops up with several program options. These options are:

Berkeley Spice	Uses the conventions of Berkeley Spice
PSpice	Uses the conventions of PSpice
HSpice	Uses the conventions of HSPICE
LVS	Uses the conventions of LVS

In addition to the buttons, there is a check box labelled *Node Names* which lists nets by name rather than by number. This box is selected by default for PSpice and HSpice.

The Hierarchical SPICE Netlister has three additional check boxes.

Sub Circuit	Produces a special version of the netlist for a sub-hierarchy when working with a split hierarchy.
Globals	Permits the use of the *.GLOBAL* instruction of HSpice. Any global nets are declared in a *.GLOBAL* statement. Global nets are not explicitly passed inside subcircuits through the node connectivity list in the subcircuit definition. If the option is not specified, no nets in the netlist are treated as global. In this case, all nets required in subcircuits are passed in through the node connectivity list in the subcircuit definition, including Vdd but not including Gnd.
Use Primitives	Determines whether the netlist expands block symbols having a Prefix specified as *X*. If the *Use Primitives* option is not specified, and the netlist processor encounters a symbol which has a Prefix specified as *X*, it searches for a sub-circuit SPICE file with the

same name as the symbol but with extension *.spi* and reads that file into the netlist without looking for any lower level schematics.

Both the flat and hierarchical spice netlist generators are modified to recognize the *X* prefix as a user defined model. In both cases, the *ModelName* symbol attribute is used as the model name if it is defined. The symbol's name is used if no model name is available.

If the *Use Primitives* option is selected, the netlist processor ignores the symbols that have a Prefix of *X* and codes the netlist using only the lowest level primitives.

Both the flat and hierarchical spice netlist generators are modified to recognize the *X* prefix as a user-defined model. In both cases, the *ModelName* symbol attribute is used as the model name if it is defined. The symbol's name is used if no model name is available.

SPICE Format Conversions

When the netlists are written with node numbers, a special preprocessor is invoked that converts names in a control file into the corresponding numbers. This allows you to generate the control file using meaningful node names and have ECS automatically convert the names to numbers for SPICE.

The Hierarchical SPICE netlister creates artificial nodes whenever it encounters unconnected pins on block symbols. Each instance of an unconnected pin gets its own unique node. In the node number format this node is a number larger than all the numbers on real nodes. In the node name format this node is assigned a name of the form *UNCnn* where *nn* is a unique number for each occurrence of an unconnected pin. Avoid assigning names of this format.

The Hierarchical SPICE netlister names instances with alphanumeric names with the Prefix attribute as the first character followed by the instance name. Unnamed instances and nets are given names beginning with *I_* and *N_* by the Hierarchy Navigator.

The Hierarchical SPICE netlister writes a hierarchical netlist. Because of this, any instance specific attribute overrides which are added by the Hierarchy Navigator do not appear in the netlist.

Working with the Hierarchical Netlist

It is sometimes necessary to associate the elements in the netlist with the instances on the schematic. When interpreting a particular SUBCKT in the SPICE netlist, first convert the element name to the original instance name by removing the Prefix. When using the normal option, with Node Numbers instead of names, the comments at the front of each SUBCKT description tell which nets represent which numbers.

To find the corresponding element on the schematic:

1. Use the **Push/Pop** command to find the schematic which represents the SUBCKT.
2. Select the **Misc->Query** command.
3. Type i= followed by the instance name from the netlist. The required element is highlighted.

The Node Names option makes it much easier to identify nets.

Working with the Flat Netlist

The Flat SPICE netlister produces files that can be quite large. Every block that is replicated in the hierarchy is replicated and assigned unique instance and net names.

The alphanumeric element names are formed by taking the Prefix (M or Q etc.) and concatenating it with the instance number, for example, M123, or Q17.

The flat netlist is very difficult to read. The **Misc->Query** command in the Navigator aids this. To find the net with a given node number:

1. Select the **Misc->Query** command.
2. Type the required node number on the prompt line. The **Query** command pushes or pops to the correct schematic and highlights the requested node.

To find the instance with a given element name:

1. Select the **Misc->Query** command.
2. Type *i=nnn* where *nnn* is the number portion of the element name. The Query command pushes or pops to the correct schematic and highlights the requested instance.

Preprocessing a Control File

Many versions of SPICE require that you specify nodes with numbers rather than names. This can be confusing if you are working with the numbered nodes for SPICE and the named nodes on your schematic. The SPICE netlister has a preprocessor that maps the names on your schematic into node numbers for SPICE. You can write the SPICE control files using the names on your schematics rather than numbers.

The preprocessing feature of the SPICE netlister works as follows:

1. Place all SPICE control instructions in a file having the file extension .*SPP*.
2. Place quotes around every net name.
3. Run the SPICE netlister with the node number option. The netlister creates a copy of the *SPP* file where the names in quotes are replaced with the corresponding numbers. This new file has the extension .*SPC*.

Attributes Required for SPICE Elements

Different SPICE elements require different attributes be assigned to them. This section lists the various attributes used with the different SPICE primitives.

Bipolar Junction Transistor

Prefix=	Q
SpiceModel=	name of Model for this device
Multi=	optional multiplication factor
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	first letter C for Collector
PinName=	first letter B for Base
PinName=	first letter E for Emitter
PinName=	first letter S for optional Substrate

A bipolar junction transistor is a four terminal device, with the fourth terminal being the substrate connection. Often transistor symbols are drawn showing only the base, collector, and emitter connections. The bulk is an understood connection to ground. SPICE needs to make this connection whether or not it is shown on the symbol. The DefSubstrate attribute on transistor symbols lets you connect the substrate node to any global net contained in the schematic.

MOSFET MESFET *and* JFET Devices

Prefix=	M for MOSFET, B for MESFET or J for JFET Devices
SpiceModel=	name of Model for this device
Width=	width in meters
Length=	Length in meters
AreaS=	Area of substrate
AreaD=	Area of Drain
PeriS=	Perimeter of substrate
PeriD=	Perimeter of drain
NRS=	number of squares of source diffusion
NRD=	number of squares of drain diffusion
Multi=	optional multiplication factor
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	first letter D for Drain
PinName=	first letter G for Gate
PinName=	first letter S for Source
PinName=	first letter B for optional Base (substrate)

A MOSFET is a four terminal device, with the fourth terminal being the substrate connection. Often MOSFET symbols are drawn showing only the drain, gate, and source connections. The bulk is an understood connection to ground or Vdd. The PSpice and Berkeley netlisters require you to provide the substrate connection either directly through the DefSubstrate attribute. The DefSubstrate attribute on MOSFET symbols lets you connect the substrate node to any global net contained in the schematic.

Transmission Line

Prefix=	T
Impedance=	characteristic impedance
Value=	transmission line delay
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	first two letters I1 for first input
PinName=	first two letters R1 for first reference node
PinName=	first two letters I2 for second input
PinName=	first two letters R2 for second reference node

Diode

Prefix=	D
SpiceModel=	model name
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	+ for cathode pin, anode name is optional

Capacitor

Prefix=	C
Value=	capacitance in farads
SpiceModel=	model name (PSpice, HSpice, and LVS)
Multi=	optional multiplication factor
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
Name=	optional name for both pins

Independent Sources

Prefix=	I for independent current source
Prefix=	V for independent voltage source
Value	voltage or current specification
SpiceLine=	optional voltage or current specification
SpiceLine2=	optional additional parameters
PinName=	+ for positive pin, negative name is optional

Linear Dependent Sources

Prefix=	E for Voltage controlled voltage source
Prefix=	G for Voltage controlled current source
Value	dependent gain for source
InitCond=	optional initial conditions
SpiceLine=	optional voltage or current specification

SpiceLine2=	optional additional parameters
PinName=	+ for positive pin, negative name is optional
PinName=	P for positive controlling node
PinName=	N for negative controlling node

Inductor

Prefix=	L
Value=	Inductance in Henries
SpiceModel=	model name (PSpice, HSpice, and LVS)
Multi=	optional multiplication factor
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	name is optional for both pins

Resistor

Prefix=	R
Value=	resistance
SpiceModel=	model name (PSpice, HSpice, and LVS)
Multi=	optional multiplication factor
SpiceLine=	optional additional parameters
SpiceLine2=	optional additional parameters
PinName=	optional name for both pins

Sub circuits (Hierarchical SPICE Only)

Multi=	optional multiplication factor
SpiceLine=	optional additional parameters on sub circuit instances
SpiceLine2=	optional parameter definition in sub circuit header

Examples

Two examples of SPICE specifications are presented. An independent voltage source shows the use of the SpiceLine attribute. A MOSFET example shows the typical attribute assignments for transistors in a CMOS IC design.

Independent Voltage Source

Symbol Attributes:

InstName=	IN7
Prefix=	V
SpiceLine=	0.001 AC 1 SIN (0 1 1MEG)

Pin Attributes:

PinName=	+ this is the pin attribute on the positive pin

The plus pin is connected to node 10 and the other pin is connected to GND.

Netlist Output:

```
VIN7 10 0 0.001 AC 1 SIN (0 1 1MEG)
```

MOSFET

Symbol Attributes:

InstName=	DIFF
Prefix=	M
SpiceModel=	P_PWELL
Width=	10E-6
Length=	1E-6
AreaS=	30E-12
AreaD=	30E-12
SpiceLine=	IC=0.2,3.7,-2.1

Pin Attributes:

PinName=	D *this pin corresponds to drain and is node 3*
PinName=	G *this pin corresponds to drain and is node 4*
PinName=	S *this pin corresponds to source and is node 5*
PinName=	B *this pin corresponds to substrate and is node 2*

Netlist Output:

```
MDIFF 3 4 5 2 P_PWELL L=1E-6 W=10E-6 AD=30E-12 AS=30E-12
                                    IC=0.2,3.7,-2.1
```

Error Messages

Missing or Invalid Prefix Code on Symbol Type ...
A primitive symbol did not have the Prefix defined.

Invalid Prefix on Element ...
The Prefix attribute is not one of the supported types.

Missing Symbol for ...
When using the Sub Circuit option, you must have a symbol for the root schematic.

Missing Sub Circuit File ...
If a symbol has TreeStop=Yes for split hierarchy, there must be a file with the given name and extension *.SPI* on the current directory.

GND not defined as a GlobalNet.
The **Controls->Global Nets** section of the ECS Preferences Editor must have an entry which includes GND.

Unconnected Pin on Bipolar Transistor ...
Bipolar transistors (Prefix=Q) must have nets wired to the Collector, Base and Emitter pins.

Unconnected Pin on Transistor ...
MOSFET (Prefix= M) , MESFET (Prefix= B) and JFET (Prefix= J) must have nets wired to the Source, Gate and Drain pins.

Unconnected Pin on Transmission Line ...
Transmission Lines (Prefix= T) must have nets wired to the pins labelled I1, I2, R1 and R2.

Unconnected Pin on Diode ...
Diodes (Prefix= D) must have nets wired to the two pins one of which is named with a plus sign ('+').

Unconnected Pin on Source ...
Independent Current Source (Prefix= I), Independent Voltage Source (Prefix= V), Voltage Controlled Voltage Source (Prefix= E) and Voltage Controlled Current Source (Prefix= G) must have nets wired to the two pins one of which is named with a plus sign ('+').

Unconnected Pin on ...
Resistor (Prefix= R), Inductor (Prefix= L) and Capacitor (Prefix= C) must have net wired to the two pins.

Failed to find Net (nn) ...
System Error, should not see this.

Missing or Unrecognized Default Substrate ... for ...
MOSFET transistors must have either four terminals or a valid DefaultSubstrate attribute. The DefaultSubstrate attribute must be the name of a global net and that net must exist on the schematic containing the transistor.

The Timemill Interface

The Engineering Capture System provides an interface to EPIC Design Technology's Timemill Simulator. This interface allows you to extract a netlist file for Timemill or Pathmill. The Hierarchy Navigator is used to examine the Pathmill results and the Waveform Tool is used to examine the Timemill results. Please refer to sections on the *Entering Your Design* and the *Waveform Tool* for operation of these two applications.

The Timemill Netlist Processor is selected from the **Processes** menu of the Hierarchy Navigator. This processor creates a netlist file with the name *design_name.ntl*, where *design_name* is your design's name. This file is a topological description of the hierarchical design. To simulate, it is combined with a user-supplied stimulus file which specifies the initial values, clock, and pattern definitions for the simulator. The pattern file is named *design_name.io*.

For critical path analysis, the netlist is combined with a user supplied static setup file which specifies the source and sink nodes. Any standard text editor can be used to create the stimulus file or static setup file.

Any instance specific attribute overrides to the Width or Length attributes on transistor elements are written to a separate configuration file named *design_name.ovr*. This file is used as input to Timemill and Pathmill.

If errors are encountered while writing the netlist, the processor invokes the editor on the netlist file. The editor is used to scroll through the netlist file and identify the errors. Errors are displayed on lines surrounded by asterisks. For example:

**** Missing Pin Named: EN ****

Configuration Required for Timemill

The Timemill netlister is accessed through the **Processes** menu of the Hierarchy Navigator. If you are using the configuration file provided by the CAD/CAM Group, the entry for the Timemill netlister probably already exists. Otherwise, you must use the ECS Preferences Editor to ensure that this entry is added. Refer to the *System Configuration* section for information on adding new applications to the **Processes** menu. The name of the Timemill netlister application is:

TimeNet	Timemill netlister

There are a number of optional flags that can be set in the ECS Preferences Editor for the Timemill netlister. These flags modify the operation of the netlisters to obtain specific results.

Option Flags

/A	Automatic mode, do not present dialog box
/M	Write sub circuit netlist

/P	Use lowest level primitives instead of models

The netlister brings up the option dialog box every time it is invoked if the *A* option is not specified. If the *A* option is specified the netlister does not display the dialog box but does use any other options specified on the command line. If other options are specified without the *A* option the dialog box pops up with the selected options as the defaults.

Necessary Attributes

There are three attributes that are reserved for use with the Timemill interface. These attributes are defined in the ECS Preferences Editor and are used to provide the necessary information for constructing the network description.

TimilModel	This symbol attribute identifies the primitive or model to be used to represent the symbol. Primitives are N, P, R, C. Anything else is treated as a model.
TimilExtra	This symbol attribute specifies the complete syntax of any parameters, states or initial values required by this type of symbol from blocks to transistors. If present, this string is appended to the network description after all the pins are listed. If a functional model is written which requires allocation of memory for states the symbol must contain the attribute: TimilExtra= (ST=*n*) where *n* is the number of states required.
TimilOrder	This pin attribute determines the order that the pins are listed in the netlist. For models, pins are identified by number and appear in the netlist in the same order as the numbers in the TimilOrder attribute. For primitives, the pins are identified by name. If the PinName is different than the name required to identify the pin for Timemill then the TimilOrder can be used as an alternate name. The netlist processor checks the TimilOrder before looking at the PinName.

There are several other attributes which are used by the Timemill interface.

TreeStop	If a symbol has TreeStop=*YES* it becomes a primitive. This is used to split large hierarchies. Refer to the *Hierarchy Navigator* section for more information on split hierarchies.
Value	The Value attribute specifies capacitance values in nanofarads on capacitors and resistance values in ohms on resistors.
Width	The Width attribute specifies the channel width of transistors in microns.
Length	The Length attribute specifies the channel length of transistors in microns.
Polarity	The Polarity attribute determines which pins are output pins.
PinName	The PinName attribute identifies pins for the primitives if the TimilOrder attribute does not specify the name.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attribute** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. The default values can be overridden for a particular instance of the symbol on the schematic by use of the **Add->Attribute** command of the Schematic Editor or Hierarchical Navigator.

Refer to the *Entering Your Design* section in this manual for more details on attributes in the ECS environment.

Controlling The Netlist Extraction

When the netlist processor is selected from the **Processes** menu of the Hierarchy Navigator a dialog box pops up with several program options.

Netlist	writes the netlist only
Sub Circuit	writes a special version of the netlist for a sub-hierarchy when working with a split hierarchy

In addition to the buttons, there is a check box labelled *Use Primitives*. Normally when the netlist processor encounters a symbol that has the TimilModel attribute specified, it codes that model into the netlist without looking for any lower level schematics. If the *Use Primitives* option is selected, the netlist processor ignores the TimilModel on higher level blocks and codes the netlist using only the lowest level primitives. This feature makes it possible to switch between simulating at the functional level and the switch level without changing anything on the schematic.

For example, a design contains a schematic with a symbol called *NAND2* and the TimilModel attribute of this symbol is *NAND.* An underlying schematic exists for the symbol *NAND2* containing transistor symbols *NMOS* and *PMOS* (TimilModel= *N* and *P*). The normal option codes the netlist with *NAND* without any reference to the underlying schematic. With the *Use Primitives* option, *NAND2* is coded as a sub-circuit using n-channel and p-channel transistor switch level elements.

Timemill Format Conversions

The Timemill netlister assigns names to nodes whenever it encounters unconnected pins on block symbols. The format of the name is *UNCnn* where *nn* is a unique number for each occurrence of an unconnected pin. Do not assign names using this format in the schematic.

Building Timemill Symbol Libraries

Timemill has several built-in primitive functions that are used in simulation. Several attributes in these primitives must take on prescribed values.

The following list gives the attributes needed for various Timemill primitives.

CMOS Transistors

TimilModel= either N or P

Width= channel width, default scale factor is microns.

Length= channel length, default scale factor is microns.

TimilOrder= or PinName= S, G, D

Resistor

TimilModel= R

Value= resistance in Ohms, default scale factor is 1.

TimilOrder= or Name= S, D

Capacitor

TimilModel= C

Value= capacitance, default scale factor is nanofarads.

This symbol should only have one pin or if it has two pins one of them must be connected to GND.

Models

TimilModel= name of the model.

TimilExtra= optional specification of states and initial values

Symbols that represent functional models require that the pin output order be specified. This is accomplished using the TimilOrder attribute. It indicates the numeric order that the pins are to be output. The pins are grouped according to Polarity.

Examples

Two examples of Timemill models are presented. A J-K flip flop shows the use of the TimilOrder and TimilExtra attributes. The second example shows how buses can be automatically ordered.

Functional Model for J-K Flip Flop:

Symbol Attributes:

TimilModel= xjkff
TimilExtra= (st=2)

Pin Attributes:

PinName	TimilOrder	Polarity
J	2	In
K	4	In
CLK	3	In

SET	5	In
CLR	1	In
Q	1	Out
QN	2	Out

In the above example, the nets connected to the input pins are listed in the order: CLR, J, CLK, K, and SET. The nets connected to the output pins are listed in the order: Q and QN. This functional model requires two states to remember Q and QN so the TimilExtra attribute is specified to allow for these two states.

Dummy Symbol Using Bus

Symbol Attributes

TimilModel= DUMMY

Pin Attributes:

PinName	TimilOrder	Polarity
OUT		Out
CLOCK	1	In
DATA[0;3]	2	In
RESET	5	In

In the above example, the nets connected to the input pins are listed as follows: CLOCK, DATA[0], DATA[1], DATA[2], DATA[3], RESET. The output pin did not require TimilOrder to be specified since there was only one output pin. If a pin represents a bus pin (DATA[0;3]) the numbering sequence should treat that pin as a sequence of numbers.

Error Messages

No Model Defined for ...
A primitive symbol did not have the TimilModel defined.

Missing Symbol for ...
The Sub Circuit option requires a symbol for the root schematic.

Missing Sub Circuit File ...
If a symbol has TreeStop=Yes for split hierarchy, there must be a file with the given name and extension *.ntl* in the current directory or in one of the project or model directories. Run the Hierarchy Navigator on the sub-hierarchy and use the Macro option on the Timemill netlister to create this file.

Missing Pin Named: ...
A primitive symbol is expected to have a pin with the given PinName or TimilOrder attribute.

