ValidGED and SYSTEM UTILITIES REFERENCE MANUAL

Manual Number: MN220 Rev A

10 March 1986

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371 9004

GRAPHICS EDITOR REFERENCE MANUAL

10 March 1986

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371 9004

TABLE OF CONTENTS

INTRODUCTION

The Graphics Editor, GED, is the primary interface between you and your VALID LOGIC DESIGNER. GED allows you to represent your logic designs from the initial concept to the completion of the detailed circuit description.

GED features and commands are specially tailored for schematic capture:

- Extensive component libraries for the most commonly used logic families are available for access by the editor. GED also provides facilities for designing and creating parts, which can be added to the component library.

- The interconnecting (wiring) of component bodies is done with conventional orthogonal lines. Direct (diagonal) wires are also available.

- A special feature called ''dynamic drag'' allows bodies to be moved in real time; wire connections are maintained when bodies are moved.

- Different versions and rotations of parts are supported.

- Properties can be assigned to objects to specify circuit characteristics.

- Notes can be added to the schematic.

Also, GED is designed for versatility and ease of use:

- A full complement of commands allows you to efficiently enter and modify the schematic.

- Commands can be entered from the keyboard or selected from a convenient on-screen menu.

- Function keys can be programmed to perform frequently used commands.

- Variable scaling, panning, and zooming functions allow you to view precise portions of the drawing.

- The default operations of the editor can be changed to meet your specific requirements.

- UNDO and REDO commands can be used to restore a drawing to any previous state.

- In the case of a power failure, automatic recovery of a drawing can be initiated.

This document describes the features and applications of the Graphics Editor. These sections are included:

- **Section 1: System Conventions** introduces general information about using GED.

- **Section 2: The Editing Environment** explains the file and directory structures of UNIX and describes how to work efficiently in the GED editing environment.

- **Section 3: Creating a Design** explains how to use GED commands to create a schematic.

- **Section 4: Design Techniques** describes methods for using GED to create hierarchical and structured designs.

- **Section 5: Producing a Hardcopy** explains how to make a plot of your schematic.

- **Section 6: Adding Physical Information** explains how to add information about the physical part assignments to your logical design.

- **Section 7: Mixing Text and Graphics** describes how to use GED to produce a mixed text and graphics document.

- **Section 8: Drawing Maintenance** describes the GED facilities for updating drawings and recovering from system failures.

- **Section 9: Command Reference** provides a complete reference to each of the GED commands, listed in alphabetical order.

- **Appendix A: GED Files** describes the format of the files created and used by GED.

- **Appendix B: Hardcopy Fonts** provides illustrations and ASCII codes for the supported fonts.

SECTION 1
SYSTEM CONVENTIONS

The Graphics Editor is used to create logic drawings (schematics) and body drawings (shapes of parts) using a high resolution CRT display, alphanumeric keyboard, and graphics tablet. In addition to creating and modifing drawings, GED interacts with the operating system to retrieve and store drawings.

This section introduces general information about using GED: Sign on and exit procedures, command conventions, the elements of the screen, the cursor controller, and the keyboard.

1.1 GED COMMANDS

Commands are issued to the Graphics Editor using both the menu and the keyboard. These commands place bodies on the drawing, connect pins with wires, add text information (signal names, notes), and manipulate the information contained in the drawing.

GED is mostly case-insensitive and recognizes commands typed in either lower-case or upper-case letters. Exceptions to this rule are noted in this manual.

GED commands are structured so that the system recognizes both the complete word and the smallest unique portion of the command name. For example, the EDIT command can be issued by typing **edit** or **ed**. In this manual, bold face type represents the smallest unique portion of the command name (**ED**IT). That abbreviation can be entered either in upper-case or lower-case letters.

In references to information to be typed on the command line, this manual uses bold face type (**ED**) to indicate literal entries and italics to represent variables, which are replaced by specific values; for example, *drawing_name*. Pressing the ENTER key or a carriage return is often expressed <cr>.

Section 9 contains an alphabetical reference to all the GED commands.

1.2 STARTING AND STOPPING

These procedures describe how to access GED, edit a drawing, and exit from GED.

To begin using GED:

1. Turn on the system and log in.

2. Make sure you are in your own login directory. Type **cd** and press the ENTER key.

3. Type **ged** and press the ENTER key.

   Optionally, you can enter the name of the required drawing after the GED command. Because you are entering the command in the UNIX operating system, enter the full name of the drawing in lower case letters. If the name of the drawing contains spaces and other special characters such as angle brackets (<>) or spaces, enclose the name string in quotation marks.