Missing Pin Order for TimeMill On ...
Models require that all pins of a given Polarity have TimilOrder specified as a number indicating the numerical order for the pin to be listed. If order doesn't matter, then all the pins of the same Polarity should have the TimilOrder left blank.

Incorrect Pin Ordering in ...
This model has a gap in the sequence of numbers in the TimilOrder attribute.

Unconnected Capacitor Pin On: ...
The capacitor pin must be connected.

Capacitance Not to GND
If a Capacitor symbol has two pins, one of the pins must be connected to GND.

Missing or Invalid VALUE (Capacitance) for ...
The Value attribute must be set to the capacitance for this node in nanofarads.

Missing or Invalid VALUE (Resistance) for ...
The Value attribute must be set to the resistance in Ohms for this resistor.

Missing or Invalid WIDTH for ...
The Width attribute must be set to the channel width of this transistor in microns.

Missing or Invalid LENGTH for ...
The Length attribute must be set to the channel length of this transistor in microns.

Verilog Interface

The Engineering Capture System provides the means of including the Verilog Simulator in the design environment. This interface is presently operational on the UNIX and Macintosh versions of ECS.

The interface between ECS and Verilog consists of three parts:

Netlist Extraction	The Engineering Capture System can extract the structural module description from the schematic. It also has the ability to extract a template description from the symbol which represents a behavioral model.
Simulator Control	The Verilog simulator can be launched and controlled from the Navigator. Various commands can be defined in which the command is selected from a menu and the elements (such as nets) are selected graphically from the schematic.
Output Viewing	The output of the simulator can be viewed graphically with the Waveform Tool. The simulation results can be viewed as the simulation progresses (dynamic mode) or after the simulation is completed (static mode) by using a history file. The Waveform Tool can either run alone (stand-alone) or in conjunction with the Navigator. In the latter case, the schematic can be backannotated with the states of signals as defined in the simulator.

The necessary code is in the directory *interfaces/verilog* as supplied on the UNIX distribution tape. A test example is included in the *interfaces/veritest* directory. A sample verilog.ini file is included in the data directory on the distribution media.

Integrating Verilog

The integration of Verilog into the ECS environment involves adding a few commands to Verilog so that it can communicate with ECS. The commands are defined by modifying the *veriuser.c* template that is provided with your Verilog release tape. The modified program is then compiled and linked with the Verilog library, the *ecs_pli* and the *simwave* object files to produce a version of Verilog that connects to the Waveform Viewer.

The following steps are to integrate Verilog version **1.6** into the ECS environment (see the *interface/verilog/ver1.5* directory for instructions on integrating Verilog version 1.5):

1. Run the *vconfig* program supplied with the Verilog release to create a script to make the Verilog executable. The default options are adequate except the following:

   A Answer no to the question

   Do you want to include graphics in this VERILOG?

   You do not need to include Verilog graphics to run the ECS Waveform Viewer interactively, but if you need Verilog graphics for some other purpose, answer yes to this question and go through step 3. Note, however, that you should not use ECS waves and gr_waves at the same time.

   B Specify the full path name to the *veriuser1.6.c* file provided with the ECS system in the *interface/verilog/ver1.6* directory in place of the template file provided with Verilog.

   C Specify the full path name to the *ecs_pli1.6.o* file provided with the ECS system in the *interface/verilog/ver1.6* directory to be added to the link as additional user routines.

   D Specify the full path name to the *simwave.o* file provided with the ECS system in the *interface/verilog/ver1.6* directory to be added to the link as additional user routines. Alternatively, you can use the sample script file (called *cr_vlog.1.6*) supplied with the ECS release by changing the paths for the files mentioned above. The sample script file does not link the Verilog graphics objects.

2. If you have used the *vconfig* program to generate the script file you might want to change the name of the resulting Verilog executable to avoid name conflicts by modifying the **cc** line of the script file.

   The sample script file provided with the ECS release uses *verilog1.6* for the Verilog executable file name. However, you need to substitute ECS_PATH with the path of the ECS directory which contains the interface directory.

3. A further modification to the script file is necessary *if* you included Verilog graphics *and if* you will be running ECS and Verilog under Motif (or any other X11 based environment) rather than SunView. Using a text editor change the line that contains:

   ```
   -lsuntool -lsunwindow -lpixrect -lm
   ```

   to:

   ```
   -lX11 -lm
   ```

   Make sure that you have the LD_LIBRARY_PATH environment variable set appropriately so that the compiler can find the libraries it needs.

4. Run the script to create the Verilog program that links to the ECS environment. Note that you would need to be in the Verilog release directory to do this since all the Verilog object file references are relative to this directory.

You can include other user commands into the *veriuser1.6.c* file. See the Verilog documentation for further information.

Setting up the Verilog Environment in ECS

The Verilog Netlister is invoked from the **Simulator** menu of the Hierarchy Navigator for hierarchical netlists. The Verilog netlister creates a top-level netlist file with the name *design_name.v* where *design_name* is the top-level name of your design hierarchy, and individual netlist files for sub-level blocks. The Verilog netlister is also invoked from the **Tools** menu of the Schematic Editor and the **Tools** menu of the Symbol Editor to create Verilog netlists of individual schematics or modules.

Setting up the ecs.ini File

To run the interface, you must enter the following information in the **ecs.ini** file.

The Simulator parameter in the Controls section of the **ecs.ini** file enables the Simulator menu of the Hierarchy Navigator with the name and options of the selected simulator. Type in **verilog** for the Simulator parameter.

The remaining parameters for the verilog simulation environment are kept in the **verilog.ini** file in the *$ECS_ROOT/data* directory. Each design can also have a separate **verilog.ini** file in its local directory.

Adding a Tools Menu Entry to the Schematic Editor

The Verilog netlister can be run as a **Tool** from the Schematic Editor, independent of the Simulator parameter. The program generates a module netlist of the schematic block. A symbol for the schematic has to exist in order for the netlist to be written since the module declaration is extracted directly from symbol files. This gives the user independent control of module information and its underlying netlist.

To create a **Tools** menu option for Verilog in the Schematic Editor, use the following procedure:

1. Select the **ecs.ini** Editor from the ECS Executive.
2. Select the **Tools->Schematic Tools** menu.
3. Create the following menu entry:

   Verilog Netlist

4. Add the following command line entry:

   vericode

Any required options are added after *vericode* on the command line.

Adding a Tools Menu Entry to the Symbol Editor

The Verilog netlister can be run as a **Tool** from the Symbol Editor, independent of the Simulator parameter. The program generates a module statement for the symbol being edited.

To create a **Tools** menu option for Verilog in the Symbol Editor, use the following procedure:

1. Select the **ecs.ini** Editor from the ECS Executive.
2. Select the **Tools->Symbol Tools** menu.
3. Create the following menu entry:
   Verilog Netlist
4. Add the following command line entry:
   vericode

Any required options are added after *vericode* on the command line.

Command Line Options

The following options are used with the vericode netlister:

-n	Show in Selected Editor
-NETS	Coerce Net & Pin Names to Upper Case
-nets	Coerce Net & Pin Names to Lower Case
-INST	Coerce Instance Names to Upper Case
-inst	Coerce Instance Names to Lower Case
-model	Coerce VeriModel Names to Lower Case
-noprim	Code Primitive Types as Modules

The selected editor is specified with the Editor option in the Controls section of the **ecs.ini** file.

Necessary Attributes

There are five attributes that are reserved for use with the Verilog interface. These attributes are defined in the **ecs.ini** file and are used to provide the necessary information for constructing the network description.

VeriModel
This symbol attribute identifies the primitive or model used to represent the symbol. It is used in conjunction with the VeriName pin attribute to generate the instantiation line for primitive and model blocks. Any name which is not one of the primitives is treated as a model. This attribute cannot be overriden in the schematic editor or the Hierarchy Navigator on an instance basis because the symbol files are used to determine the Verilog model names. Thus any such change will not have any effect

on the Verilog netlist that is created. The remainder of the attribute (after the model or primitive name) is copied into the network description without interpretation. Therefore, if a strength qualifier is desired on a primitive, it can be appended to the primitive name in this attribute. If the -model option is used, the Verilog model name is coerced to lower case; if it not used the Verimodel attribute is netlisted with whatever case is specified on the symbol definition.

VeriName
The Verimodel and Veriname attributes are useful only for blocks with hand-coded models. These attributes are used to manually match the model and pin names of the hand-coded models with the model and pin names of the symbols that represent these models. If present, this pin attribute is used to determine the pin names that are netlisted in the instantiation line for models *only if* they have a VeriModel attribute assigned. If not present, the ECS PinName attribute is used to determine the pin names. This attribute should be used if the PinName attribute is different than the name required to identify the pin in the Verilog netlist.

In cases where a pin must be ignored during netlisting (hidden pins are one example), it can be given a name beginning with a double minus ("--") using the VeriName attribute. Note however, that the Verimodel attribute must be assigned a value for the VeriName attribute to be used.

VeriStrength
This symbol attribute specifies the strength for the outputs on this gate. If present, this string is inserted into the network description between the VeriModel and the VeriTimes.

VeriTimes
This symbol attribute specifies the delay for this gate. If present, this string is inserted into the network description between the VeriModel and the pins.

VeriComp
This attribute causes a *defparam* line to be generated as needed by the Logic Automation Incorporated libraries. If a VeriComp attribute is specified, the line:

```
    defparam "instance_name" COMPONENT="VeriComp_value"
```

precedes the instance line.

There are several other attributes used by the Verilog interface.

Polarity
The Polarity attribute is used by the Verilog netlister to determine which pins are the output pins for several of the primitive types. Legal values are INPUT, OUTPUT, BIDIR.

PinName
The PinName is used to identify pins for some primitives and for all models if the VeriName attribute is not present or is not used.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attribute** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. Note that the Verilog netlister writes a hierarchical netlist. Because of this, any instance specific attribute overrides added by the Hierarchy Navigator do not appear in the netlist. Please refer to the *Attribute* section in this manual for more details on attributes in the ECS environment.

Setting up the verilog.ini file

The Hierarchy Navigator can be configured to control the Verilog simulator. The first step is to specify the *Simulator* parameter to be verilog using the ecs.ini editor. This causes the Navigator to read the *verilog.ini* file which must be in either the local directory or in the ECS_ROOT/data directory.

The **verilog.ini** file contains all the necessary commands and directives to control the Verilog simulator. The contents of this file have changed for the 2.4 version of the ECS software. The most important changes are:

Syntax of some commands that are passed to Verilog

Addition of Verilog wire type controls

Full bus probing capability

A sample **verilog.ini** file is shown below. A copy of this file is provided with the ECS release tape for your convenience, and it is recommended that you use this file until you become more familiar with the ECS simulation environment (please refer to the *Simulation Environment* section in this manual for more details on this topic).

```
[ModelBuilder]
Method = Schematic/Symbol
Process = vericode %
RootProcess =
Extension =.v
Path =
ModelAttribute = 20
NetAttribute = 10

[SimTitle]
  title = Verilog

[SimControls]
  Command = verilog -pXXXX -s %%.v -y .../lib
+libext+.v
  Waves = waves -nav -sim@ %
  First = $ecs_init ("@");
  Setup = cmdtool notepad %.v
  FlattenBuses = No
Reload = Yes
  MaxLine = 40
```

```
    RootPrefix = %
    RootIOprefix = %
    InstPrefix = %
    SubPrefix = %
    HierChar = .
    Terminator = ;
    Separator = ,
    Prompt = >
    ExitMsg = $finish;

  [SimButtons]
    Run = .
    Stop = ^C$ecs_stop
    Delta = #% $ecs_stop;.

  [SimMenu]
    Monitor = $ecs_waves ( "%n" )
    Force = force %n = %{ 0 1 }
    Release = release %n
    Finish = $finish

  [NetAttributes]
  WIRE =
    WAND = WAND
    WOR  = WOR
    REG  = REG
    TRI  = TRI
    TRIAND = TRIAND
    TRIOR = TRIOR
    TRI1 = TRI1
    TRI0 = TRI0
  TRIREG(S) = TRIREG(S)
    TRIREG(M) = TRIREG(M)
    TRIREG(L) = TRIREG(L)
    SUPPLY0 = SUPPLY0
    SUPPLY1 = SUPPLY1
```

Verilog Command Line

Generally, the -f option of Verilog is used to include most of the command line information that is needed by Verilog but the commands can also be included in the *verilog.ini* file itself. The command line typically contains:

The command to start Verilog, i.e.,

```
.../verilog
```

or if using a remote host,

```
rsh remote_host .../verilog
```

A password option,

```
-p1234abcd
```

The option to include a commands file, i.e.,

```
-f .../commands_file
```

The option to run Verilog in interactive mode:

```
-s
```

The netlist file for the design to be simulated:

```
%%.v
```

and the option to include a library path to be scanned:

```
-y .../lib +libext+.v
```

The paths (.../) to the commands file and the netlist files must be from the view of the host running Verilog. The directories containing these files must therefore be mounted on the host. See your network administrator for assistance in determining the correct way to access these files.

Building Verilog Symbol Libraries

Verilog has several built in primitive functions that are used in simulation. Several attributes in these primitives must take on prescribed values.

The following list gives the attributes needed for various Verilog primitives.

Combinatorial Gates
  VeriModel= one of AND NAND OR NOR XOR XNOR BUF NOT
  Polarity= OUT on the output pins
  Polarity= IN on the input pins

Bidirectional Pass Gates
  VeriModel= one of TRAN RTRAN
  Polarity= BI on both pins

Tri-State Bidirectional Gates
  VeriModel= one of TRANIF0 TRANIF1 RTRANIF0 RTRANIF1
  Polarity= BI on the two main pins
  Polarity= IN on the control pin

Tri-State Drivers
  VeriModel= one of BUFIF0 BUFIF1 NOTIF0 NOTIF1
  Polarity= OUT on the output pins
  VeriName= or PinName= IN on the input
  VeriName= or PinName= EN on the enable pin

MOS Gates

VeriModel= one of NMOS PMOS RNMOS RPMOS

VeriName= or PinName= D, G, S

CMOS Gates

VeriModel= one of CMOS RCMOS

VeriName= or PinName= D, GN, GP, S

Pullup & Pulldown Gates

VeriModel= one of PULLUP PULLDOWN

Polarity= OUT on the pin

Models -- User Defined Primitives

VeriModel= name of the primitive.

Polarity= OUT on the output pin

Polarity= IN on the input pins

VeriName= User defined name

The pin names of the user defined primitives (or models) are netlisted in alphabetical order. The pin name matching is done explicitly.

Example

The following example shows a model for dummy symbol with a bus:

Symbol Attributes:

VeriModel= DUMMY

Pins Attributes:

Pin Name	VeriName	Polarity
OUT		Out
CLOCK		In
DATA[0;3]		In
RESET		In

In the above example the pins are listed as follows:

CLOCK, DATA[0:3], OUT, RESET

Netlist Extraction

The **Code Verilog** commands in the Hierarchy Navigator are enabled by setting the *Simulator* option to *verilog* using the ecs.ini editor. A file named *verilog.ini* must exist in either the local directory or in the *ECS_ROOT/data* directory.

Selecting **Code Verilog** in the Simulator menu of the Hierarchy Navigator creates a set of structural module definitions for each schematic linked into the hierarchy including the top level.

Selecting **Verilog Netlist** in the Tools Menu of the Schematic Editor creates a complete structural module definition from the schematic and the associated symbol file. Note that a symbol for the schematic must exit so that the module netlist can be completed since the module declaration of the block is extracted from the symbol file.

Selecting **Verilog Netlist** in the Tools Menu of the Symbol Editor creates a template module definition from the symbol *only if* an associated schematic is not available. The intent is that the user then completes the behavioral description in text mode.

All three versions of this function create (or update) files with the base name of the schematic or symbol and a *.v* suffix.

Controlling The Netlist Extraction

The following parameters and controls allow you to tailor the netlist extraction for your own needs.

VeriModel Attribute

The VeriModel symbol attribute is the most useful way to control Verilog Netlist extraction. Normally when the netlist processor encounters a symbol which has the VeriModel attribute specified, it codes that model into the netlist with the specified VeriName pin attributes, if any, without looking for any lower level schematics. If the VeriModel attribute is not specified, the netlister would ignore the VeriName pin attributes of that symbol and code the lower level schematics associated with that symbol. Note that the VeriModel and VeriName attributes are only used to code the module instantiations (i.e. calls to modules), and not module declarations.

-noprims Command Line Option

The *-noprims* command line option of the netlister causes the it to skip the test for the Verilog primitive names and to code all instances as normal modules. This results in further coding of Verilog primitives.

For example, a design contains a schematic with a symbol called *NAND2* and the VeriModel attribute of this symbol is *NAND*. An underlying schematic exists for the symbol *NAND2* containing transistor symbols *NMOS* and *PMOS* (VeriModel = *NMOS* and *PMOS*). Normally, *NAND2* is coded with the NAND primitive name without any reference to the underlying schematic. With the *-noprims* option, *NAND2* is coded as a sub-circuit using n-channel and p-channel transistor switch level elements.

Existing Verilog Files

In some cases, template module definitions are generated from symbol files which are then completed with user defined Verilog behavioral models or stimulus. In order to preserve this user defined data the next time the file is updated (for example after a change in the number of pins of the symbol) the Verilog netlister only replaces the header comment, the module statement and any lines that identify the polarity of the pins in the module statement. Note that if there is a schematic associated with the symbol under consideration, a completely new file is generated each time, and no manually entered data is preserved.

Pin Ordering

The Verilog netlister uses explicit naming for instantiating modules. The symbol files are used to extract pin names for module declaration lines as well as instantiation lines. In both cases, the pins are sorted alphabetically without any regard to their polarity (input, output, inout). In the case of module instantiations, each pin is matched with the connected net before the sort.

Vectors

Vectors that have their bits spread over many pins are regrouped together. For example for the pin assignments below:

```
|
|----o DD[0:3]
|----o A,B,DD[4:5]
|----o DD[6]
|----o DD[8]
|
```

the vector DD[0:8] will be used as the port (note that bit DD[7] is generated). It is the user's responsibility to insure that all of the bits of a vector are coded with the same polarity. The netlister randomly selects one of the pins to determine the polarity of the vector.

For module instantiations, vector pins are coded with the vector name and the list of signals connecting to the vector. If any bits are not connected, either because of an unconnected pin or an embedded bit that was not on a pin, an extra comma is inserted to the list of nets. When a list of nets on a pin includes one or more commas, it is surrounded by {}'s inside the ()'s. In case of bus to bus pin count mismatches, if too many signals appear on a bus pin, the extra signals are dropped; if too few signals are connected, the remaining bits become unconnected.

Running Verilog

To simulate, the netlists are first combined with a user-supplied stimulus file which specifies the initial values, clock, and pattern definitions for the simulator. Then the Verilog Simulator is started from the **Simulator** menu of the Hierarchy navigator which brings up the simulator window, as well as the Dynamic Waveform Tool for viewing the simulation results dynamically:

1. Start the Navigator on the top level of your design. **Verilog** appears on the menu bar. If **Verilog** does not appear, the setup is not correct or the

verilog.ini file was not found in either the local directory or in the $ECS_ROOT/data directory. Correct and restart the Navigator.

2. Select the **Start Simulator** command in the **Verilog** menu. The Simulator Control window opens showing the 3 buttons (Run, Stop and Delta). Below the buttons is a window showing the command:

   ```
   $ecs_init (192.9.200.xxx);
   ```

   The Verilog sign-on messages appear shortly. If there are any problems starting verilog, they are reported. Eventually, you see two Verilog prompts:

   ```
   C1 > C2 >.
   ```

   If the WAVES option was requested, the waves window opens at this time.