   For example:

   csh% ged "<scald directory>my drawing.logic.1.2"

   Refer to Section 2 for more information about directory and drawing names.

To begin working on a drawing:

Type **EDIT** *drawingname* (the name of the required drawing) and press the ENTER key.

If the drawing already exists, the Graphics Editor accesses the appropriate file and displays it on the screen. If you are creating a new drawing, a blank page is displayed.

To exit from GED and return to the system prompt:

1. OPTIONAL: Use the WRITE command to save the current drawing (type **WRITE** and press the ENTER key).

2. Type **EXIT** or **QUIT** and press the ENTER key.

   GED displays a message if there are unwritten changes to the current drawing.

   - Retype the command and press the ENTER key to override the warning, discard the changes, and exit from GED.

     or

   - Select or type any other command to cancel the EXIT command.

1.3 THE DISPLAY SCREEN

After you issue the GED command, the machine takes a few seconds to read in the program. Then the cursor, on-screen menu, status line, and command prompt are displayed on the screen.

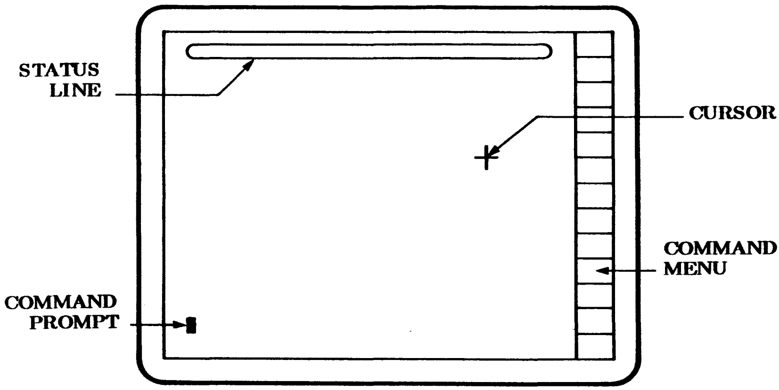


Figure 1-1. CRT Display

CURSOR The cursor appears as a cross on the screen. By moving the mouse, you move the cursor and can select a command from the menu or an object to be changed by pointing to the item and pressing one of the buttons on the mouse. You also use the cursor to draw lines, position library parts, and move items on drawings.

MENU The menu of most frequently used commands is displayed along the right side of the screen. Most of the command names are self explanatory. The semicolon (;) is used to end commands. The last box on the menu is a free box where commands issued from the keyboard are displayed. The last command issued is highlighted to remind you which command is currently active.

STATUS LINE

The status line is displayed at the top of the screen. This status line displays the name of the drawing currently being edited, the grid setting, and the name of the current working directory.

COMMAND PROMPT

At the bottom of the screen is the command prompt. Characters are displayed on this line as they are entered from the keyboard. If necessary, use the Back Space key to back up and retype misspelled words.

PROGRAM MESSAGES

GED displays a message when it cannot interpret or perform a command. GED also informs you when some operations, such as WRITE, GROUP, and CHECK, are complete. Messages are displayed in the upper left portion of the screen. To clear the error messages, select the **WINDOW;** command to redraw the screen.

GRID

GED uses a grid to define where objects can be placed on a drawing. When you first access GED, the grid is turned off. To display the grid, type **GRID** and press the ENTER key.

The default grid setting is displayed on the status line: **0.1 5**. The first number represents the grid size; there is a grid point every 0.1 inch. The second number represents the grid multiple that is displayed, every fifth grid point. The GRID command changes the way the grid is displayed.

Components and wires can only be added and connected at grid points. Valid component libraries depend on the default grid size to function properly. Do NOT change the default grid size; bodies could be placed off the grid and wires would not be connected.

1.4 KEYBOARD

The keyboard is a standard alphanumeric keyboard with up to 10 programmable function keys. The primary purpose of the keyboard is for typing commands, signal names, properties, notes, and other text information required to create a drawing. In addition, the programmable function keys allow you to perform complex operations with a minimum number of key strokes.

Default function key assignments for commonly used GED commands are supplied with the system.

Use the ASSIGN command to change default values and program additional function keys. To see the function key values programmed for the system, type **SHOWKEYS** and press the ENTER key.

Table 1-1. Default Function Keys

Key	Command	Description
PF1	not assigned	
PF2	WINDOW FIT	Redisplays drawing to fit the screen
PF3	DISPLAY BOTH	Displays the name and values of selected properties
PF4	SHOW ATTACH	Displays the attachments between properties and objects
PF5	WINDOW;	Refreshes the screen
PF6	SHOW PROP	Displays the name and values of properties
PF7	DIRECTORY	Lists the drawings in the current directory
PF8	DISPLAY 1.25	Enlarges the selected text 25%
PF9	DISPLAY 0.8	Reduces the selected text 80%
PF10	not assigned	

1.5 CURSOR CONTROLLER

The cursor controller consists of a mouse and a graphics tablet. The graphics tablet can be placed on the table where the CRT display and keyboard are mounted or anywhere else that is convenient. The mouse is used to move the cursor on the CRT and to select and execute commands.

When you enter GED, you must initialize the mouse. To initialize the mouse, move it clockwise in a circle three or four times until the cursor begins to follow the movement of the mouse. The circle should be several inches in diameter. Once the mouse is initialized, the cursor follows the movements of the mouse.

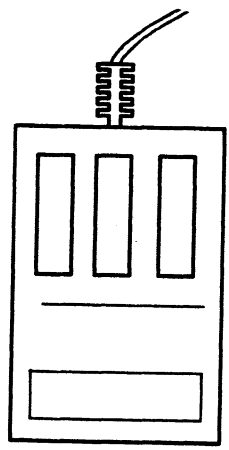


Figure 1-2. Graphics Mouse

There are three buttons on the mouse that are used to signal to GED that a command is to be selected or executed. When one of the buttons on the mouse is pressed, the coordinates of a point on the drawing are reported to GED. This is the normal mechanism for indicating ''points'' to GED commands. The interpretation of the point depends on the position of the cursor and the particular button that is pressed.

The left and center buttons use the nearest grid intersection as the point for the operation being performed. The right button refers to the vertex, or attachment point, of the nearest object.

For example, if a wire is started with the left button, GED places the beginning of the wire at the grid point nearest to the cursor. To start a wire on a vertex, use the right button. One press of the right button snaps the wire to the nearest vertex of an object or wire endpoint.

The center button is used to change the direction of wire. To change the direction of a wire, press the center button.

The remaining distinction between the center and the right and left buttons involves operations on groups. The center button operates on defined groups; the right and left buttons operate on individual objects or points.

Table 1-2: Mouse Operations

CURSOR CONTROL BUTTONS	
LEFT (Yellow)	Selects commands Selects items to be edited Attaches items to nearest grid point Picks up items Starts wires at the nearest grid point
CENTER (White)	Operates on groups Toggles through WIRE options
RIGHT (Blue)	Attaches items at the nearest vertex Picks up items Starts wires at the nearest vertex

1.6 ON-LINE HELP

On-line help is provided to allow you to use GED efficiently. The top command on the GED menu is HELP.

To display help files:

1. Select HELP from the menu.

2. Select a topic from the menu

   or

   Type the topic on the keyboard and press the ENTER key.

When help is displayed, use the WINDOW command to reduce or enlarge the text. For example:

WINDOW 0.75	Reduces the text 25%.
WINDOW 1.25	Enlarges the text 25%.

For more information, see Section 3 and the discussion of the WINDOW command in Section 9.

To display the list of topics for which HELP is available, use one of these procedures:

- Move the cursor to the HELP command and press the left button twice.

- Select HELP from the menu and then type **HELP** and press the ENTER key.

- Select HELP and then select the semicolon (;) from the menu.

To exit from the HELP command, type or select any GED command except WINDOW or the semicolon (;).

SECTION 2
THE EDITING ENVIRONMENT

The SCALDsystem maintains a design database where information about drawings is stored and accessed. The way GED locates and stores drawings in the UNIX operating system makes up the editing environment. This editing environment consists of several elements used to create, store, and manage drawings.

This section explains the directory system, file names, types of files, and how to work efficiently in the GED editing environment.

2.1 USER ACCOUNTS

Drawings and their related files are stored in UNIX directories. GED automatically manages file storage and retrieval operations.

A user account consists of a home UNIX directory containing a startup.ged file as well as several other command and data files that are used by Valid tools. These files provide access to the other SCALDsystem design tools:

case.dat	This file is used to enter data for the Timing Verifier to test the timing for specific cases.
compiler.cmd	This is the command file for the Compiler.
delay.dat	This file is used to enter delay data to the Timing Verifier to test for specific cases.
packager.cmd	This is the command file for the Packager.

simulate.cmd	This is the command file for the Simulator.
startup.ged	This file is used to specify the libraries and SCALD directories to be used for a design project. You can also add ASSIGN commands to establish function key operations and SET commands to determine the default options used by GED.
td.cmd	This is the command file for the PLOTTIME program, which plots timing diagrams.
verifier.cmd	This is the command file for the Timing Verifier.