You are now prepared to run Verilog. Notice that the menu changes to show the commands specified in the *verilog.ini* file. Use the Navigator menu and the control buttons on the simulator window to control the Verilog program. Commands that are not included in the menu can be entered directly into Verilog by moving the mouse to the control window and typing the command.

Test your setup by running the small example which is included in the *interface/verilog/veritest* directory.

VHDL Interface

The ECS creates VHDL netlists incorporating your user defined models. The connectivity of a schematic is correctly mapped into a structural representation. VHDL model templates are automatically created from ECS symbol descriptions. The Hierarchy Navigator can be used to interactively control a VHDL simulation. The Waveform Tool can be used to view the results of a VHDL simulation.

The VHDL netlister can be accessed from the Symbol Editor from the **Tools** menu. You must add a command line using the **Tools->Symbol Tools** section of the **ecs.ini** Editor that contains the module name:

    vhdl

When called from the Symbol Editor, the VHDL netlister creates a template with all the input, output, and bidirectional ports predefined based on the symbol pins. A section of the template is marked for you to insert any required functional model code. If you run the VHDL netlister after a vhdl file has been created, any functional model definition is left untouched. This means you can change the number of pins on the symbol and the functional portion of your model is not overwritten when you netlist.

The VHDL netlister can be accessed from the Schematic Editor from the **Tools** menu. You must add a command line using the **Tools->Schematic Tools** section of the **ecs.ini** Editor that contains the module name:

    vhdl

When called from the Schematic Editor, the VHDL netlister creates a structural netlist.

The VHDL netlister can be accessed from the Hierarchy Navigator from the **Tools** menu or the **Simulator** menu. When accessing from the **Tools** menu, you must add a command line using the **Tools->Navigator Tools** section of the **ecs.ini** Editor that contains the module name:

    vhdl

When called from the Hierarchy Navigator, the VHDL netlister creates examines each primitive element in the design to see if a vhdl file exists describing the primitive. If a vhdl description does not exist, one is created. The same is true for all the schematics contained in the design.

All VHDL files created by ECS are given the file extension *.vhd.*

The Simulation interface section contains information describing how simulators are integrated into the ECS environment.

Design Analysis Tools

This section contains information describing the Design Analysis Tools (DAT) package. This software works in conjunction with the Engineering Capture System (ECS) to accomplish three major objectives.

- Identify errors in your design as early as possible.
- Provide the ability to calculate an attribute's value based on other attributes.
- Provide extended Printed Circuit Board (PCB) functionality in the packaging area.

The major topics covered in this section are:

- Installation
- Proper ECS configuration using the ECS Preferences Editor
- Electrical rules checking including fanout analysis
- Printed circuit board checking including package checking
- Packaging of PCB elements
- Report viewer that aids location of design errors
- Derived attributes
- Critical Path Viewer

Design Analysis Tools

The Design Analysis Tools (DAT) package provides tools that aid design in three main areas:

- tools to help identify errors in complex circuits
- tools to help with some areas of PCB design such as autopackaging
- functionality for derived attributes

The following tools or functions are included in the DAT.

Electrical Rules Checking	The Electrical Rules Checker checks basic connectivity and current loading or fanout analysis on a completed design. The checker shortens design turn-around time by eliminating many errors prior to simulation.
Printed Circuit Board Checker	The Printed Circuit Board Checker performs a packaging check on schematics used for Printed Circuit Board technology. The check ensures that the assignment of Gates to packages is correct.
View Report	This command opens a specified file and displays the contents of the file in a list box. If any instances, nets, pins or symbol types are referenced on a given line in the list box, then clicking on that line causes the display to jump to that portion of the schematic containing the referenced item. You can quickly locate error conditions in this manner.
Packager	The packager supports a number of commands for automatically assigning reference designators and pin numbers to PCB gate, component and pin symbols.
Derived Attributes	Derived attributes take their value from other attributes on symbols, symbol pins or nets connected to symbol pins. You can pass parameters up and down a design hierarchy using derived attributes and you perform calculations using derived attributes. Derived attributes allow you to analyze your design using spread sheet techniques, changing a parameter value in one place and watching the effects of the change ripple throughout your whole design.

Checks are also done on the symbols and schematics in the Symbol Editor and the Schematic Editor using the **Misc->Check** command. See the *Command Reference* section for more information.

Configuring Your System

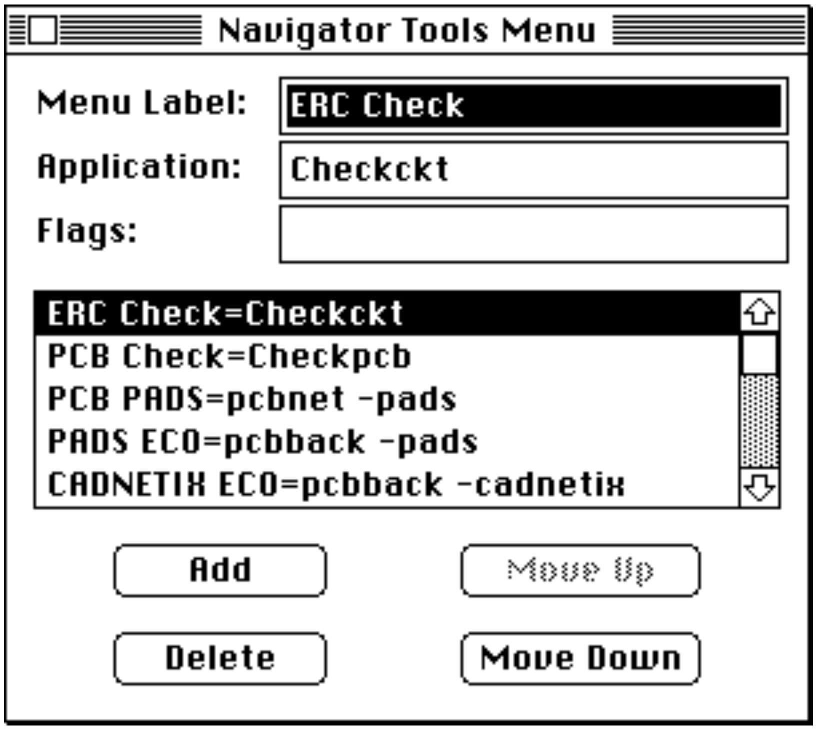
To run the tools in the DAT package, you must have purchased the DAT module. The *ecs.cfg* license file supplied with your software enables the use of DAT tools. The **View Report** command, **Packager** commands and derived attributes are automatically activated when you purchase the DAT. You must explicitly add the Electrical Rules Checker and Printed Circuit Board Checker to the **Tools** menu using the procedure below.

Adding a Menu Entry to the Hierarchy Navigator

To create a **Tools** menu option for the Electrical Rules Checker in the Hierarchy Navigator, use the following procedure.

1. Run the ECS Preferences Editor.
2. Open your configuration file. This is typically called **ecs.ini** and stored in the main ECS folder but you can call the file any name and store it in any folder.
3. Select the **Tools->Navigator Tools** menu. The dialog box in Figure 7-1 is shown.
4. Create the following *Menu Label*:
   - ERC Check
5. Add the following entry for *Application*:
   - CheckCkt

6. Any required options are added in the *Flags* field.



Menu Label: ERC Check
Application: Checkckt
Flags:

ERC Check=Checkckt
PCB Check=Checkpcb
PCB PADS=pcbnet -pads
PADS ECO=pcbback -pads
CADNETIX ECO=pcbback -cadnetix

Figure 7-1 ***Tools->Navigator Tools*** *Dialog Box of ECS Preferences Editor*

Electrical Rules Checker Flags

/A	automatic mode, do not present dialog box
/W	Issue warnings about nets with no load
/L	Perform fanout analysis
/H	Perform High/Low current loading analysis

The checker displays an option dialog box every time it is invoked if the *A* flag is not specified. If other options are specified without the *A* flag, the dialog box pops up with the selected options as the defaults. If the *A* flag is specified, the checker does not display the dialog box but does use any other flags specified.

The PCB Checker is added in the same way except the **Tools** menu entry is *PCB Check* and the command line is *CheckPCB*.

ERC and PCB Checkers

The following description describes the setup and operation of the Electrical Rules Checker and the Printed Circuit Board Checker.

Building DAT Compatible Libraries

All of the primitive elements in a PCB library must be either Pin, Gate or Component symbols. The primitives in an integrated circuit library must be Cell symbols. Many of the attributes that need to be assigned are common to both PCB and IC applications. The attributes described below need to be properly assigned to the elements in a library.

There are eight attributes that are used by both the ERC and PCB Checkers. These attributes are defined using the ECS Preferences Editor and provide the information necessary to perform many of the required checks.

Polarity	This pin attribute determines which pins are output pins and which pins are input pins. If the Polarity is not specified for a pin, Input is assumed. The possible values for Polarity are, Input, Output and Bidirectional. Only the first character is needed to set the Polarity (I, O, or B).
FanIn	This pin attribute represents the load on a pin and is used for fanout analysis. Fanout analysis checks to see that the FanOut driving a net is greater than the sum of all FanIns attached to the same net. Because this is a relative measurement, the units for FanIn are arbitrary as long as they are consistent with the units for FanOut.
FanOut	This pin attribute represents the driving capability of a pin and is used for fanout analysis. Fanout analysis checks to see that the FanOut attached to a net is greater than the sum of all FanIns attached to the same net. Because this is a relative measurement, the units for FanOut are arbitrary as long as they are consistent with the units for FanIn.
LoadLow	This pin attribute is used for the Hi/Lo Loading Analysis. The LoadLow represents the load of the input pin when the net is in the logic low state. Because the analysis compares LoadLow to DriveLow, the units are arbitrary as long as they are consistent throughout the library.
DriveLow	This pin attribute is used for the Hi/Lo Loading Analysis. The DriveLow represents the drive of the output pin when the net is in the logic low state. Because the analysis compares LoadLow to DriveLow, the units are arbitrary as long as they are consistent throughout the library.
LoadHigh	This pin attribute is used for the Hi/Lo Loading Analysis. The LoadHigh represents the Load of the input pin when the net is in the logic high state. Because the analysis compares LoadHigh to DriveHigh, the units are arbitrary as long as they are consistent throughout the library.
DriveHigh	This pin attribute is used for the Hi/Lo Loading Analysis. The DriveHigh represents the drive of the output pin when the net is in the logic high state. Because the analysis compares LoadHigh to DriveHigh, the units are arbitrary as long as they are consistent throughout the library.
WireOr	This pin attribute indicates that a pin has tri-state, open-collector, or wired-or outputs. The values for this attribute are *Yes* for wired-or, *O* for open-collector or *Tri* for tri-state. Only the first character is used.

Output pins with WireOr specified as tri-state should have both a Load and a Drive. The Load represents the Load of the pin while in the tri-state condition.

The DAT currently does not distinguish between wired-or and open-collector connections. The truth table of error conditions shown on the next page assumes two gates A and B have their outputs tied together as in Figure 7-2.

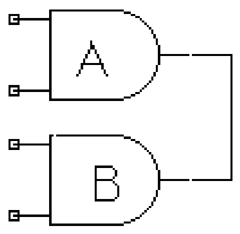


Figure 7-2 Two Gates with Outputs Connected Together.

A	B	Legal Connection/Error
Tri	Tri	OK
Tri	OC	Error
OC	Tri	Error
OC	OC	OK
OC	wire-ed OR not specified	Error
wire-ed OR not specified	wire-ed OR not specified	Error
wire-ed OR not specified	OC	Error
Wire-ed OR not specified	Tri	Error
Tri	Wire-ed OR not specified	Error

The following three attributes are used for the PCB Checker only.

RefDes	This symbol attribute represents the reference designator of a part on a printed circuit schematic. This attribute must be unique for component symbols. Gates which are part of the same package have the same RefDes. Symbols of type PIN, used for edge connectors, can have the same RefDes if they belong to the same connector.
PinNumber	This pin attribute represents the number of the physical pin on the package. For Components, this attribute is a single number. When a Gate symbol is created, its PinNumber attribute is set to a list of numbers. This list represents the number of the pin in each of its possible sections.

When the Gate is assigned to a package with the **Package->Pin Numbers** command of the Schematic Editor or the **Misc->Pin Numbers** command of the Hierarchy Navigator, the PinNumber attribute takes on the actual number of the pin for the section it represents. Some packages have Gates which have common clock or

	enable signals. In this case, the PinNumber lists the number of the common pin repeatedly, once for each section.
PinGroup	This pin attribute indicates that certain pins belong to the same group and can be swapped. The PinGroup should be a number greater than zero. If two pins of the same Polarity have the same PinGroup, they can be swapped using the **Package->Pin Numbers** command in the Schematic Editor or the **Misc->Pin Numbers** command in the Navigator.

These attributes are assigned to the symbol and its pins by use of the **Add->Symbol Attribute** and **Add->Pin Attribute** commands of the Symbol Editor. If a symbol has an attribute assigned, it becomes the default value for each instance of the symbol on the schematic. The default values can be overridden for a particular instance of the symbol on the schematic by use of the **Add->Attribute** command of the Schematic Editor or Navigator.

Refer to the *Attribute* section in this manual for more details on attributes in the ECS environment.

Fanout Analysis

Fanout analysis checks to see that the FanOut attached to a net is greater than the sum of all FanIns attached to the same net. Becasue this is a relative measurement, the units for FanOut are arbitrary as long as they are consistent with the units for FanIn.

The attributes required for fanout analysis are attached to pins as follows:

Pins	Attributes
Input	Polarity = Input; FanIn specified
Output	Polarity = Output; FanOut specified
	WireOr = T, Y, or O. (Tri-state, Yes, Open collector) The WireOr attribute is an optional attribute used when necessary.
Bidirectional	Polarity = Bidir; Both FanIn and FanOut specified

In addition to the above pin specifications, tri-state or open-collector output pins should have the WireOr attribute set to one of *Y*, *O*, or *T*. These represent *Yes*, *Open collector*, and *Tri-state*. Bidirectional pins can have the WireOr attribute set only if the pin is part of a block symbol. Bidirectional pins on primitive symbols cannot have the WireOr attribute specified.

Hi/Low Analysis

This is more comprehensive than a fanout analysis. The loads and drives are compared for low and high logic values separately. This allows logic families such as TTL, which have strong drive when driving low and weak drive when driving high, to be correctly analyzed. The attributes required for the symbol pins in the Hi/Low analysis are:

Pins	Attributes
Input	Polarity = Input; LoadLow and LoadHigh specified
Output	Polarity = Output; DriveLow and DriveHigh specified
Bidirectional	Polarity = Bidir; LoadLow, LoadHigh, DriveLow, and DriveHigh specified

In addition to the above pin specifications, tri-state or open-collector output pins should have the WireOr attribute set to one of *Y*, *O*, or *T*. These represent *Yes*, *Open collector*, and *Tri-state*. Bidirectional pins can have the WireOr attribute set only if the pin is part of a block symbol. Bidirectional pins on primitive symbols cannot have the WireOr attribute specified.

Sometimes it is desirable to perform a Loading or Fanout Analysis on a portion of the circuit. A problem that arises is that the input pins from the top level block don't have any drive specified and the output pins from the top level block don't have any load specified. The electrical Rules Checker has a method of overcoming this difficulty.

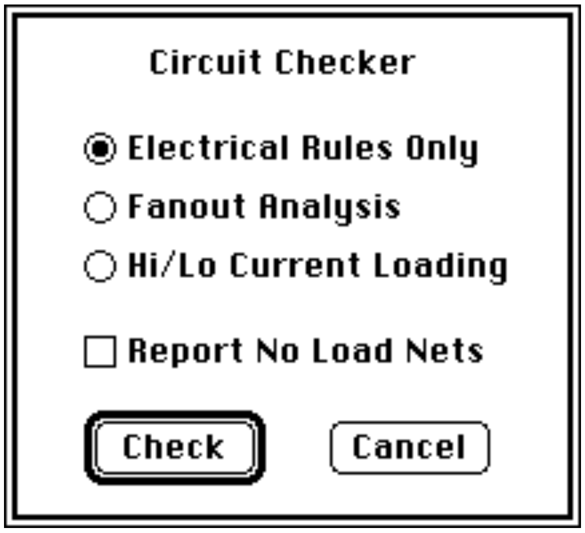
The default pin attributes of the top level Block symbol determine the load and drive from external sources. The value of the load attribute (FanIn, LoadLow, and LoadHigh) on an input pin represent the minimum external driving capability of the pin. This is checked against the loading on all the lower level schematics.

The value of the driving attribute (FanOut, DriveLow, and DriveHigh) on an output pin represents the sum of the external loads on the pin. This value is checked against the drive on the lower level schematic. Also, if an external input pin is to be wire-ored with an internal output pin, the pin in the top level symbol should have the WireOr attribute set.

For Printed Circuit Board technology, primitives should be either Gates or Components. Each symbol instance should have a RefDes assigned. Each pin must have a PinNumber. Pins that can be swapped should have the same PinGroup. Pins which represent pins in multiple sections should have a list of numbers in the PinNumber attribute.

Operating the Electrical Rules Checker

When you select **ERC Check** from the **Tools** menu of the Hierarchy Navigator, a dialog box pops up with several check box options as shown in Figure 7-3.



● Electrical Rules Only
○ Fanout Analysis
○ Hi/Lo Current Loading
☐ Report No Load Nets

Figure 7-3 ***ERC Check*** *Dialog Box*

The dialog box is explained below.

Electrical Rules Only	Performs basic connectivity checking and flags open pins and unconnected nets. This includes checking to ensure that each net is driven by the output of some gate. Multiple outputs driving a single net cause warning messages unless each of the outputs are open-collector or tri-state.
Fanout Analysis	Invokes the fanout analysis phase in addition to the connectivity checking. This check produces error messages whenever the sum of the loads (FanIn) on a net exceeds the capacity (Fanout) of the driving gate.
Hi/Lo Current Loading	Performs a loading analysis on both the high and low states. This is useful when the high and low drives are asymmetrical as in TTL. This check is similar to the Fanout analysis except that it compares the LoadLow versus DriveLow and the LoadHigh versus DriveHigh.
Report No Load Nets	When checked, a warning is generated for each net that does not drive any inputs.
Check	Invokes the Electrical Rules Checker and writes any errors found to the file *design_name.err*. The error list is also displayed in a list box as described below.
Cancel	Terminates execution of the Electrical Rules Checker.

Each error is written to a separate line in a file having the name *design_name.err* where *design_name* is the root name of the design. This file is automatically opened and the the error messages are displayed in a list box. Clicking the mouse on a given line in the list box causes the schematic to pan and/or traverse the hierarchy to show the offending net or symbol. This net or symbol is highlighted and a plus mark is placed as close to the exact error condition as possible.

Using this tool, you can quickly step through and identify most of the electrical problems in your design.

Operating the PCB Checker

The Printed Circuit Board Checker is invoked from the **Tools** menu of the Hierarchy Navigator. The checker verifies that the assignment of Gates and Components in a design is correct using two main criteria:

- If two symbols are assigned the same reference designator then they must be the same type of Gate. Gate symbols represent only one section of a Component. For example, a 7400 Gate symbol represents one NAND gate from a package of 4. The reference designator (RefDes attribute) assigns the symbol to a package.
- The checker also verifies that each Gate is used only once in a package. The different sections of a package should each have different Pin Number assignments. If a package has common pins (e.g. 74175 has four D type flip flops with a common clock and common clear), then the common pins must each be connected to the same net.

Each error is written to a separate line in a file having name *design_name.err*.

This file can be read using the **Tools->View Report** command. Clicking the mouse on a given line in the dialog box causes the offending net or symbol to be highlighted on the schematic. This allows you to step through the errors, one at a time.