Also listed in the home directory are the names of existing drawings. These names appear with no extension. Each is a separate UNIX directory containing the files required by GED and the other SCALDsystem tools to correctly interpret the drawing. Access to these files is supported by GED. *Do not change any of these files.*

Additionally, a SCALD directory is listed in the home UNIX directory with a .wrk extension. The SCALD directory is an index file that GED creates for its own use. Each time you create and save a drawing, GED creates an entry in a SCALD directory that maps the GED drawing name to the UNIX directory where the drawing is stored. *Do not edit this file.* If you do, GED cannot locate drawings.

The startup.ged file contains the command **use** *username*.wrk. This creates the SCALD directory, *username*.wrk, the first time you write a drawing. The USE command is also located in other command (.cmd) files to specify the SCALD directory where the drawing reference is contained. If you have more than one SCALD directory in a UNIX directory, you have to edit the command files to include the required drawings to be referenced by other SCALDsystem tools.

To effectively use the SCALDsystem, keep each design with all its related drawings in a separate UNIX directory. Each UNIX directory should contain one SCALD directory with the drawings for a single project, a startup.ged file, and a set of command and data files for each design project.

CREATING A UNIX DIRECTORY FOR A DESIGN PROJECT

To create a directory for a new project:

1. Use the UNIX command **cd** to move into the home directory.

2. Type **mkdir** and the name of the new directory. This creates a new directory under the directory.

   For example: **mkdir proj1**. The full pathname is /u0/*username*/proj1.

3. Copy the default files into the new directory with the command:

   cp *.cmd startup.ged case.dat delay.dat *newdirectory.* (Substitute the actual name of the new directory for *newdirectory.*)

   This copies all the command files (files with the extension .cmd) and the files startup.ged, case.dat, and delay.dat into the new directory.

4. Use the **cd** command to go to the new directory. For example, **cd proj1.**

5. Edit the startup.ged file (**vi startup.ged**).

   - Change the first line (use *username*.wrk) to read **use** ***newdirectory*.wrk** (**use proj1.wrk**).

   - Add library commands for the libraries required by this project.

- Save the new startup.ged file (type **ZZ**).

6. Edit the other command files and change the name of the SCALD directory to the name of the new directory. For this example, in the compiler.cmd file, change the DIRECTORY directive to **directory proj1.wrk**.

When you log in, connect to the new directory (**cd newdirectory**), type **ged** and edit the first drawing. When you save the drawing, an entry is made in the new SCALD directory (proj1.wrk) as specified in the startup.ged file.

2.2 SCALD DIRECTORIES

The SCALD directory is the filing system that GED uses to store drawings in the UNIX operating system. Each drawing is stored in a separate UNIX directory with the group of files containing information about the drawing.

Table 2-1. Drawing Files

ASCII	Contains the graphic information about the drawing in readable form.
BINARY	Contains the graphic information in binary form.
CONNECTIVITY	Contains information about the parts and interconnections in the drawing; used by the Compiler.
DEPENDENCY	Contains a list of all the parts upon which the drawing is dependent. This information is used during the update procedure. See Section 8 for more information.

Although GED places no restrictions on drawing names, the operating system used to support the SCALDsystem has file naming conventions that prohibit the use of the

GED drawing name as the name of the physical file to store the drawing. The solution is an index file, called the SCALD directory, which maps GED drawing names to physical file names (actually UNIX directories).

The UNIX directory, where drawings are stored, is created automatically by the Graphics Editor. The name of the UNIX directory is the GED drawing name, automatically shortened to 14 characters with special characters removed.

SCALD directories are given special "types" that identify the function of the directory; for example, LOGIC, TIME, SIM, and SPICE. Designs are usually developed in a LOGIC directory. The other types are generally used for library development.

LOGIC When a new directory is created with the Graphics Editor, it is a LOGIC directory. A LOGIC directory (type = LOGIC_DIR) contains the drawings you create. This is the default directory type. Drawings with any type can be placed in a LOGIC directory.

TIME A TIME directory (type = TIME_DIR) contains drawings that describe Timing Verifier primitives (special parts understood by the Timing Verifier and used to construct timing models). A TIME directory can contain only drawings with the TIME or PRIM types. Timing Verifier primitives are defined by drawings with the .PRIM type. These primitives are predefined within the Timing Verifier and should not be changed.