Error Messages for ERC Check

For the Fanout Analysis, *LOAD* means FanIn and *DRIVE* means FanOut. For the Hi/Lo Current Loading Analysis, *LOAD* means either LoadHigh or LoadLow and *DRIVE* means either DriveHigh or DriveLow.

**** Too many messages to Display ****
Only the first 150 errors are listed. Some of these errors should be corrected before running the checker a second time.

Unconnected Input Pin ...
Each Input or Bidirectional pin must be connected to a net.

Invalid LOAD specified ...
The load attribute must be a number.

Invalid DRIVE specified ...
The drive attribute must be a number.

LOAD Specified for OUTPUT Pin ...
An output pin should only have a load if it is a tri-state pin (WireOr = T).

No LOAD Specified for Tri-state OUTPUT Pin ...
An output pin must have a load if it is a tri-state pin (WireOr = T).

No DRIVE Specified for OUTPUT Pin ...
Every output pin must have a Drive.

No LOAD Specified for BIDIR Pin ...
Every bidirectional pin must have a Load.

No DRIVE Specified for BIDIR Pin ...
Every bidirectional pin must have a drive.

No LOAD Specified for INPUT Pin ...
Every input pin must have a load.

DRIVE Specified for INPUT Pin ...
An input pin should not have a drive.

There are no PINS in Net ...
Every net should be connected to at least one pin.

There are no Loads in Net ...
Every net should drive some input pin.

There is no Drive pin in Net ...
Every net should be driven by some output pin.

Warning - Wired OR in Net ...
If more than one output pin is driving a net, each of the output pins should have the WireOr attribute set to Tristate or OpenCollector or Yes.

LOAD (100) > DRIVE (50) in Net ...
The total of all the loads on the net is greater than the output capacity of the driving pin.

Error Messages for PCB Check

**** Too many messages to Display ****
Only the first 150 errors are listed. Some of these errors should be corrected before running the checker a second time.

Missing RefDes on Symbol Instance ...
Each Gate or Component must have a reference designator assigned.

Missing Connector Name on I/O Pin Instance ...
Symbols of type Pin should have a connector name. Use the Reference Designator command to add one.

Missing .SCH file for Block Symbol ...
Each primitive symbol in a Printed Circuit design should be a Gate or Component.

Encountered CELL Type Symbol ...
Each primitive symbol in a Printed Circuit design should be a Gate or Component.

Pin ... is Connected to Different Nets
Two Gates of the same part have one of their common pins wired to different nets. Or, two Gates of a package have the same pin numbers.

RefDes ... Assigned to Symbols of Different Types
If two symbols have the same reference designator, they belong in the same package. They must be of the same symbol type (DeMorgan equivalents are considered the same).

Duplicate Assignment of RefDes ...
Components must have unique reference designators.

Tools->View Report

This command is automatically added to the **Tools** menu of the Hierarchy Navigator if you have the DAT option. This command displays a dialog box allowing you to choose a file. The contents of this file are displayed in a list box. Error messages from netlisters and other PIK programs now use this as the standard interface to display error messages in the Navigator. Third party programs can also make use of this interface.

A special syntax allows the the **Tools->View Report** command to jump to and highlight specified nets, instances, pins or symbol types in a design. If you click on a line in the list box, the line is parsed looking for a keyword followed by a valid identifier. The keyword and identifier can be embedded inside a comment. The keyword and identifier are separated by: any number of spaces (including 0) followed by an equal sign (=) followed by any number of spaces (including 0). You can use uppercase, lowercase or mixed case for the keyword and identifier. The keywords are:

I *or* Inst *or* Instance	represents an instance, identifier is instance
N *or* Net	represents a net, identifier is net name
P *or* Pin	represents a pin, identifier is pin name
T *or* Type *or* Symbol	represents a symbol type, identifier is type

Examples of lines in the list box are:

```
I=adf.pdiff   Bad spice attribute on this line
Unconnected pin  =  adf.pdiff-input1
The following symbol=NAND2 has a connectivity problem
```

Clicking on the first example line causes the Navigator to jump to schematic *adf* which contains instance *pdiff*. Instance *pdiff* is highlighted. The text of the error message is ignored. This free format allows the keywords in the error message to be relatively unobtrusive.

Viewing Critical Paths

The above command, **Tools->View Report**, allows the display of critical paths. The file syntax required is as shown in the example below:

```
[path 1]
first net=.input
first inst=.adder.nand3
second net=.adder.nand3.N_23
…
[path 2]
first net=.cin
first inst=.adder.mux
second net=.adder.mux.control
…
```

where header information is between square brackets and items in each sublist below the corresponding header are treated as described above for **Tools->View Report**. When the command is first invoked, a list box is displayed containing the lines from each header. Clicking on a header line in this list box displays a second list box containing the individual entries for the selected header.

Clicking on an entry in the second list box causes the Hierarchy Navigator to jump to the instance or net contained in the selected line. The system of keywords and identifiers as described for **Tools->View Report** is valid here also.

This syntax can be used to view any group of errors or features in a design. For example, a checking program that can identify ten different types of errors could write out a file with the following general format.

```
[errors of type 1, nets with more than 7 connections]
first error of type 1 found on net=.input
second error of type 1 found on net=.adder.sub.n_6
third error of type 1 found on net=.clock
…
[errors of type 2, instances with more than 10 pins]
first error of type 2 found on inst =.control_block
second error of type 2 found on inst = .reg.mux
third error of type 2 found on inst = .ram.decode1.I_3
…
```

Packaging PCB Devices

The packager is a Design Analysis Tools (DAT) utility that assigns reference designators and pin numbers to PCB primitive symbols. Reference designators and pin numbers are usually required for PCB layout purposes. They are also necessary for a well-documented PCB design. Having commands to manipulate and perform checks on the reference designators and pin numbers saves time and prevents errors in the PCB design process. You must have the DAT option in order to enable the following packaging functions.

Iterated symbols must be packaged in the Hierarchy Navigator using the **Misc->Ref Designator** command.

The remainder of this section describes the necessary setup or configuration information required, as well as the individual packaging commands.

The **Package** menu of the Schematic Editor contains the six commands listed below.

Query Packaging	Queries design by packaging information.
Check Packaging	Sets on-line checking mode of operation.
Clear Packaging	Erases the packaging information on a symbol.
Auto Package	Automatically assigns the packaging information.
Ref Designator	Assigns the reference designators.
Pin Numbers	Assigns the pin numbers.

Configuration Information

The packaging commands are activated in the Schematic Editor by setting the application mode in the ecs.ini Editor (ECS Preferences Editor on Macintosh) to *PCB* or *both*. This adds the **Package** menu to the Schematic Editor.

The packaging commands operate on the *RefDes* and *PinNumber* attributes of Gate symbols and Pin symbols and on the *RefDes* attributes of Component symbols.

The ECS accepts reference designators of one or two alpha characters followed by an integer. Pin numbers are limited to four alphanumeric characters.

The ECS expects the *RefDes* attribute in a Gate or Component symbol definition to be set to the prefix desired for the reference designator. This prefix can be one or two alpha characters (A-Z). A prefix with any other characters are ignored. If no default value is assigned to the attribute, a prefix of U is assigned for Gate and Component symbols when packaging in the Schematic Editor. Pin symbols must have their reference designators manually assigned.

In the case of Gate symbols, each pin should have its *PinNumber* attribute set to the list of pins that the pin could have for each gate in the package. The same pin number can appear more than once in a list indicating that the pin is common to more than one gate. The same pin should not appear in lists for different pins on the symbol. See the PCB example in the main manual section, *Entering Your Design,* for more information about pin numbers and reference designators.

Package->Query Packaging

This command scans the schematic and lists groups of Gate and Component symbols according to their packaging information.

The **Package->Query Packaging** command operates in one of three ways:

- Typing a specific reference designator, such as *U12,* displays in a list box the sheet/zone location and gate section of all elements designated *U12*.
  Clicking on a symbol results in the same action.
- Typing in a reference designator prefix, such as *C,* displays in a list box the sheet/zone location and gate sections of all elements whose reference designators start with *C*. Any elements whose reference designators have not been assigned are indicated as such.
  Clicking on a symbol whose reference designator has not been assigned results in the same action.
- Typing in a symbol's file name, such as *NAND2*, displays in a list box the sheet/zone location and gate sections of all *NAND2* elements in the schematic.
  Clicking on a symbol while holding down the *Shift* key results in the same action.

Package->Check Packaging

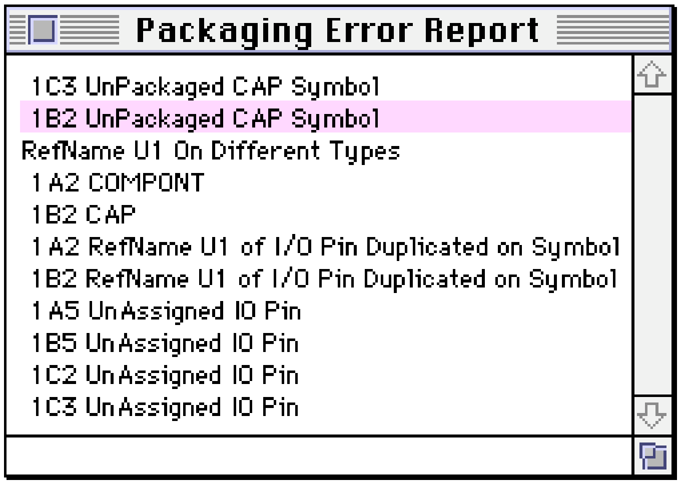
This command runs in an on-line mode in conjunction with all but the **Package->Query Packaging** command. The command can be terminated by selecting the **Package->Query Packaging** command or any command not on the **View** or **Package** menus.

In checking mode, the system scans the schematic for packaging errors. The checking command works in conjunction with the other four packaging commands. Each time any packaging information is changed, the checking report is updated and displayed in the Packaging Error Report display.

If no errors are detected, *No Errors Detected* is displayed.

If any packaging errors are detected, the error is displayed. If the error involves a conflict between two or more symbols, the error is followed by a series of indented lines showing the sheet/zone of each symbol causing the error. Errors that involve only one symbol are listed with the sheet/zone information on the same line.

Selecting an error from the list box, as shown in Figure 7-4, causes the offending symbol to be shown. The cursor is temporarily changed to an X and moved to mark the symbol. Moving the mouse restores the cursor to its original position and appearance.


Packaging Error Report

```
 1C3 UnPackaged CAP Symbol
 1B2 UnPackaged CAP Symbol
RefName U1 On Different Types
 1A2 COMPONT
 1B2 CAP
 1A2 RefName U1 of I/O Pin Duplicated on Symbol
 1B2 RefName U1 of I/O Pin Duplicated on Symbol
 1A5 UnAssigned IO Pin
 1B5 UnAssigned IO Pin
 1C2 UnAssigned IO Pin
 1C3 UnAssigned IO Pin
```

Figure 7-4 Error Selected From List Box

The checking command also acts as a repackage command if no other packaging command is selected. Clicking on any symbol repackages the symbol as if you *Cleared* and then *Auto Packaged*.

Error Messages

The following errors are reported by the packaging checker:

Gate Multiply Assigned

The same reference designator or the same pins appear on two or more gates. The two or more gates that are multiply assigned appear in an indented list immediately below the error message.

Invalid Pins on Gate

The pin numbers on the instance are not a valid set according to the symbol definition.

RefDes on Different Types

The same reference designator is used on symbols of differing types where type refers to the name of the symbol file. The offending symbols appear in an indented list immediately below the error message.

RefDes Multiply Assigned

The same reference designator appears on two or more components. The offending symbols appear in an indented list immediately below the error message.

Unpackaged Symbol

A symbol has not yet been packaged.

In all cases, specific symbols mentioned in the error listing are shown with their respective sheet/zone locators at the beginning of the line. This sheet/zone locator has the form of an integer representing the sheet, followed by a letter and number coordinate pair that specifies the XY location of the symbol on the sheet. For example, a symbol having a sheet/zone locator of *3D5* would be located on sheet 3 at the region of the sheet defined by the intersection of the D location along the side of the page and the 5 location on the bottom of the page.

Package->Clear Packaging

This command is used to clear the packaging information on one or more symbols. It can clear:

- a single symbol by clicking on the symbol
- all symbols of the same type by holding the *Shift* key while clicking on one symbol of the desired type
- all symbols enclosed in a drag box
- all symbols with a specified prefix by entering the prefix from the keyboard
- all symbols by entering *ALL* on the keyboard

After being cleared, the reference designator shows as the prefix that you specified in the symbol definition. The pins on Gate symbols are set to the values for the first gate section in the package.

Package->Auto Package

This command automatically selects an appropriate reference designator for Component symbols and reference designator and pin assignments for Gate symbols. It ignores Pin symbols.

Reference designators for Component symbols are assigned by choosing the first available name in numeric order with the prefix specified in the symbol definition file. The numeric part of the reference designator is greater than or equal to a minimum number that can be entered from the keyboard or allowed to default to 1.

The minimum number can be specified by appending it to the prefix of all commands entered via the keyboard. It can also be entered independently without a prefix. The minimum number can be reset to 1 by entering 1, entering a blank line, or by restarting the command from the menu. If a minimum number is specified, the Prompt line shows: *Package (# >= nn) =*.

Reference designators and pin assignments are made for Gate symbols by first using any free gate sections in existing packages and then assigning a new package. The name is selected for new packages as they are selected for Component symbols.

The **Auto Package** command can process one, several or all unpackaged symbols in the schematic. The command packages:

- a single symbol by clicking on the symbol
- all symbols of the same type by holding the *Shift* key while clicking on one symbol of the desired type
- all symbols enclosed in a drag box

- all symbols with a specified prefix by entering the prefix via the keyboard
- all symbols by entering *ALL* on the keyboard

To package iterated instances, you must use the **Misc->Ref Designator** and **Misc->Pin Numbers** commands in the Hierarchy Navigator.

Package->Ref Designator

This command explicitly sets the reference designator on a Gate, Component, or Pin symbol. Invalid assignments are permitted so that you can move reference designators among symbols. The checking program traps such errors.

You first specify the reference designator to be assigned. This is done using one of the following methods:

- entering the full name (i.e., *U23*)
- entering the prefix (i.e., *U*), in which case the system selects the first available unique name with that prefix
- clicking on a symbol causes the system to select a unique name with the same prefix as the selected symbol
- clicking on a Gate or Pin symbol with the *Shift* key depressed, in which case the system copies the name from the symbol

The name you are assigning is displayed on the cursor. The command assigns the name to:

- A single symbol by clicking on the symbol. The name on the cursor is then set to the next available name.
- A single symbol by clicking on the symbol with the *Shift* key depressed. The name on the cursor is attached to the selected symbol and the name attached to the cursor is not incremented. This is useful for gate assignments where multiple sections are in the same package and thus have the same reference designator.
- All symbols enclosed in a drag box. The symbols are assigned unique names starting with the selected name. The name attached to the cursor is incremented to the next available.
- All symbols enclosed in a drag box with the *Shift* key depressed. All selected symbols have the assigned name attached to them and the name attached to the cursor remains the same.

To package iterated instances, you must use the **Misc->Ref Designator** and **Misc->Pin Numbers** commands in the Hierarchy Navigator. The RefDes attribute on an iterated instance consists of a list of reference designators. The list is comma or blank delimited. A sequence of reference designators can be formed by enclosing the numeric range in ()’s or []’s.

For example, a symbol with instance name *CC[1:20]* specifies 20 iterations of a symbol. If you assign this symbol a reference designator of *C1,C3,C(5:20), C22,C24*, then the 20 iterations of the symbol are referenced as *C1, C3, C5, C6, C7 …C20, C22, C24*.

Package->Pin Number

This command explicitly sets the pin numbers on Gate symbols. The command assigns a gate symbol to a particular section of a component. It also lets you swap pins on a Gate symbol.

You must enter the gate section or pin number to be assigned by using one of the following methods:

- typing the pin number
- typing a letter corresponding to the gate section in the package (i.e., *A, B, …*)
- clicking on a pin to copy its number
- clicking on a symbol body to copy its gate section

The gate section or pin number is displayed on the cursor. The gate to be numbered is designated by clicking on its body or on one of its pins. The command checks the requested assignment to ensure that it is a valid gate for the symbol.

The command then assigns the correct pin numbers to the gate. If an explicit pin number is requested and if one of the gate's pins is selected, the command attempts to assign that pin by swapping with other pins in the same pin group.

The command does not prevent duplicate gate assignments from being made. The command also permits common pin conflicts to be created. These errors are detected and reported by the checking command.

To package iterated instances, you must use the **Misc->Ref Designator** and **Misc->Pin Numbers** commands in the Hierarchy Navigator.

Derived Attributes

Derived attributes illustrate the true power of attributes in the ECS environment. In many respects, derived attributes provide the power and capability of a spreadsheet in a schematic entry system. Standard attributes are described in the *Entering Your Design* section.

Derived attributes take their value from:

- other attributes on the symbol, symbol's pins, or nets connected to pins
- attributes on the symbol's parents or children
- global attributes accessible throughout the design hierarchy

Instead of entering a value for a derived attribute, you enter a format string that defines how the attribute should be derived. The value is interpreted every time the attribute is used.

The syntax of the format string for a derived attribute is below.

- Normal characters are transferred from the format string without interpretation. This is identical to a simple attribute. In fact, derived attributes can contain simple values just like normal attributes.
- Whenever a pound sign (#) is followed by a number, the attribute with that number is copied to the output string. An alternate syntax to achieve the same result is the pound sign (#) followed by the attribute name in square brackets. For example, #[width] takes the value of the width attribute.
- If *#number* or *#[attr name]* is immediately followed by a back slash (\) and another number representing a field index, the given field is stripped from the attribute. The fields of an attribute are separated by spaces, commas, slashes, colons, semicolons, or equal signs. This allows a single number to be taken from a list of values.
- Different instances in the design are accessed by preceding the instance name with the @ sign. This allows attributes on children instances to be accessed. For example, *@inst1#[width]* takes the width symbol attribute from instance *inst1*.
- Attributes on parent instances are accessed using two periods to indicate one level up in the hierarchy. For example, *@..#35* takes the width symbol attribute from the parent instance.
- Pin attributes are accessed by specifying an instance name followed by a minus sign and the pin name. Pin names are terminated by a space, , equal sign, =, number sign, #, or dollar sign, $. If the pin name is empty, it refers to the current pin, `@-,` or the original pin`, @.-.`

  For example, *@inst2-in3#[SilosLoad]* takes the value of the *SilosLoad* pin attribute from pin *in3* on instance *inst2*.
- If a pin name in the preceding syntax is followed by an equal sign the net connected to the pin is used instead of the pin, `@-=, or, -pinname=`.
- A single period is used to indicate the original instance. For example, *@.-in3#42* takes the value of pin attribute 42 from pin *in3* on the original instance.
- Global attributes are accessed by a dollar sign followed by either the global attribute number or the attribute name in square brackets. For example, both *$17* and *$[lambda]* access global attributes.
- Table data in schematics is accessed using the following syntax:

  &table_name[row,column]

  where

table_name	is the table's name spelled exactly
row	may either be a number or *=value* or *=#attr* or *=$const*. When row contains an equal sign the table is searched to find the row that has a label equal to *value* or equal to the value of attribute *attr* on the current symbol instance. Row numbers start at one. Row zero is the column label.

column may either be a number or *=value* or *=#attr* or *=$const*. When a column contains an equal sign, the table is searched to find the column that has a label equal to *value* or equal to the value of the global constant *const*.

Column numbers start at one. Column zero is the row label.

The following examples help clarify the use of derived attributes. The attributes are first defined in the **ecs.ini** file the as shown in Table 7-1.