SIM A SIM directory (type = SIM_DIR) contains drawings that describe Logic Simulator primitives (those special parts that are understood by the Logic Simulator and used to construct simulation models). A SIM directory can contain only drawings with the SIM or PRIM types. Logic Simulator primitives are defined by drawings with the .PRIM type. These primitives are predefined within the Logic Simulator and should not be changed.

SPICE A SPICE directory (type = SPICE_DIR) contains SPICE device primitives. SPICE primitives are understood by the Berkeley SPICE 26 program (for IC simulations) and by the Analog Designer (for analog circuit simulations). These primitives should not be changed.

You can also create special directory types. If, for example, a special purpose simulator is available for which a special set of primitives is needed, a directory containing these primitives can be created and given the name (for example) MYSIM_DIR. Within this directory, drawings with the MYSIM and PRIM types are permitted.

There are two directory types that are forbidden: PRIM_DIR and PART_DIR. The types PRIM and PART have special meanings in the SCALDsystem and directories of these types are meaningless. The libraries supplied by Valid contain a number of directories with special types. Some examples are MCLDL_DIR, LOGCAP_DIR, and TEGAS5_DIR.

```
FILE_TYPE = LOGIC_DIR;
"SIZE SHIFTER" 'sizeshifter';
"LS112" 'ls112';
"LS373" 'ls373';
"SUPER HYPER MUX BOX" 'superhypermuxb';
END.
```

Figure 2-1. Sample SCALD Directory

In the example, the drawing named SIZE SHIFTER is stored in the UNIX directory sizeshifter. Double quotes surround the SCALD drawing names. Single quotes surround the UNIX directory where the drawing is stored.

Each time you save a drawing with the WRITE command, GED creates an entry in the appropriate SCALD directory with the drawing name and the UNIX directory where the files for that drawing are stored.

A SCALD directory is automatically created in the current UNIX directory when you save your first drawing with the WRITE command. The name of this SCALD directory is determined by a line in the startup.ged file: **use** ***username*.wrk.**

The SCALD directory name is specified: name.extension. By convention, the name is an alpha-numeric string of eight characters or less that begins with an alphabetic character. The extension is an alpha-numeric string of one to three characters. The extension should be either .WRK (for a user directory) or .LIB (for a parts library). Although the .wrk and .lib extensions are not required, they make the SCALD directory files stand out in the UNIX listing.

The SCALD directory, then, is a name translation file between GED drawing names and UNIX directory names. This translation file provides more flexibility for drawing names and allows you to work efficiently without extensive knowledge of the UNIX operating system.

LISTING DIRECTORY INFORMATION

The DIRECTORY command is used to list information about SCALD directories. You can display the name and contents of the current SCALD directory You can list all SCALD directories and related drawings, or list the contents of a specific SCALD directory. These examples illustrate some of the ways to use the DIRECTORY command:

DIR	Lists all drawing names in the current directory.
DIR *	Same as DIR.
DIR <*>	Lists all directories (but no drawing names).
DIR <time>*	Lists all drawing names in the TIME directory.

DIR ls*	Lists all drawing names beginning with LS in the current directory.
DIR *.body*	Lists all bodies in the current directory.
DIR <*>*	Lists all drawing names in all active directories.
DIR *.*.*	Lists the name, type, and version of each drawing in the current directory.

In the examples the asterisk (*) is used as a wild card character to match other character strings.

CHANGING TO A DIFFERENT SCALD DIRECTORY

The USE command specifies the SCALD directory where GED can locate and store the drawings you edit. You must specify the UNIX pathname of the new SCALD directory. Otherwise, GED creates a new SCALD directory file in the current UNIX directory.

CREATING A SEARCH STACK

When you issue the USE command to change SCALD directories, you create a list of directories called a search stack. You can put more than one USE command in the startup.ged file to create a search stack for GED. The last USE statement in the list or entered at the keyboard determines the current working SCALD directory. The active SCALD directory is listed on the status line at the top of the screen.

The SCALD directories in the search stack tell GED where to look for drawings you want to edit. If you tell GED to use a SCALD directory that does not exist, GED creates that SCALD directory when you save a drawing with the WRITE command.

You also put LIBRARY commands in your startup.ged file to specify which libraries of parts you require for your designs. Since libraries are accessed for using parts in drawings, and are not usually written into, the LIBRARY command positions libraries at the bottom of the search stack.

For example:

These GED commands	Create this Search Stack
use susan.wrk use proj1/proj1.wrk lib tutorial lib lsttl	proj1/proj1.wrk susan.wrk tutorial lsttl

Figure 2-2. Startup.ged File with Search Stack

This search stack allows the user (Susan) to access drawings stored in proj1.wrk and susan.wrk. Susan can also access parts from the tutorial and lsttl libraries. When a new drawing is written, it is stored in the current working SCALD directory. The current working directory is the one on top of the stack (proj1.wrk).

BORROWING A DRAWING FROM ANOTHER USER

By default, GED stores drawings in the directory where each drawing originates. Use the WRITE, USE, or IGNORE commands to change the current directory. This allows you to edit a drawing from one directory and store it in another.

After you edit the required drawing, issue the USE command to change to another SCALD directory. When you write the drawing, it is stored in the current directory.

You can also save a drawing in a different SCALD directory with the WRITE command. This saves the drawing in the required SCALD directory without changing the current working directory. Include the name of the required directory in angle brackets with the WRITE command. This command can also be used to change the name of the drawing. For example, WRITE <*directory*>*drawing name.* The IGNORE command deletes a SCALD directory from your search stack. To borrow a copy of a drawing from another SCALD directory:

1. Enter GED. The current directory (project.wrk) is specified in the startup.ged file.

2. Type **USE /u0/otheruser/otheruser.wrk.** This changes the current directory to otheruser.wrk.

3. Type **EDIT** *drawing.*

4. Type **IGNORE otherproject.wrk**. This deletes otherproject.wrk from the search stack and returns to the first directory (project.wrk).

5. Type **WRITE**. This stores *drawing* in your directory and puts an entry in your SCALD directory, project.wrk.

To see the current search stack, type **DIR <*>**. A list of files similar to Figure 2-3 is displayed.

```
proj1.wrk
user1.wrk
user2.wrk
standard
lsttl
```

Figure 2-3. Sample Search Stack

The UNIX **cp** command is not useful for copying GED drawings. If you copy a drawing with **cp**, no entry is placed in the SCALD directory. GED cannot locate the drawing if you try to edit it.

2.3 DRAWING NAMES

The drawing name identifies a design. Whenever you create, edit, or process a drawing, you specify the drawing by its name. Several GED commands such as EDIT, WRITE, GET, and ADD allow or require the name of the drawing.

Drawing names have four parts in the form: *name.type.version.page*. If you only enter the name of the drawing, GED accesses version 1 and page 1 of a LOGIC drawing.

Table 2-2. Drawing Names

You enter	System assumes
WRITE 32 BIT ALU	32 BIT ALU.LOGIC.1.1
EDIT NAND.BODY	NAND.BODY.1.1
GET REGISTER..2	REGISTER.LOGIC.2.1
EDIT MUX BOX...4	MUX BOX.LOGIC.1.4
ADD NAND	NAND.BODY.1.1*
ADD LS00..2	LS00.BODY.2*

*The ADD command requires the specified drawing to be a BODY drawing.

NAME

The first part of the drawing name is the user-defined identification of the drawing. In general, the name describes the intended function of the object. Some examples are:

ANSI Disk Controller
32 bit alu
LS112
10109
HIGH-SPEED RAM

The name is not restricted to short alphabetic identifiers or to upper-case letters. The name can be up to 255 characters long and can contain any printing ASCII character except the period (.), quotation marks ("), and the tilde (~). A space or blank is also permitted.

TYPE

The second part of a drawing name identifies the particular type of drawing. If the extension is not specified, GED uses LOGIC as the default. Consequently, typing **EDIT 32 BIT ALU** has the same effect as typing **EDIT 32 BIT ALU.LOGIC**. There are six standard drawing types: BODY, LOGIC, TIME, SIM, PART, and PRIM.

BODY — A BODY generally refers to a Library component which you add to a design. Components refer to BODY, SIM, and TIME drawings, which are often in libraries. A BODY can also be the symbolic representation for a drawing used in hierarchical designs to refer to a collection of logic without having to include that logic in a drawing.

LOGIC — A LOGIC drawing is the standard type of drawing created with GED. It is used to define a circuit made up of parts (such as TTL or CMOS). Parts are defined in libraries or in user-created directories. A LOGIC drawing can only contain bodies defined in LOGIC directories. (Bodies defined in TIME or SIM directories cannot be added to a LOGIC drawing.)

TIME — A TIME drawing is used to define a timing model for a part. A TIME drawing can contain bodies defined in a LOGIC directory or a TIME directory. If a body from a LOGIC directory is used, that body can refer only to drawings that use timing verifier primitives; that is, no drawings with Logic Simulator primitives are allowed.