Attribute Number	Attribute Modifier	Attribute Window	Attribute Name	Comments
0	!	0	InstName	Instance Name
1	!	2	Type	Symbol Name
3			Value	
35		8	Width	MOS transistor width
36		9	Length	MOS transistor width
103	*	10	W/L	combined width and length
110	*	11	NewWidth	width stripped from 103
111	*	12	NewLength	length stripped from 103

Table 7-1 Attributes Defined in ***ecs.ini*** *File*

When you edit the symbol for a MOS transistor, you can create a new attribute that contains both the width and length with a slash separating them from the letters *W=* and *L=*. When this new attribute is displayed, no matter how a symbol is oriented in a schematic, the width always precedes the length.

Figure 7-5 indicates what happens if you display separate width and length parameters on a symbol that is rotated 180° or mirrored. The order of the width and length is reversed in the 180° instance. If you use a single attribute string to represent the width followed by the length, then the display order is independent of orientation.

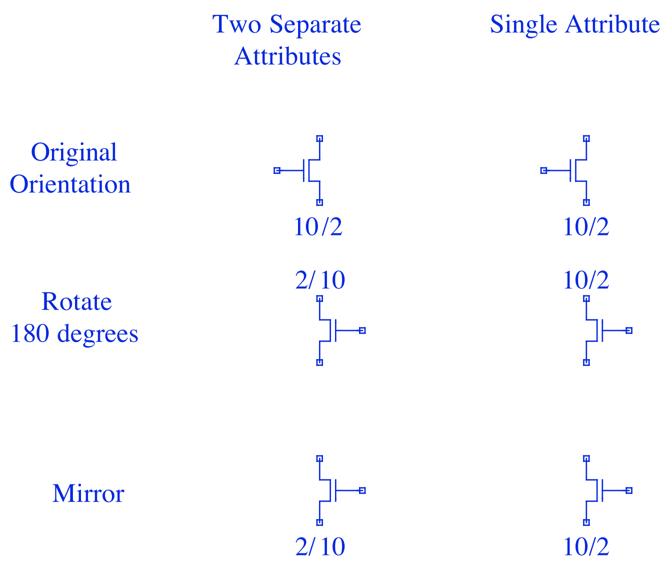


Figure 7-5 Effects of Rotation on Attributes

You can use attribute number 103 to represent this new attribute. The following shows how to add the new attribute to the MOS transistor symbol from the Symbol Editor.

1. Select **Add->Symbol Attributes**. This displays the Attribute Editor.
2. Select the *W/L* attribute. This is attribute number 103 from Table 7-1.
3. With this attribute in the edit field, type either:

   W=#35/L=#36

   or

   W=#[width]/L=#[length]

   followed by a *Return*.

The *W=* is taken literally and put into the new attribute as is. It is followed by the value of attribute number 35, which in this case happens to be the width. The */L=* is interpreted literally and followed by the value of attribute 36, which is the length. If you then display attribute number 103 in a window and assume the width is 10 and length is 2, the following appears.

W=10/L=2

The MOS transistor symbol can be rotated and not lose the proper order of width followed by length. Also, the derived attributes can be used to set values of ECS-defined attributes in the attribute number range 0 to 99.

If attribute 103 is defined as above, you can strip off the width and length separately, and place into two new derived attributes 110 and 111. When you edit the symbol for the MOS device, you use the **Add->Symbol Attributes** command and edit the *NewWidth* and *NewLength* attributes to read:

#103\2	newly defined width, attr 110, returns *10*
#[W/L]\4	newly defined length, attr 111, returns *2*

This strips the second and fourth fields and places them into the *NewWidth* and *NewLength* attributes. The W and L are treated as ordinary fields. The = sign and / act as delimiters so you can manipulate text and strings.

If you defined a derived attribute 104 and gave it the following value:

#103	returns *W=10/L=2*

it returns the exact contents of attribute 103, that is *W=10/L=2*. The literal characters *W=*, *L=*, and the slash *(/)* all become part of the new attribute.

Calculated Derived Attributes

Attributes are text strings and have numerical meaning only in the context of another program that decodes the text strings. There is one exception where numeric values are actually treated as numbers. This occurs if you place some attributes inside parentheses (). When parentheses are encountered in derived attributes, the expression inside the parentheses is evaluated numerically. The expressions are limited to decimal numbers with no more than three digits after the decimal point. The basic four arithmetic operations are allowed as well as several comparison operations. The comparisons are evaluated as shown below.

(a > b)	1 if a is greater than b, otherwise 0
(a >= b)	1 if a is greater than equal to b, otherwise 0
(a < b)	1 if a is less than b, otherwise 0
(a <= b)	1 if a is less than or equal to b, otherwise 0
(a = b)	1 if a is equal to b, otherwise 0
(a != b)	1 if a is not equal to b, otherwise 0

Several examples of attribute definitions containing simple arithmetic calculation follow.

Format String	*Result*
AREA=(#[length]*#[width])	AREA=nnn where nnn is width*length from this symbol

A=(#35*#36* ((#35*#36) > 10)) + (#35*#36*1.2*((#35*#36) <= 10)))	This expression sets *A* equal to length times width, if the area is greater than 10. If the area is less than or equal to 10, then *A* is set equal to the area times the constant 1.2.
(4* (5+3))	32
ABC(1.2/100+ .001)	ABC0.013

Derived attributes can reference other derived attributes, but there is a limit of four levels of nesting. When an attribute references an instance that is not available, the entire attribute string for that instance is left blank.

Derived Attributes and Hierarchy

All of the previous discussion explains how to move different attributes or attribute fields within one level of hierarchy. One of the most powerful features of attributes in ECS is their ability to pass information both up and down the hierarchy tree. This takes place in the context of the Navigator where the whole design hierarchy is maintained. It is not possible to pass attributes up and down in the Schematic Editor or Symbol Editor.

To extract an attribute from another instance, the @ symbol is used followed by the instance specification. The instance specification for the parent symbol is two periods (..), and the instance specification for a child instance on the underlying schematic is the particular instance name.

Being able to pass attributes up and down the hierarchy is useful during back annotation of actual transistor sizes extracted from layout back to the schematic. It is often helpful to have actual transistor sizes annotated onto inverters and other gates. If transistor sizing information is extracted from a layout, this sizing can be annotated back to the transistor level in ECS. At that point, if the attributes are set up correctly, ECS can automatically transfer the low-level transistor sizes up through the hierarchy to the gate level or higher.

Case Study Using Derived Attributes

This example shows how an IC design can be scaled for different processing dimensions.

One of the fundamental goals in integrated circuit design is to equalize the delays for rising and falling signals passing through a circuit. This is usually achieved by sizing n-channel and p-channel devices so that the path from vdd to the output has the same resistance as the path from the output to ground. In a simple CMOS inverter, this results in the p-channel device to vdd having a width about twice that of the n-channel device to ground. In an inverter on a 2μ process, the n-channel device might have a width of 2μ and a length of 4μ, while the p-channel device would have a W/L of $8\mu/2\mu$ in order to maintain a balanced delay. On a 1μ process, the corresponding values would be $2\mu/1\mu$ for n-channel and $4\mu/1\mu$ for p-channel.

As the process dimension decreases, everything scales linearly. The following example uses this fact to design a latch that can be automatically scaled to different process dimensions. Dimensionless W/L attributes are attached to transistors. At the primitive symbol level, these dimensionless quantities are multiplied by a process scale factor, Lamda, that causes actual dimensions to be assigned to the transistor. This allows a complete design to be scaled to different process dimensions by changing a single attribute, Lamda, inside the lowest-level primitives, the n- and p-channel transistors. The alternative to this approach is to regenerate the library every time a new process is used, and to include the new widths and lengths on all the symbols.

The attribute definitions in Table 7-2 are used in this example.

Attribute Number	Attribute Modifier	Attribute Window	Attribute Name	Comments
35	*	8	Width	MOS transistor width
36	*	9	Length	MOS transistor width
102	-		Lambda	smallest processing geometry
105	*	13	Zn	dimensionless W/L for n-ch
106	*	14	Zp	dimensionless W/L for p-ch

Table 7-2 Attribute Definitions for Derived Attribute Case Study

The width and length attributes are derived attributes that represent the actual width and length of the MOS transistors. Lamda is the scale factor that, when multiplied by the dimensionless width and length terms, gives the actual transistor widths and lengths. Zn and Zp are the dimensionless widths over lengths in the format W/L. Zn is for the n-channel transistors and Zp is for the p-channel transistors.

Table 7-3 shows the different attributes and their values at the various levels of the hierarchy. The bottom level in the hierarchy is the MOS transistor, with symbols nch.sym and pch.sym. The width and length attributes are generated by multiplying the process scale factor, Lamda, by the dimensionless width over length terms, Zn and Zp.

Lamda is defined at the transistor symbol level; the minus sign as an attribute modifier ensures that it cannot be overridden at higher levels. Zn and Zp are overridden at the latch schematic level and attached to each individual transistor or inverter instance in the schematic. Zn and Zp are then passed down to the transistor symbol level where they are used in the calculation of actual width and length of the transistors. The 10/2 and 20/2 in the inv.sym column are the default values. As you can see from the schematics in Figures 7-6, 7-7, and 7-8, all the instances have different values that are set at the latch.sch level in the hierarchy.

	nch.sym	pch.sym	inv.sch	inv.sym	latch.sch
Width ->	(#102*@..#105\1)	(#102*@..#106\1)			
Length ->	(#102*@..#105\2)	(#102*@..#106\2)			
Lamda ->	2	2			
Zn ->			@..#105	10/2	
Zp ->			@..#106	20/2	

Table 7-3 Attributes and Values in Preferences Editor

The next three figures show a latch with the various levels of hierarchy underlying it.

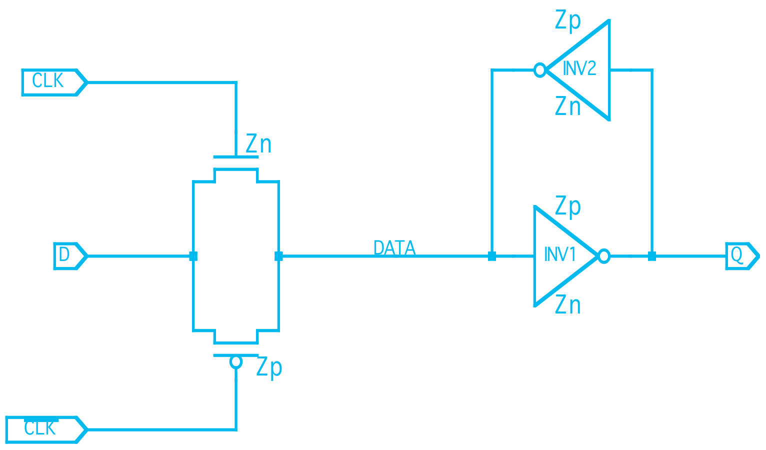


Figure 7-6 latch.sch Schematic

Figure 7-6 shows latch.sch with instances of MOS transistors and inverters. All transistors have different sizes. Attributes Zn and Zp are edited on n-channel and p-channel pass devices and passed down to MOS symbols. Attributes Zn and Zp are both edited on each inverter, with the values passed down to the inverter schematic. Attributes Zn and Zp are dimensionless width to length for n-channel and p-channel transistors.

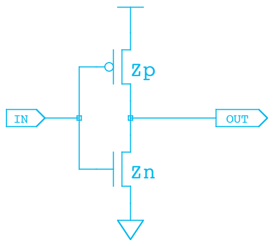


Figure 7-7 inv.sch Schematic

Figure 7-7 shows inv.sch. Attributes Zn and Zp are passed down from above and passed down to individual transistors below.

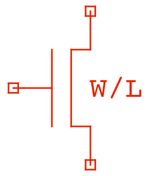


Figure 7-8 nch.sym Symbol

Figure 7-8 shows nch.sym. Attribute Zn is passed down from above. The dimensionless width is stripped from Zn and multiplied by Lamda to make the attribute Width (35). The same is true for attribute Length (36). These two device characteristics scale as process variable Lamda scales. This makes it easy to use the same schematic for different fabrication processes, such as 5μ, 1.2μ, 0.5μ…They can be changed with a single attribute, Lamda. The p-channel device is similar except it uses Zp from hierarchy.

The SPICE netlist below shows that the dimensionless transistor widths and lengths are multiplied by the Lamda scale factor (1.2) to give physical dimensions in microns.

```
M1 Q DATA VDD VDD PMOS L=2.4U W=12U
M2 Q DATA GND GND NMOS L=2.4U W=6U
M3 DATA Q VDD VDD PMOS L=1.2U W=1.2U
M4 DATA Q GND GND NMOS L=2.4U W=1.2U
M5 DATA CLK' D VDD PMOS L=2.4U W=24U
M6 DATA CLK D GND NMOS L=2.4U W=12U
```

8

Waveform Tool

This section describes the features and operation of the Waveform Tool (WT) used in the Engineering Capture System (ECS). The WT is a waveform display and analysis tool. It is typically used in conjunction with the Hierarchy Navigator but can also be used in stand-alone mode.

The major topics covered in this section are:

- Conventions used throughout this section
- Installation of the Waveform Tool
- Viewing waveforms
- Manipulating waveforms in the display window
- Performing time and logic value measurements on waveforms
- Interaction with the Hierarchy Navigator (cross probing)
- Waveform Tool Command reference

Waveform Tool

The Waveform Tool (WT) interprets the results of logic or timing simulations. The logic states of schematic nets are displayed as time line traces (waveforms). The nets whose waveforms are to be displayed are chosen interactively from the schematic. Query functions can be used to trace signals to their source on the schematic. Trigger functions can be used to locate the occurrence of a logic event. Delays between events can be measured with markers.

The Waveform Tool has two modes of operation. It can be used in conjunction with the Hierarchy Navigator and also as a stand-alone waveform analyzer. In addition, the WT currently interfaces to Verilog waveforms, SILOS waveforms and Timemill waveforms.

Installation

The WT is installed using the same procedure described in the *Before You Begin* section. The application itself is called *Waves* with the icon shown in Figure 8-1.

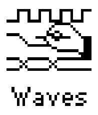


Figure 8-1 Icon for Waves Application

Basic Operations

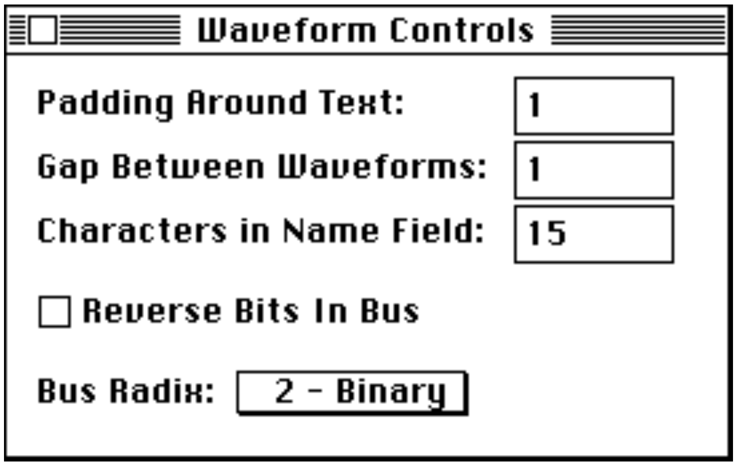
This section outlines:

- Configuring the WT display parameters
- Starting and stopping the WT
- The WT window
- Saving waveforms
- Printing waveforms

WT Configuration

There are several configuration variables that control display parameters in the waveform display area. These variables are modified with the **Controls->Wave Controls** menu in the ECS Preferences Editor as shown in Figure 8-2. These variables and their purpose are:

Padding Around Text	The amount of padding above and below text in the waveform name area. This determines the signal waveforms height. The padding is measured in units of one-quarter the text height.
Gap Between Waveforms	The amount of space between waveforms. The gap is measured in units of one-quarter the text height.
Characters in Name Field	The number of characters visible in the waveform name area.
Reverse Bits in Bus	If this is *checked*, the least significant bit (LSB) of buses is displayed to the left whenever the bus value is displayed as a number. If it is *not checked*, the bus value is displayed with the same bit order as a binary number, that is with the LSB to the right.
Bus Radix	The radix, or base, that is used to display buses. The choices are decimal, octal, hexadecimal, and binary.


Waveform Controls

Padding Around Text: 1

Gap Between Waveforms: 1

Characters in Name Field: 15

☐ Reverse Bits In Bus

Bus Radix: 2 - Binary

Figure 8-2 Waveform Controls Dialog Box in ECS Preferences Editor

You can also adjust the following colors with the **Controls->Colors** menu in the ECS Preferences Editor.

Simulation Value	Text showing simulation value on schematic.
SimValue 0	Small square on schematic at probed nodes indicating logic low at node.
SimValue 1	Small square on schematic at probed nodes indicating logic high at node.
SimValue X	Small square on schematic at probed nodes indicating unknown state at node.
SimValue Z	Small square on schematic at probed nodes indicating high impedance node.
Wave Marker	Time Marker used for measuring time deltas.
WaveName	Display of waveform names.
BusWave	Display of buses.
Highlight Wave	Highlight used for waveforms.
SignalWave	Display of driving strength signals.
Unknown Wave	Display of unknown strength signals.
High Z Wave	Display of high impedance signals.
ResistiveWave	Display of resistive signals.
Supply Wave	Display of supply strength waves.

Starting and Stopping the Waveform Tool

The Waveform Tool can be started from one of three places.

- the **Tools** menu of the Hierarchy Navigator. Refer to the *Before You Begin* section for information about how to add a **Tools** menu entry.
- the **Simulator** menu of the Hierarchy Navigator. Refer to the *Simulation Interface* section for more information.
- as a stand-alone tool. In this case the program is started by double-clicking on the *waves* application. A dialog box is displayed allowing you to choose a valid waveform history file to open.

In each case the syntax of specifying the Waveform Tool is:

waves *command_line_options*

where *command_line_options* is one or more of the following options:

name	the parameter *name* specifies the root name for the display save file *.wav* and either the timewave history *.his* or the binary save file. The *name* parameter must be specified for the Waveform Tool to start.
-nav	connects to navigator, used only in conjunction with Hierarchy Navigator
-sim	specifies that waves is running dynamically with simulator
-time	reads waveform data from timewave history file. The timewave history file format is explained in the *Simulation Interface* section.
-haz	adds the command **Misc->View Report** to the Waveform Tool menu.

To exit the WT, select the **File->Quit** command. If you changed the waveform display in any way, a dialog box appears with the following choices:

Save	Allows you to save the positions of the waveforms,trigger information and a binary waveform database.
Discard	Discards any changes made to the waveform configuration such as waveform positions and trigger information.
Cancel	Returns to the WT.

WT Window

Figure 8-3 below shows a typical Waveform Tool window.