SIM — A SIM drawing is used to define a simulation model for a part. A SIM drawing can contain bodies defined in a LOGIC directory or a SIM directory. If a body from a LOGIC directory is used, that body can refer only to drawings that use Logic Simulator primitives; that is, no drawings with Timing Verifier primitives are allowed.

PART — The PART drawing itself contains no logic; it is a place holder for the association of physical information. The part drawing is used as a pointer to physical information, such as power, cost, size, and weight, that is used by the physical design system.

PRIM — PART and PRIM are the same. For identification, use PART for real parts and PRIM for Timing Verifier and Simulator primitives.

You can also specify other drawing types to match specific design tools in use.

VERSION

The next field, version, is used to identify different symbolic representations of body drawings. If the version is not specified, GED assumes version number 1.

The drawing version selects different versions of a body. For example, a NAND gate has two representations. One of the body drawings is called **LS00.BODY.1**, and the other is called **LS00.BODY.2**.

Figure 2-4. LS00 Body — Versions 1 and 2

When you include the body with the ADD command, specify the version in the name of the body (**ADd LS00..2**). Or, use the VERSION command to display different versions of a body.

You can also create several versions of a drawing (a time model, for example) containing different values and parameters. You can then use select expressions to specify a particular version for an application or process. The Language Reference Manual and the Compiler Reference Manual contain more information about using select expressions.

The version field of the drawing name is NOT used to store revisions of a drawing. Use different drawing names to store various drawing revisions.

PAGE

The final field is used to create drawings that extend over more than one page. Paging is useful when the amount of logic required to define a particular design does not fit on a single page. To begin working on the second page of the current drawing, type **EDIT ...2**. This tells GED to edit page 2 of the drawing. The three dots hold the place of the first three fields, one for each field. The default value of PAGE is 1. The number of drawing pages is not limited.

SPECIFYING DRAWINGS

The most common GED command used to specify a drawing is the EDIT command. After you enter GED, type **EDIT** and the name of the drawing to edit. If the drawing exists in the SCALD directories in the search stack, GED displays the drawing on the screen. If the drawing does not exist, GED displays a blank screen where you can begin the design.

You can edit a second drawing without writing the current drawing. EDIT saves the first drawing in a temporary file and then displays the new drawing. You can re-edit the first drawing or use the RETURN command to display the previously edited drawing. The SHOW HISTORY command lists the drawings edited during the current session.

If you make some changes to a drawing and then decide to go back to the original version of the drawing and start again, use the GET command. This replaces the drawing you are editing with the copy that is stored on the disk. Type **GET** and the name of the drawing.

SAVING A DRAWING

To save the drawing on disk, use the WRITE command. You can specify the SCALD directory and the name of the drawing to be used for storing the file. If you are saving a newly created design, GED stores the drawing in the current directory. GED stores a drawing you borrow from another directory in the directory from which it was retrieved unless you delete the directory with the IGNORE command or specify another SCALD directory with the WRITE command.

RENAMING A DRAWING

The DIAGRAM command changes the name of a drawing. Issue the DIAGRAM command from GED after you edit the required drawing. When you WRITE the drawing, a copy is saved under the new name. The original copy of the drawing is also saved.

DELETING A DRAWING

To delete a drawing, use the REMOVE command. When you specify a drawing name with REMOVE, GED displays the names of the default LOGIC drawing along with the names of the associated UNIX files. To procede with the delete operation, type a semicolon (;).

You can delete specific drawings by issuing the entire drawing name. For example:

REM CIRCUIT.LOGIC.1.2	Removes only page 2 of the drawing, CIRCUIT.
REM CIRCUIT.BODY	Removes the BODY drawing, not the LOGIC drawing.

SECTION 3
CREATING A DESIGN

GED provides you with a flexible and easy to use method for entering your designs. A convenient, on-screen menu displays the commands you use most often.

The following rules ensure compatibility between your schematics and other SCALD design tools.

- You cannot write a drawing into a drawing of a different type. For example, if you are editing shifter.logic, you cannot include shifter.sim. You can use the DIAGRAM command to change the name (including drawing type) of a drawing.

- Bodies cannot be added into other body drawings and then saved. Although other bodies can be added to body drawings for comparison purposes or as part of a new body, the Graphics Editor displays an error message if the body drawing is written out. In order to use another device as a base to work from, add the body to the body drawing and then use the SMASH command before you write the drawing.