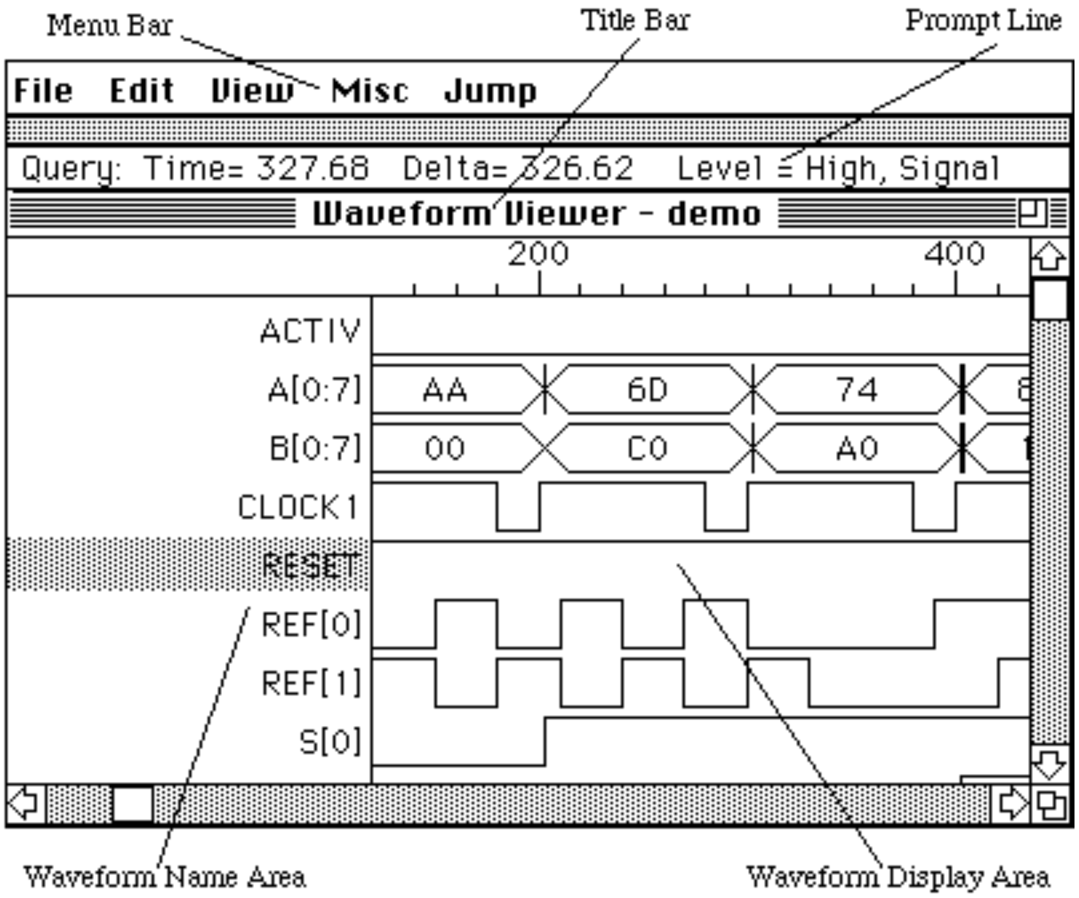


Figure 8-3 Typical Waveform Tool Window

The following information describes the elements of the Waveform Tool window shown above.

Title Bar
The title bar across the top shows the name of the current design. This name is derived from the design being examined.

Menu Bar
The menu bar is above the title bar. All the WT commands are accessed through this menu bar.

Waveform Name Area
The waveform name area contains the names of any waveforms displayed.

Waveform Display Area

The waveform display area has a time scale on the top. The scroll bar underneath the waveform display allows you to pan backwards and forwards through the waveforms. If there are too many waveforms to be visible at one time in the waveform display area, the scroll bar at the right of the display allows you to access all the waveforms. The right-hand scroll bar provides a vertical pan function.

The operation of the scroll bars is described in the *Before You Begin* section of this manual.

Prompt Line

A prompt line is initially located below the menu bar but can be moved any where on the screen. This prompt line serves two purposes:

- Any command prompts are shown here.
- If there are no commands pending or if the **Misc->Query** command is active, the prompt line contains the following information.

Time	This is the time corresponding to the Query Cursor, a solid vertical line in the waveform display.
Level	This is the digital level of the selected waveform at the intersection with the Query Cursor. The selected waveform is highlighted in the waveform name field. This takes any values supported by the simulator being used — typically zero (0), one (1), or unknown (X). Bus values are shown with the current radix.
Trig	This indicates whether the current trigger conditions are *True* or *False*. This only appears on the prompt line if triggers are defined.

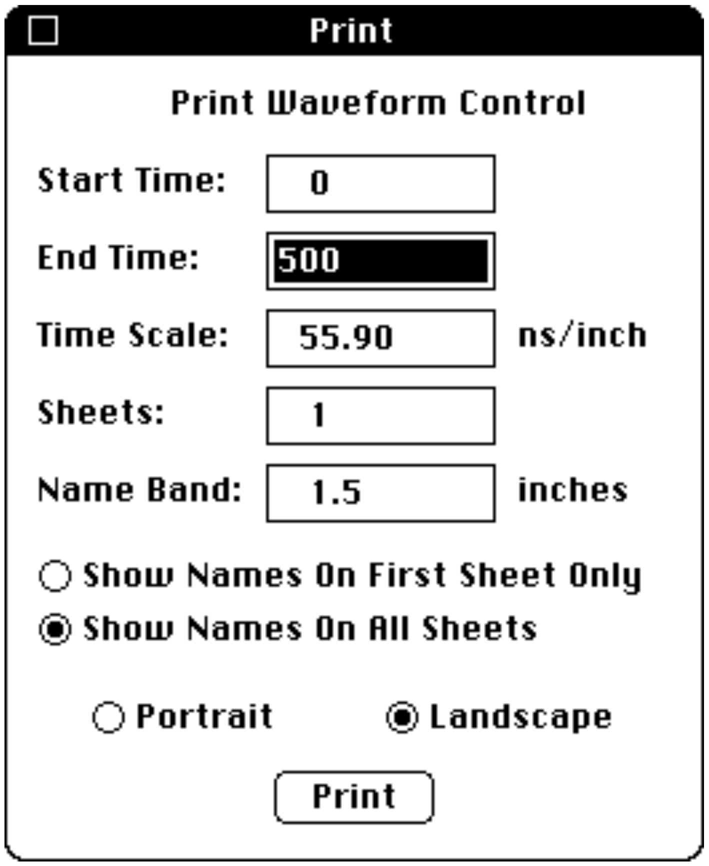
Saving Waveforms

After you have completed your analysis of the waveforms, you can save the WT configuration and binary data using the **File->Save** and **File->Save As** commands. The information saved consists of:

- Binary waveform database
- Waveform names displayed
- Trigger conditions

Printing Waveforms

The waveform display can be printed using the **File->Print** command. The **File->Print** command displays a dialog box as shown in Figure 8-4. The various controls are described below.



Print

Print Waveform Control

Start Time: 0

End Time: 500

Time Scale: 55.90 ns/inch

Sheets: 1

Name Band: 1.5 inches

○ Show Names On First Sheet Only

● Show Names On All Sheets

○ Portrait ● Landscape

Figure 8-4 Print Waveform Control Dialog Box

Start Time	Simulation time at which the plot is to begin.
Stop Time	Simulation time at which plot is to finish.
Time Scale	Scale of plot measureed in nanoseconds/inch, where the units of simulation are assumed to be nanoseconds. If you force the time scale, the number of pages required to display from the start time to the stop time is automatically calculated.
Sheets	Number of sheets required to plot the waveforms. If you force the number of sheets, the scale is calculated to just fill the specified number of sheets.

Name Band	Waveform names are shown in a vertical strip along the left hand edge of the page. The width of this strip is determined by this parameter. This strip is printed on every page if the *Show Names On All Sheets* radio button is selected, else only the first sheet displays the names of waveforms.
Portrait	If this radio button is selected, the time axis of the plot is displayed along the short edge of the paper.
Landscape	If this radio button is selected, the time axis of the plot is displayed along the long edge of the paper.
Close	Clicking the close box terminates the current selection.
Print	Uses the current settings and sends the plot to the printer.

The **File->Page Setup** command allows you to change many of the characteristics of the printed page.

Moving Around

Once the waveforms are displayed, there are several ways to change the waveform display area. One simple way is to change the size of the window in which the WT is resident. This is described in the *Before You Begin* section of this manual.

View Commands

The **View** commands change the horizontal time dimension of the waveform display and hold the vertical scale and position constant. This allows different time segments of the waveforms already present in the display to be viewed. Details of the different **View** commands follow.

View->ZoomIn	The **ZoomIn** command doubles the horizontal magnification each time it is executed. This shows a shorter time segment of the waveform in more detail. The center of the current window remains centered after the **ZoomIn**.
View->ZoomOut	This command halves the current magnification each time it is executed. It shows a longer time span with less detail.
View->Window	You use this command to magnify an area in greater detail. After selecting the command, you drag the mouse along the time axis in the area to be magnified. This highlights an area along the top of the waveforms. The highlighted area is then expanded to fill the waveform window when the mouse is released.
View->FullFit	Selecting this command causes the total simulation time of the waveforms to be displayed. It is handy to survey the extent of

	the waveforms before performing any analysis on them. You can then use the **View->Window** command to magnify any particular portion in detail.
View->Redraw	This command redraws the current waveform display.

Scroll Bars

Scroll bars are displayed along the bottom and right edges of windows allowing the window to be panned over the entire range of waveform data. The amount of movement is controlled by clicking or dragging various portions of the scroll bar.

- The horizontal scroll bar under the waveform display controls the time scale.
- The vertical scroll bar controls the position within the set of visible waveforms.

Arrows	Clicking on one of the arrows at the end of the scroll bar causes the window to move a small amount.
Slider	The slider indicates the approximate location of the window relative to the entire range of data. Dragging the slider along the bar moves the window accordingly.
Paging	Clicking in the area between the slider and the arrow causes the window to move almost a full page in that direction.

The scrolling controls are described in detail in the *Before You Begin* section.

Manipulating Waveforms

The most fundamental operation in the WT is adding waveforms to the display. Once waveforms are displayed, they can be moved, deleted, copied, and converted to bus format using the following commands.

Many of the commands described below require you to select a waveform in order to operate on it. To select a waveform, use the **Misc->Query** command. Holding the shift key down or dragging the mouse allows you to select multiple waveforms at once.

Edit->Add Wave	Using this command is the most basic way of adding single waveforms to the display. Type the name of the required waveform. The waveform is added below the last waveform. The window pans vertically to show the waveform just added. Multiple waveforms are added together to display a bus by typing the desired names on the prompt line separated

	by commas. A maximum of 256 waveforms can be added to the display.
Edit->Clear	Waveforms that are no longer needed in the display are removed using the **Edit->Clear** command. This command does not affect the contents of the clipboard. It operates on waveforms selected with the **Misc->Query** command.
Edit->Cut	This command removes waveforms from the display and places them in the clipboard.
Edit->Copy	This command copies waveforms from the display and copies them to the clipboard.
Edit->Paste	This command pastes waveforms stored in the clipboard.
Edit->Expand Bus	This command expands the currently selected bus into its constituent signals. The individual signals are added to the display immediately below the target bus.
Edit->Create Bus	This command concatenates multiple signals or buses that are currently selected and creates a new bus combining all the signals or smaller buses.
Edit->Duplicate	This command copies the currently selected waveforms to the bottom of the waveform display. This command does not affect the contents of the clipboard. This is useful for duplicating reference signals like clocks.

Moving a Waveform

Waveforms are moved in the waveform display using the following procedure.

1. Select the waveform(s) to be moved. The waveform names show up highlighted.
2. Click the mouse on the selected waveforms and drag them to the new location. A horizontal bar appears over the top selected waveform in the name band field. This bar moves during the drag the drag operation to show where the selected waveforms are inserted.

Tools->Probe Item (from Navigator)

If a schematic for the design exists, the **Misc->Probe Item** command from the Hierarchy Navigator provides the easiest way to add waveforms to the display. Use the following procedure to add waveforms from the Navigator.

1. Select the **Misc->Probe Item** command in the Hierarchy Navigator.
2. Click on the required net in the Hierarchy Navigator, and the waveform for that net is added to the display. Buses from the schematic can be probed, but the bus must be probed at the highest level at which it exists in the hierarchy. **Probe Item** is only available when the WT is used with the Navigator.

Analysis Techniques

This section explains some techniques used to extract useful information from the waveforms. Logic waveforms consist of logic values changing with time. The WT is typically used to determine when time logic values occur. You should find it easy to obtain the logic values and the times at which events occur.

Accelerator keys greatly speed execution for the following commands. The accelerator key codes are displayed in the menus next to the commands. The accelerator keys are documented in the *WT Command Reference* section.

Logic Level and Time Measurements

The **Misc->Query** command allows you to measure both logic levels and times on any displayed waveforms. When the **Query** command is in use, a query cursor is displayed on the screen. This is a solid vertical line that passes through the point where you click the mouse.

The prompt line displays the time of this query cursor and the logic level of the selected signal. If the selected signal is a bus, the value of the bus is displayed. The selected signal is highlighted in the name field and is determined by the vertical position of the cursor when the mouse is clicked.

Relative time measurements are made between the marker and the query cursor. The marker is set using the **Misc->Set Marker** command. This sets the marker to the current query cursor location. To measure the time difference between two events:

1. Move the query cursor to the first event.
2. Set the marker to this location using the **Misc->Set Marker** command.
3. Move the query cursor to the second event. The relative time between these two events is shown on the prompt line under the heading, *Delta*.

Moving the Query Cursor

Several commands allow you to move the query cursor.

Jump->Left and **Jump->Right**
These commands move the cursor to the left and right by one small tick mark. This is useful for scanning through a waveform slowly or for accurately positioning the cursor over an event. The time represented by one small tick mark changes as the scale is changed with the **View** commands.

Jump->Page Left and **Jump->Page Right**
These commands move the query cursor to the left or right by one large tick mark (equal to 10 small tick marks). The time represented by a large tick mark changes as the scale is changed with the **View** commands.

The **Jump->To Start** or **Jump->To End** command jumps to the beginning or end of the waveform.

The **Jump->To Marker** command jumps to the current marker position.

Jump->Up and **Jump->Down**
These commands move the selection point up or down one waveform. You can quickly scan through a number of waveforms with these commands and read off the logic values from the prompt line for each waveform as you scan. The up and down arrow keys also activate these commands.

The **Jump->To Time** command allows you to specify a time. The display is then centered about this specified time.

Jumping to Events

Events are logic level changes on a waveform. Any signal changing inside a bus is considered an event on the bus. Timing measurements are usually made between events. Several commands in the WT make it easier to find events and also align the cursor to events.

This is especially helpful if the scale is zoomed out and the visible resolution in the display is not very great. In this case, you can jump to the exact location of the event. It is difficult or impossible to get the same accuracy by manually moving the cursor.

Jump->Next Change moves the query cursor to the next event on the selected waveform. A common procedure is to measure the time delay from one event to another.

1. Position the query cursor on the waveform containing the first event. The cursor is placed before the first event.
2. Move the query cursor to the event by executing the **Jump->Next Change** command. This locks the cursor exactly to the first target event.
3. Execute the **Misc->Set Marker** command to set the marker at the first event.

4. Position the query cursor on the waveform containing the second event. This places the cursor before the second event.
5. Move the cursor to the second event by executing the **Jump->Next Change** command. This locks the cursor exactly to the second target event. The time difference between the two events is displayed on the prompt line.

This procedure works equally well for the **Jump->Next Trigger** command.

Triggers

A trigger is an event that meets some specified criteria. The signal conditions used to define a trigger in the WT are:

- High
- Low
- Unknown
- Change
- Positive Edge
- Negative Edge
- Bus = specified value

The **Misc->Set Trigger** command allows you to apply any of the above conditions to any waveforms in the display. When the conditions applied to each waveform are met, a trigger event is defined. This can be a very selective process if conditions are applied to many waveforms in the display.

The **Jump->Next Trigger** command advances the query cursor from the current location to the next time a trigger event occurs. If no trigger event exists, the cursor advances to the end of the waveform display.

Displaying Bus Values

Bus values are displayed on the bus waveforms and on the prompt line if a bus is selected.

The **Misc->Bus Radix** command allows you to set the radix or base used to display bus values. The available choices are:

- Binary
- Octal
- Decimal
- Hexadecimal

Misc->View Report

The **Misc->View Report** command reads error information from a file and then interactively displays the errors. The file is displayed in a list box, each line of the list box representing one error condition. Clicking on a line in the list box causes the waveform display to jump to that particular error condition. If used with the Hierarchy Navigator, the schematic displays and highlights the pin driving the problem net. This allows you to quickly scan through an error report and see the waveform conditions causing the error and see exactly where in the circuit the error is caused.

Any third party tool can be used to create the error file. Refer to the *WT Command Reference Section* for more details.

Interaction with the Hierarchy Navigator

The **Tools->Find Item** command from the Hierarchy Navigator menu is used to locate the area of the circuit that is driving a particular waveform. The Navigator automatically jumps to the schematic containing the device driving the waveform. This works across levels of hierarchy and across many sheets in one schematic. The net associated with the waveform is highlighted in the design.

This is useful if you find an interesting event in the waveform display and want to locate the source of the event on the schematic. This command only works with the Hierarchy Navigator.

The **Misc->Query** command causes the net associated with the currently selected waveform to be highlighted on the schematic. If the query window is already open in the Hierarchy Navigator (open by executing **Misc->Query** from the Navigator), the contents change to reflect the latest net queried with the **Misc->Query** command in the WT.

The **Tools->Probe Item** command adds waveforms to the WT display when you probe a net in the schematic.

Displaying Simulation Values on Schematic

The logic values of any waveforms in the WT are displayed on the schematic. The values reflect the placement of the query cursor in the waveform display. As the query cursor is moved, the logic values on the schematic are continuously updated to reflect any changes.

The logic values are displayed on the schematic using two methods.

- A small colored square is attached to any symbol nodes on the schematic that are probed. The color of these squares indicates different logic values. The colors are set in the ECS Preferences Editor. The colored squares are useful when the schematic is at a low magnification and the text is too small to read.
- Inside the small colored square is the text representation of the logic value. The text value is one of: 1, 0, X (for unknown), or Z (for high Z). Any text or symbol supported by the simulator can be used.

WT Command Reference

This section describes the commands available in the Waveform Tool (WT).

The following conventions are used to describe the commands.

- The commands are listed in alphabetical order.
- ECS commands are always shown in bold face type.
- If they exist, accelerate keys are shown to the right of the command name heading at the top of each command description.
- ECS commands are usually shown with the name of the menu where the command is found. For example, the **Expand Bus** command is found under the **Edit** menu. It is referred to as **Edit->Expand Bus**.

Quick Command Reference

command name	Keyboard Short Cut
->About	
->Help	⌘?
Edit->Add Wave	
Edit->Clear	*Delete* key
Edit->Copy	⌘C
Edit->Cut	⌘X
Edit->Create Bus	
Edit->Duplicate	⌘D
Edit->Expand Bus	
Edit->Paste	⌘V
Edit->Redo	
Edit->Undo	⌘Z
File->Page Setup	
File->Print	⌘P
File->Quit	⌘Q
File->Save	⌘S
Jump->Down	⌘ *Down Arrow*
Jump->Left	⌘ *Left Arrow*
Jump->Next Change	
Jump->Next Trigger	
Jump->Page Left	
Jump->Page Right	
Jump->Right	⌘ *Right Arrow*
Jump->To End	
Jump->To Marker	
Jump->To Start	
Jump->To Time	
Jump->Up	⌘ *Up Arrow*
Misc->Bus Radix	
Misc->Hide Marker	
Misc->Query	⌘I
Misc->Set Marker	
Misc->Set Trigger	⌘T

Misc->View Report	
View->Full Fit	
View->Redraw	
View->Window	
View->Zoom In	□J
View->Zoom Out	□K

->About ...

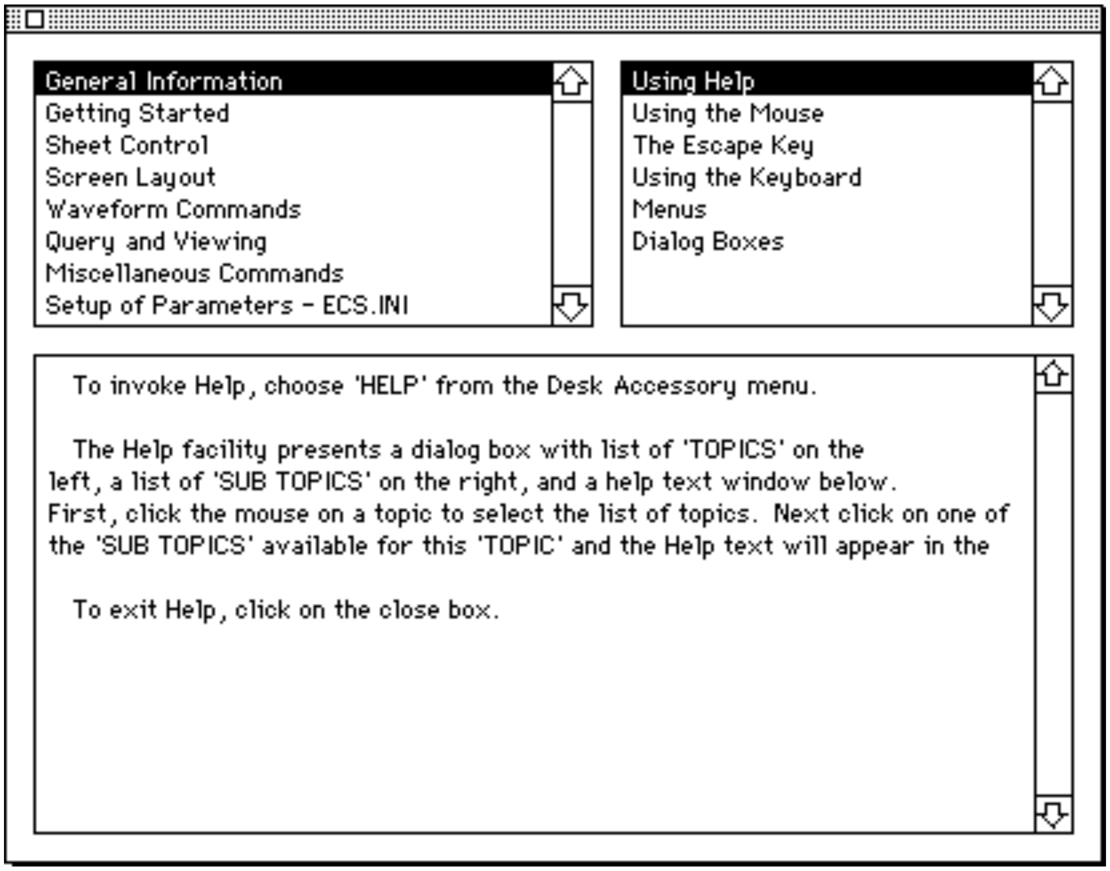
The About box under the apple icon lists the software version number. It also lists the serial number of the copy being used.

->Help

⌘?

On-line help is available at any time with the **Help** command. The **Help** facility presents a dialog window with a list of *topics* on the left and a list of *subtopics* on the right as shown in Figure 8-5. To access the help information:

1. Select the *topic* using the mouse. The list of *subtopics* is updated.
2. Select a *subtopic* from the list using the mouse. The on-line help information for that particular *subtopic* is displayed in the large bottom window.



To invoke Help, choose 'HELP' from the Desk Accessory menu.

The Help facility presents a dialog box with list of 'TOPICS' on the left, a list of 'SUB TOPICS' on the right, and a help text window below. First, click the mouse on a topic to select the list of topics. Next click on one of the 'SUB TOPICS' available for this 'TOPIC' and the Help text will appear in the

To exit Help, click on the close box.

Figure 8-5 Help Screen

Edit->Add Wave

The **Edit->Add Wave** command adds waveforms to the display. When you select the command, a dialog box like that shown in Figure 8-6 is displayed.

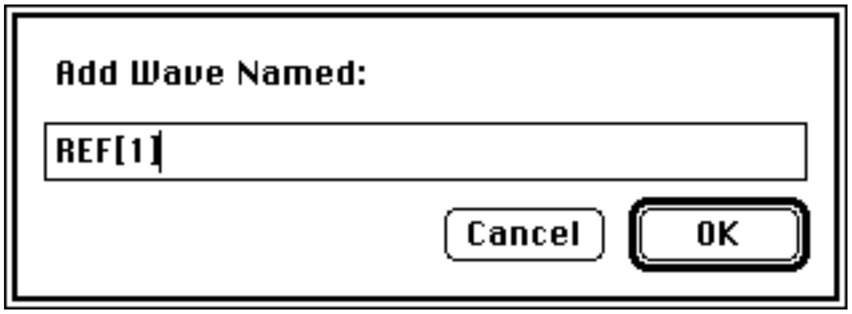


Figure 8-6 ***Edit->Add Wave*** *Dialog Box*

To add a waveform to the display:

- Type the required signal name and press *Return*. The new waveform is added to the bottom of the display.

If you enter a compound name, a bus is created containing each of the specified signals. For a description of compound names, see the *Entering Your Design* section.

Multiple copies of waveforms can be added to the display.

See Also

- **Misc->Probe Item** is a command in the Hierarchy Navigator that adds waveforms directly from the schematic.
- **Wave->Move** moves waveforms from the bottom of the display.
- **Wave->Delete** removes waveforms from the display.

Edit->Clear

Delete key

This command deletes waveforms you have selected. This command does not affect the contents of the clipboard.

Once a set of waveforms are in the selected set (see **Misc->Query** for information on selecting waveforms) they can be deleted from the drawing by choosing the **Edit->Clear** command. A shortcut for the **Edit->Clear** command is to use the *Delete* key. Use this command instead of **Edit->Cut** when the clipboard contains data you don't want to overwrite.

See Also

- **Edit->Cut**
- **Edit->Copy**

Edit->Copy

□C

The **Edit->Copy** command copies selected waveforms to the clipboard.

Once a set of waveforms is in the selected set (see **Misc->Query** for information on selecting waveforms), it can be copied to the clipboard by choosing the **Edit->Copy** command. The original data in the waveform display is unaffected by the **Edit->Copy** command.

See Also

- **Edit->Duplicate**
- **Misc->Query**

Edit->Cut

▯X

The **Edit->Cut** command deletes selected items from the symbol or schematic drawing and places them on the clipboard.

Once a set of waveforms is in the selected set (see **Misc->Query** for information on selecting waveforms) it can be copied to the clipboard by choosing the **Edit->Cut** command. Use this command instead of **Edit->Clear** when you want to affect the contents of the clipboard.

See Also

- **Edit->Clear**
- **Misc->Query**

Edit->Create Bus

This command adds together currently selected waveforms to form a new bus waveform. The selected set of waveforms (see **Misc->Query** for information on selecting waveforms) can contain bus waveforms as well as individual signal waveforms.

Use the following procedure to create a bus:

1. Select the waveforms you wish to have added together. (see **Misc->Query** for information on selecting waveforms)
2. Choose the **Edit->Create Bus** command. The selected waveforms are added together to form the new bus. The waveforms are added together in such a way that the top-most waveform is mapped to the Most Significant Bit (MSB) of the new bus while the lowest selected waveform maps to the Least Significant Bit (LSB) of the new bus.

   The original selected waveforms are left intact in their original positions. The new bus is added to the bottom of the waveform display.

See Also

- **Edit->Expand Bus**
- **Misc->Query** selects waveforms
- **Misc->Bus Radix**

Edit->Duplicate ▯D

This command allows you to copy waveforms from one area of the display to another. It is useful for adding copies of global signals, such as clocks. It is often convenient to have a reference signal like a clock close to the event being examined.

The **Edit->Duplicate** command does not affect the contents of the clipboard.

To operate:

1. Select the waveforms you wish to have duplicated. (see **Misc->Query** for information on selecting waveforms)
2. Select the **Edit->Duplicate** command. The selected waveforms are copied to the bottom of the waveform display. The original selected waveforms are left intact in their original positions.

See Also

- **Misc->Query**

Edit->Expand Bus

This command expands a bus into its constituent signals.

Use the following procedure to expand a bus.

1. Select the bus waveform(s) you wish to have expanded. (see **Misc->Query** for information on selecting waveforms)
2. Choose the **Edit->Expand Bus** command. The selected bus waveforms are expanded out into the individual signals comprising the bus. The expanded signals are added to the display immediately below the bus waveform being expanded. The original selected bus waveforms are left intact in their original positions.

The **Edit->Expand Bus** command shows up as greyed in the **Edit** menu until at least one waveform has been selected.

See Also

- **Misc->Query** selects waveforms
- **Edit->Create Bus** creates bus waveforms from individual waveforms

Edit->Paste ⌘V

This command allows you to paste data from the clipboard. Data must be on the clipboard in order for the **Edit->Paste** command to be active. If there is no data on the clipboard, the command shows up greyed in the **Edit** menu.

See Also

- **Edit->Duplicate** copies a wave from one location and places it in another.
- **Edit->Cut** removes a waveform and adds to the clipboard.
- **Edit->Copy** copies a waveform and adds to the clipboard.

Edit->Redo

Edit->Redo allows you to recover from changes caused by the **Undo** command. As the **Undo** command backs up through the session, a log is retained. **Redo** can go forward, playing back the undone events until it reaches the end of the log.

The Undo log is discarded whenever a command (other than **Undo** or **Redo**) causes an entry in the data base. Once this log is discarded, **Redo** cannot recover undone commands.

The **Redo** command is most commonly used in the WT to recover deleted waveforms.

See Also

- **Edit->Undo** allows the previous state of the waveforms display to be recovered after making changes.

Edit->Undo ⌘Z

The **Undo** command allows you to back up in the edit process. The **Undo** command operates on all commands that result in data base changes in the Waveform Tool. It does not recognize any of the view control commands.

Undo backs up one event each time it is issued. The command is capable of undoing all of the events back to the last time the file was opened or saved.

The **Undo** command is most commonly used in the WT to recover deleted waveforms and to restore the original positions of waveforms on the display.

If you **Undo** too far back by mistake, you can recover your edits by using **Edit->Redo**.

See Also

- **Edit->Redo** allows recovery from **Undo**.

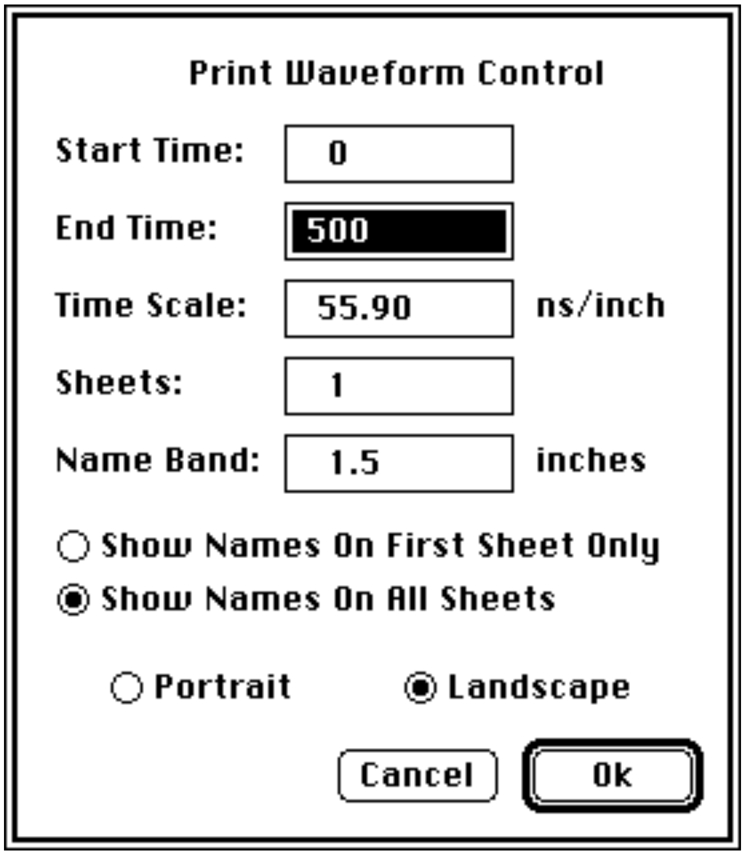
File->Page Setup

Please refer to **File->Page Setup** in section 5, the *Command Reference* section.

File->Print

⌘P

The waveform display can be printed using the **File->Print** command. The **File->Print** command displays a dialog box as shown in Figure 8-7. The various controls are described below.



Start Time: 0

End Time: 500

Time Scale: 55.90 ns/inch

Sheets: 1

Name Band: 1.5 inches

Show Names On First Sheet Only [✓ Show Names On All Sheets]

Portrait [✓ Landscape]

Figure 8-7 Print Waveform Control Dialog Box

Start Time	Simulation time at which the plot is to begin.
Stop Time	Simulation time at which plot is to finish.
Time Scale	Scale of plot measureed in nanoseconds/inch, where the units of simulation are assumed to be nanoseconds. If you force the time scale, the number of pages required to display from the start time to the stop time is automatically calculated.
Sheets	Number of sheets required to plot the waveforms. If you force the number of sheets, the scale is calculated to just fill the specified number of sheets.

Name Band	Waveform names are shown in a vertical strip along the left hand edge of the page. The width of this strip is determined by this parameter. This strip is printed on every page if the *Show Names On All Sheets* radio button is selected, else only the first sheet displays the names of waveforms.
Portrait	If this radio button is selected, the time axis of the plot is displayed along the short edge of the paper.
Landscape	If this radio button is selected, the time axis of the plot is displayed along the long edge of the paper.
Cancel	Terminates the printing of the plot.
OK	Uses the current settings and sends the plot to the printer.

Use the following procedure to print the waveform display.

1. Arrange the waveforms in the display in the order you wish them to be printed.
2. Select the **File->Print** command. The dialog box shown in Figure 8-7 is displayed.
3. Modify the display controls in the dialog box to obtain the required view of the waveforms in the plot.
4. Click the *OK* button to send the plot to the printer.

File->Quit

⌘Q

File->Quit terminates the execution of the Waveform Tool. The command also checks the current display for modifications that were made since the file was last saved. If the waveform's name, order or triggers are modified, the command offers three options:

Save	Saves the latest waveform display changes to the file *root.wav* where *root* is the design name being examined.
Discard	Ignores all waveform display changes and exits from the application.
Cancel	Returns to the application program.

Once the file has been saved or discarded, the **File->Quit** command closes all of the application's windows and terminates the Waveform Tool.

File->Save File->Save As

The **File->Save** command stores the current waveform database to a binary file. It also saves the current configuration of a work session to a file. The **File->Save As** command performs the same actions as **File->Save** except you can specify the name of the file to be saved.

The files saved allow you to store your waveforms and examine them at a later time. You can also compare the results of different simulations by saving various results to differently named files.

The configuration information is stored in a file having a file extension of *.wav*. This file includes the names of the waveforms currently displayed and any triggers currently defined.

The default extension for the binary database file is *.bin*. The root name of the binary file is determined from one of the following.

- the design name if the waveforms originated from the simulation interface
- the name of the timewave history file if the waveforms originated from a timewave history file
- the name of the binary history waveform file if the waveforms originated from a binary waveform file

See Also

- **File->Open** opens a timewave history file or a binary save file

Jump->Down

 Down Arrow

This command moves the current selection point down one waveform. If multiple waveforms are selected, the selection point moves one waveform down from the lowest selected waveform in the display.

The **Jump->Down** command is selected by

See Also

- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Right** moves query cursor one tick mark to the right.
- **Jump->Page Left** moves query cursor one major tick mark to the left.
- **Jump->Page Right** moves query cursor one major tick mark to the right.

Jump->Left

 Left Arrow

This command moves the query cursor one tick mark to the left. This has the effect of sliding the cursor back in time.

The interval between tick marks changes as you change the time scale with the **View** commands. **Jump->Left** tracks this different interval.

See Also

- **Jump->Down** moves selection point down one waveform.
- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Right** moves query cursor one tick mark to the right.
- **Jump->Page Left** moves query cursor one major tick mark to the left.
- **Jump->Page Right** moves query cursor one major tick mark to the right.

Jump->Next Change

This command moves the query cursor to the next logic transition in the currently selected waveforms. If the currently selected waveform is a bus, the query cursor moves to the next transition of any signal in the bus.

This command is useful for traversing the simulation data. Typically, you use this command to scan a primary input. Because primary inputs give rise to activity inside a design, you can jump from one initiating event to another. You can find later events on other signals that result from the changes on the primary input.

Use the following procedure to operate this command.

1. Position the query cursor to a point on the time scale before the logic transitions of interest.
2. Use the **Misc->Query** command to select the waveforms you wish to scan for changes. The selected waveforms are highlighted.
3. Choose the **Jump->Next Change** command from the menu. The query cursor jumps to the next logic transition of any of the selected waveforms. If the next transition is outside the current display, the waveform display is panned to center the query cursor in the middle of the display.

   If no logic transitions are detected in the selected waveforms, the query cursor jumps to the end of the waveform display.

See Also

- **Jump->Next Trigger** jumps to the next time that the trigger is active.
- **Misc->Query** selects waveforms

Jump->Next Trigger

This command advances the query cursor to the next time that satisfies the specified trigger conditions. If the query cursor is advanced off the current display, the display is centered around the new query time.

Use the following procedure to operate this command.

1. Use the **Misc->Set Trigger** command to set up all the required trigger conditions.
2. Position the query cursor before the section of waveforms you wish to investigate.
3. Choose the **Jump->Next Trigger** command. The query cursor jumps to the next time at which all the trigger conditions are met. If the query cursor jumps outside the current display, the display is panned to show the new query cursor placement.

See Also

- **Jump->Next Change** jumps to the next change on the selected waveform.
- **Misc->Query** positions the query cursor.
- **Misc->Set Trigger** sets the trigger conditions for the **Jump->Next Trigger** command.

Jump->Page Left

This command moves the query cursor one major tick mark to the left. This has the effect of sliding the cursor backward in time. One major tick mark equals 10 minor tick marks.

The interval between tick marks changes as you change the time scale with the **View** commands. **Jump->Page Left** tracks this different interval.

See Also

- **Jump->Down** moves selection point down one waveform.
- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Right** moves query cursor one tick mark to the right.
- **Jump->Page Right** moves query cursor one major tick mark to the right.

Jump->Page Right

This command moves the query cursor one major tick mark to the right. This has the effect of sliding the cursor forward in time. One major tick mark equals 10 minor tick marks.

The interval between tick marks changes as you change the time scale with the **View** commands. **Jump->Page Right** tracks this different interval.

See Also

- **Jump->Down** moves selection point down one waveform.
- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Right** moves query cursor one tick mark to the right.
- **Jump->Page Left** moves query cursor one major tick mark to the left.

Jump->Right

□*Right Arrow*

This command moves the query cursor one tick mark to the right. This has the effect of sliding the cursor forward in time.

The interval between tick marks changes as you change the time scale with the **View** commands. **Jump->Right** tracks this different interval.

See Also

- **Jump->Down** moves selection point down one waveform.
- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Page Left** moves query cursor one major tick mark to the left.
- **Jump->Page Right** moves query cursor one major tick mark to the right.

Jump->To End

This command moves the query cursor to the last time point of the simulation. This is the right-most point on the time axis of the waveform.

Selecting this command from the menu causes the query cursor to move to the right-most point on the display. It also causes the waveform display to pan to the end of the waveforms. This is typically at time 0.

See Also

- **Jump->To Beginning**

Jump->To Marker

This command returns the query cursor to the marker. It also centers the display around the marker.

The marker is indicated by a dashed line. The query cursor is indicated by a solid line. After the **Jump->To Marker** command is executed, the marker and query cursor are displayed in the same place. This is indicated with a dotted line.

You can use this command to measure the delta time between two points as follows:

1. Place the marker on the first of two time points you wish to measure.
2. Execute the **Jump->To Marker** command.
3. Move the query cursor using a combination of **Jump->Next Change** and **Jump->Next Trigger** to the second event or time point. The time delta between the two time points is indicated on the prompt line of the display.

See Also

- **Jump->Next Change** jumps to the next change on the selected waveform.
- **Jump->Next Trigger** jumps to the next time that the specified trigger conditions are met.

Jump->To Start

This command moves the query cursor to the start time of the waveform display. This is the left-most point on the time axis of the waveform.

Selecting this command from the menu causes the query cursor to move to the left-most point on the display. It also causes the waveform display to pan to the beginning of the waveforms. This is typically at time 0.

See Also

- **Jump->To End**

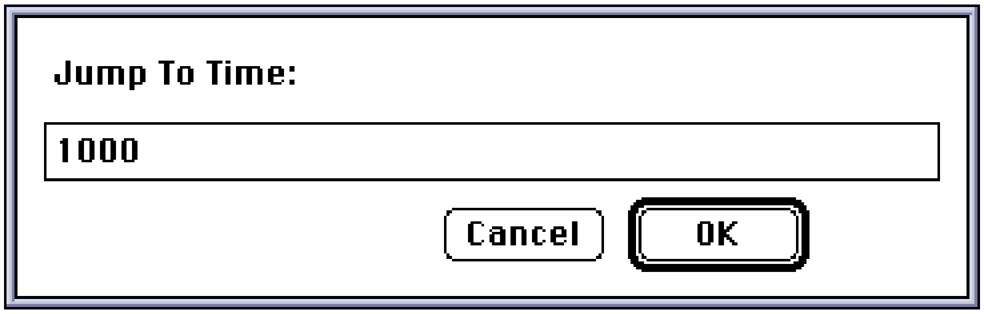
Jump->To Time

This command allows you to specify a time to which the waveform display pans.

Use the following procedure to examine the waveforms at any specified time.

1. Select the **Jump->To Time** command from the menu. A dialog box is displayed as shown in Figure 8-8.
2. Type the required time into the dialog box.
3. Clicking on the OK button causes the waveform display to pan to the specified time. The Query cursor also moves to the specified time.

   Clicking on the Cancel button aborts the jump.


Jump To Time:

1000

Figure 8-8 ***Jump->To Time*** *Dialog Box*

See Also

- **Jump->To End**
- **Jump->To Start**

Jump->Up

▯ *Up Arrow*

This command moves the current selection point down one waveform. If multiple waveforms are selected, the selection point moves one waveform down from the lowest selected waveform in the display.

The **Jump->Down** command is selected by

See Also

- **Jump->Up** moves selection point up one waveform.
- **Jump->Left** moves query cursor one tick mark to the left.
- **Jump->Right** moves query cursor one tick mark to the right.
- **Jump->Page Left** moves query cursor one major tick mark to the left.
- **Jump->Page Right** moves query cursor one major tick mark to the right.

Misc->Bus Radix

This command allows you to display buses using binary, octal, decimal, or hexadecimal notation.

Selecting this command opens a dialog box as shown in Figure 8-9, where you specify which notation is to used for displaying buses. All buses are displayed with the specified radix.

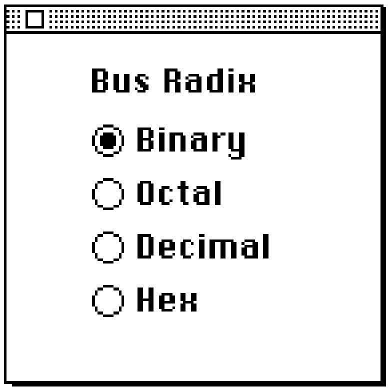


Figure 8-9 Dialog Box for ***Misc->Bus Radix*** *Command*

Misc->Hide Mark

This command removes the marker from the waveform display. This is useful when you print a section of the waveform display and do not want the marker to be visible.

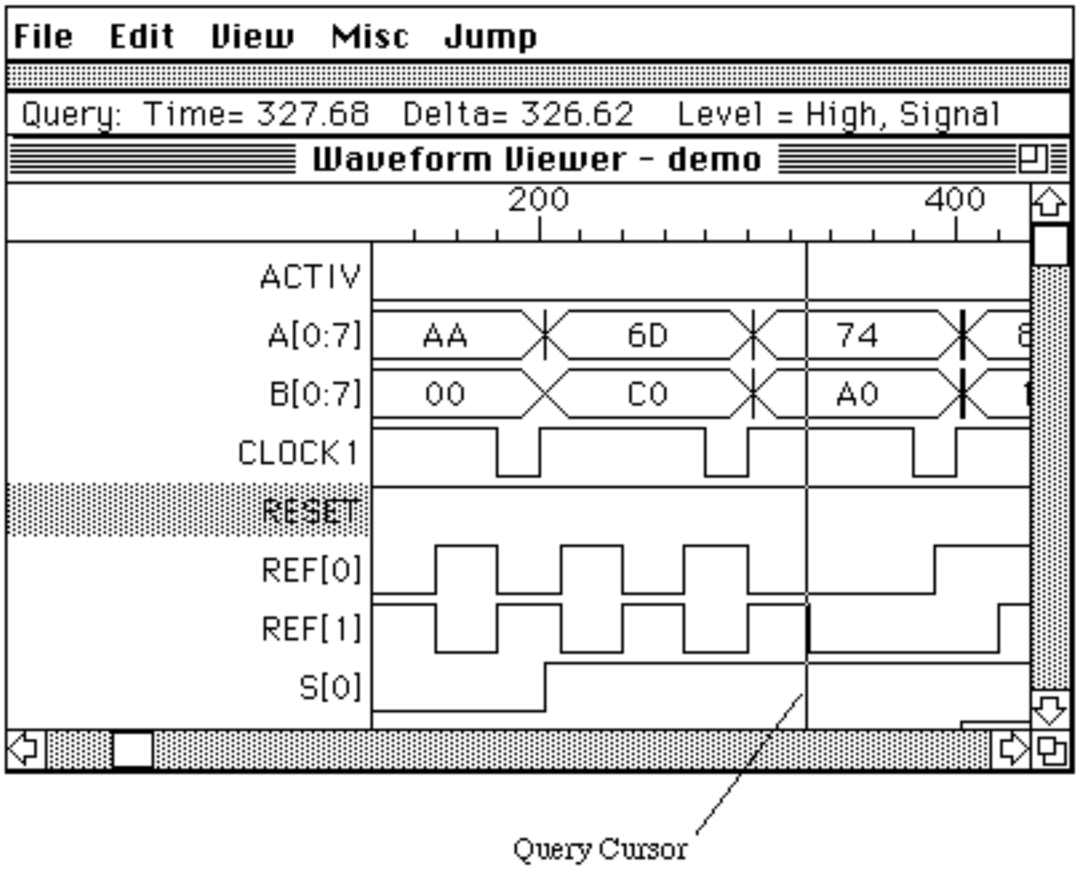
To execute the **Misc->Hide Mark** command, select it from the menu.

See Also

- **Misc->Place Mark** inserts the mark at the query cursor location.
- **Misc->Query** is used to measure relative time delays to the marker.

Misc->Query

The **Misc->Query** command positions the query cursor, selects waveforms and can move waveforms within the display.



Query: Time= 327.68 Delta= 326.62 Level = High, Signal

Figure 8-10 Waveform Window Showing Query Cursor

Query Cursor

The query cursor is a vertical line as shown in Figure 8-10. Information about the currently queried signal is displayed on the prompt line as shown in Figure 8-10. Use the following procedure to place the query cursor

1. Select the **Misc->Query** command.
2. Position the cursor in two dimensions within the main waveform window.

Time axis	sets the time of the query cursor
Wave axis	sets the selected waveform

3. Clicking the mouse locks the query cursor time and locks the selected waveform. The Selected waveform is highlighted in the name area as shown for the *RESET* signal in Figure 8-10. The time of the query cursor is displayed in the prompt line. The value of the selected waveform is also displayed as the *Delta* on the prompt line.

   If a marker is placed, the delta time between the current query cursor and the marker is displayed. This useful for measuring the relative time between two events.

There are several other options for selecting new times or waveforms.

- To select a new Query time without changing the selected waveform, click or drag the mouse in the time scale region.
- To select a new waveform without changing the Query time, click or drag the mouse in the names region.
- Click on a net of the schematic while in the **Tools->Probe Item** command of the Hierarchy Navigator. This adds the waveform to the display if it is not already present. It also makes the waveform the selected waveform.
- Any of the jump commands can be used to change the selected wave or time.

Selecting Waveforms

One or more waveforms must be selected before the following **Edit** commands can be activated.

- **Edit->Cut**
- **Edit->Copy**
- **Edit->Clear**
- **Edit->Duplicate**
- **Edit->Expand Bus**
- **Edit->Create Bus**

Selected waveforms are indicated by the name field being highlighted like the *RESET* waveform in Figure 8-9. Use the following procedure to select a single waveform.

1. Select the **Misc->Query** command.
2. Click the mouse over the target waveform to be selected.

Multiple waveforms are added to the selected set by holding down the *Shift* key while clicking the mouse over the names of the other waveforms to be added.

A number of adjacent waveforms are selected at one time using the following procedure.

1. Select the **Misc->Query** command.
2. Click the mouse over the name of the waveform to be added.
3. Drag the mouse in the name field until the cursor is over the last waveform in the set to be added. All of the waveforms between the initial mouse click and the release point of the drag are added to the selected waveform set.

Moving Waveforms Within the Display

Use the following procedure to rearrange the order in which waveforms appear in the display.

1. Choose the waveforms to be moved using the selection procedure outlined above. The selected waveforms are now highlighted.
2. Click the mouse on the name of one of the selected waveforms. A thin horizontal appears over this waveform name.
3. Drag the mouse to the new location where the waveforms are to be moved. Releasing the mouse adds the selected waveforms at this point on the display.

Finding Nets in the Schematic

Misc->Query is used in conjunction with the **Tools->Find Item** command from the Hierarchy Navigator to find the device on the schematic that is driving a particular waveform. The command works as follows:

1. Select the required waveform in the WT using **Misc->Query**.
2. Select **Tools->Find Item** from the Hierarchy Navigator menu.

The Navigator displays the schematic containing the component that is driving the selected waveform. The net associated with the selected waveform is highlighted in the displayed schematic. The Navigator determines if a signal is driving a net by looking at the polarity set on pins tied to the net. Any pins with output polarity are considered to be driving. The first such pin fond is displayed when using **Misc->Query**.

Query in the WT works in conjunction with the **Misc->Query** command in the Navigator if the Query box in the Navigator is open. If it is, then the Navigator Query box reflects the currently selected waveform in the WT.

See Also

- **Misc->Set Marker** makes relative measurements.
- **Tools->Probe Item** in *ECS Command Reference* section adds waveforms from the schematic.
- **Tools->Find Item** in the *ECS Command Reference* section is used to display a node in the schematic when you select the waveform using **Misc->Query**.

Misc->Set Marker

The **Misc->Set Marker** command inserts the marker at the current location of the query cursor. The marker is useful as a reference point for measuring times between events.

Use the following procedure to operate the **Misc->Set Marker** command.

1. Use the **Misc->Query** command to position the query cursor where you want the marker to be placed.
2. Choose the **Misc->Set Marker** command from the menu. The marker is made coincident with the query cursor. A dotted line shows the position of the marker. The display does not pan to the location of the newly placed marker.

The marker is a dotted, colored, vertical line that can be placed at any time.

During operation of the **Query** command, the time difference between the current time at the query cursor and the time at the marker is displayed on the prompt line.

See Also

- **Misc->Hide Mark** removes the marker from the display.
- **Misc->Query** is used to measure time delays relative to the marker position.

Misc->Set Trigger

⌘T

The **Set Trigger** command is used to set trigger conditions that detect when the circuit reaches a given state. A trigger is defined by specifying the desired state of one or more signals or buses.

To set a trigger for a signal:

1. Select the **Misc->Set Trigger** command. A dialog box opens as shown in Figure 8-11.
2. Select the waveform for which you wish to set a trigger condition.
3. Click the mouse on the *Add* button of the dialog box.
4. Click the mouse on the required trigger condition.
5. Repeat steps 2 through 4 until all trigger conditions are specified. Bus triggers can be mixed with signal triggers. All active triggers are displayed in the list box.

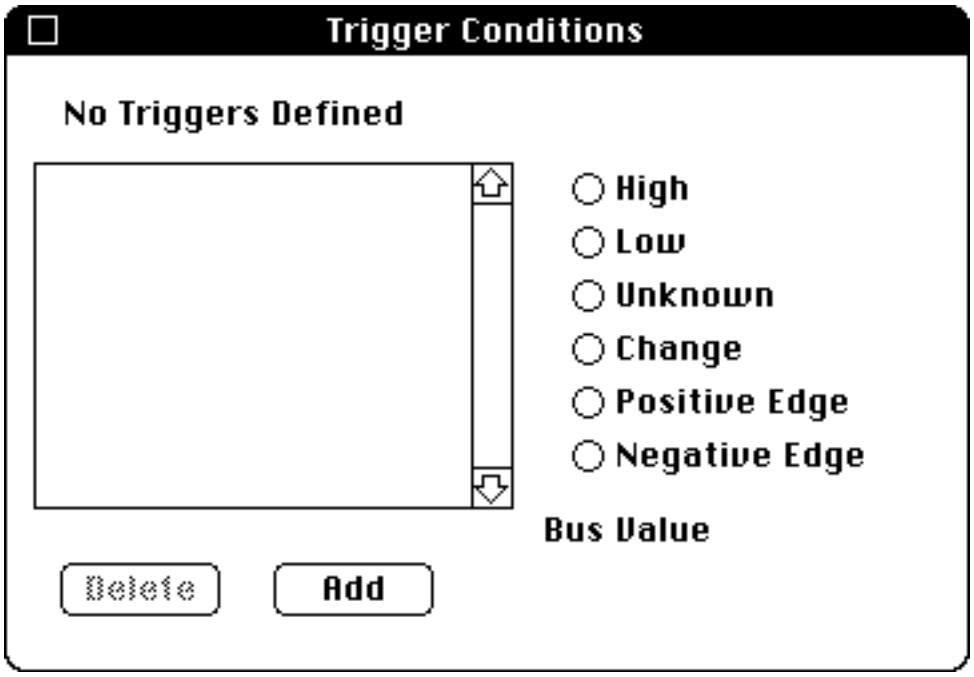


Figure 8-11 Trigger Condition Dialog Box

A trigger condition on a bus is based on the state of each signal in the bus. The trigger condition for a bus has one character for each signal in the bus. The character can be ‘0’, ‘1’, or ‘X’. The ‘X’ character is a *don’t care* or wildcard. Anything will match an X.

To set a trigger for a bus:

1. Select the **Misc->Set Trigger** command. The dialog box shown in Figure 8-10 opens.
2. Click the mouse on the bus waveform that is to have a trigger condition. The bus name and binary value appear in a list box to the right of the dialog box. An edit field named, *Bus Value*, appears at the bottom of the signal conditions dialog box.
3. Edit the *Bus Value* edit field to reflect the binary bus value to be used as a trigger.
4. Repeat steps 2 and 3 until all bus trigger conditions are specified. Bus triggers can be mixed with signal triggers. All active triggers are displayed in the list box.

To remove a trigger condition:

1. Select the required signal from the list box in the set trigger dialog box.
2. Press the *Delete* button in the dialog box.

During the operation of the Query command, the current state of the trigger conditions is displayed on the prompt line.

See Also

- **Jump->Next Trigger** display jumps to the time when the specified trigger conditions are met.

Misc->View Report

The **Misc->View Report** command displays hazard information and/or any other information that is best viewed along with waveforms.

Selecting this command displays a dialog box allowing you to choose the file name to open. The default file name is *design_name.haz*. The file must contains lines with the following format:

```
error_time1    net_name1    Comment about error1
error_time2    net_name2    Comment about error2
```

where *error_time* is a time in integer format and *net_name* is an ECS hierarchical net name and the comment field is any text information. The three fields are separated by one or more spaces.

After opening the file you can select a particular line (typically a hazard or other error condition) from a list box and the Waveform Tool jumps to display the time specified at the beginning of the line. The WT also highlights the specified net if it is contained in the waveform display window. The Navigator, if connected, jumps to highlight the problem net.

The Waveform Tool must have the command line option *-haz* specified in order for the command to be added to the Waveform Tool menu.

Error conditions following the current query cursor time are displayed in the list box by the **Misc->View Report** command. Error conditions prior to the the current query cursor time are pruned from the list box display.

View->Full Fit

Full Fit changes the magnification of the time scale to allow the full duration of any current waveforms to be displayed.

The **View->Full Fit** command is activated as soon as it is selected from the menu.

There is a maximum magnification factor that ECS does not exceed. All other magnifications are multiples of two from this base scale. For example, if the maximum magnification is 1x, the other magnifications are .5x, .25x, .125x, etc.

See Also

- **View->Zoom In** display doubles magnification.
- **View->Zoom Out** display decreases magnification by a factor of two.
- **View->Window** display increases magnification so that selected area fills display.
- **View->Redraw**

View->Redraw

View->Redraw causes the waveform display window to be repainted. This is useful in cases where a specific sequence of commands results in the display not being updated correctly.

To operate the **View->Redraw** command, select it from the menu. The display is redrawn immediately.

See Also

- **View->Full Fit** display decreases magnification until complete waveform is displayed.
- **View->Window** display increases magnification so that selected area fills display.
- **View->Zoom In** display doubles magnification.
- **View->Zoom Out** display decreases magnification by a factor of two.

View->Window

View->Window increases the current drawing magnification, showing a portion of the current drawing in more detail.

There are two modes for **View->Window**, a click mode and a drag mode. Use the following procedure to invoke the **View->Window** command in *Click* mode.

1. Select the **View->Window** command from the menu.
2. Click the mouse at the point on the display you wish to view in more detail. The magnification increases by a factor of two and the display is centered in time about the point where you clicked the mouse.

Use the following procedure to use the **View->Window** command in *Drag* mode.

1. Select the **View->Window** command from the menu.
2. Press the mouse at the first time point you would like to view.
3. Drag the mouse to the second time point and release the mouse. The time span between the two points is highlighted on the display's timescale. ECS centers the chosen time slice inside the current window and adjusts the magnification to the largest possible value that still allows the complete time slice to be viewed.

There is a maximum magnification factor that ECS does not exceed. All other magnifications are multiples of two from this base scale. For example, if the maximum magnification is 1x, the other magnifications are .5x, .25x, .125x, etc.

- **View->Zoom In** display doubles magnification.
- **View->Zoom Out** display decreases magnification by a factor of two.
- **View->Full Fit** display decreases magnification until complete waveform is displayed.
- **View->Redraw**

View->ZoomIn

View->Zoom In increases the current window magnification on the time axis by two times, showing a portion of the current waveforms in more detail.

The **View->Zoom In** command is activated by selecting it from the menu. The waveform display is immediately redrawn at twice the magnification. The new display is centered at the same point in time.

There is a maximum magnification factor that ECS does not exceed. All other magnifications are multiples of two from this base scale. For example, if the maximum magnification is 1x, the other magnifications are .5x, .25x, .125x, etc.

- **View->Zoom Out** display decreases magnification by a factor of two.
- **View->Full Fit** display decreases magnification until complete waveform is displayed.
- **View->Window** display increases magnification so that selected area fills display.
- **View->Redraw**

View->Zoom Out

K

View->Zoom Out decreases the current window magnification on the time axis and displays more of the current waveforms in less detail.

The **View->Zoom Out** command is activated by selecting it from the menu. The waveform display is immediately redrawn at half the magnification. The new display is centered at the same point in time.

There is a maximum magnification factor that ECS does not exceed. All other magnifications are multiples of two from this base scale. For example, if the maximum magnification is 1x, the other magnifications are .5x, .25x, .125x, etc.

- **View->Zoom In** display doubles magnification.
- **View->Full Fit** display decreases magnification until complete waveform is displayed.
- **View->Window** display increases magnification so that selected area fills display.
- **View->Redraw**