- You cannot add incompatible bodies to drawings. For instance, simulator primitives (.SIM) are illegal in time drawings (.TIM). Both GED and the Compiler display error messages if you include illegal bodies in drawings. The DIRECTORY command displays information regarding the compatibility of directories for the current drawing.

- You should not use NOT bodies or the Bubble Check facility in drawings to be analyzed by the Analog Designer.

3.1 ADDING DESIGN ELEMENTS

All SCALD drawings are constructed from seven types of objects or primitives: bodies, wires, signal names, dots, arcs, notes, and properties. Each of these items is added to a drawing by a vertex. The vertex is also used to identify an object to be moved or changed. Wires have a vertex at each end and at each bend. A Body has a vertex on the body itself and a vertex for each pin. A text string (signal names, notes, or properties) has one vertex located at the lower left corner of a left-justified text string and the lower right corner of a right-justified text string.

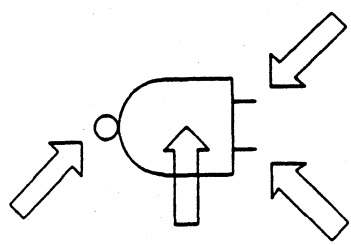


Figure 3-1. Body — Vertices at Center and at Pins

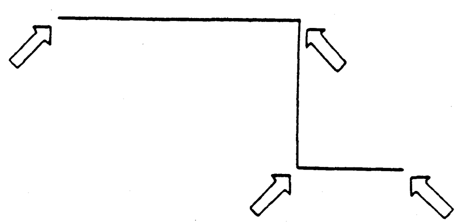


Figure 3-2. Wire — Vertices at Each End and Corner

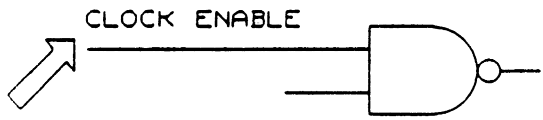


Figure 3-3. Text String — Vertex at Lower Left Corner

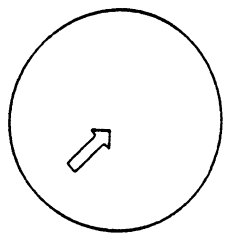


Figure 3-4. Dot — Vertex at Center

Figure 3-5. Circle (360 degree arc) — Vertex at center

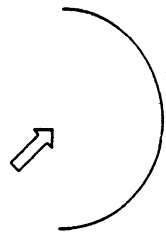


Figure 3-6. Arc — Vertex at Center

ADDING BODIES

A body is the symbolic representation of a drawing. The body drawing can refer to a collection of logic without having to include that logic in a design. Generally, bodies are the components that are defined in the libraries purchased with your SCALD system.

You can also create bodies to represent repeated sections of a design or circuit. Refer to Section 4 in this manual for information about using BODY drawings in hierarchical designs.

In order to add bodies to a drawing, you have to specify the required library with the LIBRARY command. Also, a Standard library is included with GED. This library contains special bodies. Refer to the Library Reference Manual for more information.

To add a body to a drawing:

1. Make sure that the proper library is accessed. Issue the LIBRARY command in GED or include the required library statements in your startup.ged file.

2. Use the EDIT command to begin working on a schematic.

3. Type: **ADD** *bodyname* and press the ENTER key. (Substitute the name of the required component.)

   This command displays a picture of the specified body at the cursor. The ADD command is displayed in the last box on the menu.

4. Move the mouse to position the body on the drawing. The body drags across the screen with the cursor.

5. Press the left button to place the body on the drawing. To add another copy of the same body, press the left button, move the cursor to the new location, and then press the button again.

As long as the ADD command is active (displayed in the last menu box) you can add bodies to the drawing by specifying the name of each body and pressing the left button to place it on the drawing.

Many bodies in the Valid component libraries are represented by more than one version. Body versions support different, but equal representations of a part as well as vectored and non-vectored representations of sizeable parts.

When you use the ADD command to specify the name of the body, GED assumes that you require version 1. You can specify the version (*bodyname*..2) with ADD, or you can use the VERSION command to display the available versions of the body. To use the VERSION command, select VERSION and then point to the part and press the left button.

The BUBBLE command changes the state of a pin from active high to active low. (The library part must be defined to support this feature.) Issue the BUBBLE command and then point to the pin and press the left button.

GED also supports several commands that position and rotate bodies to meet design requirements.

ROTATE	Rotates the body 90 degrees each time you press the button with mirror images of the body at 180 and 270 degrees.
SPIN	Rotates the body 90 degrees each time you press the cursor button.
MIRROR	Creates a mirror image of the selected body about the y axis.

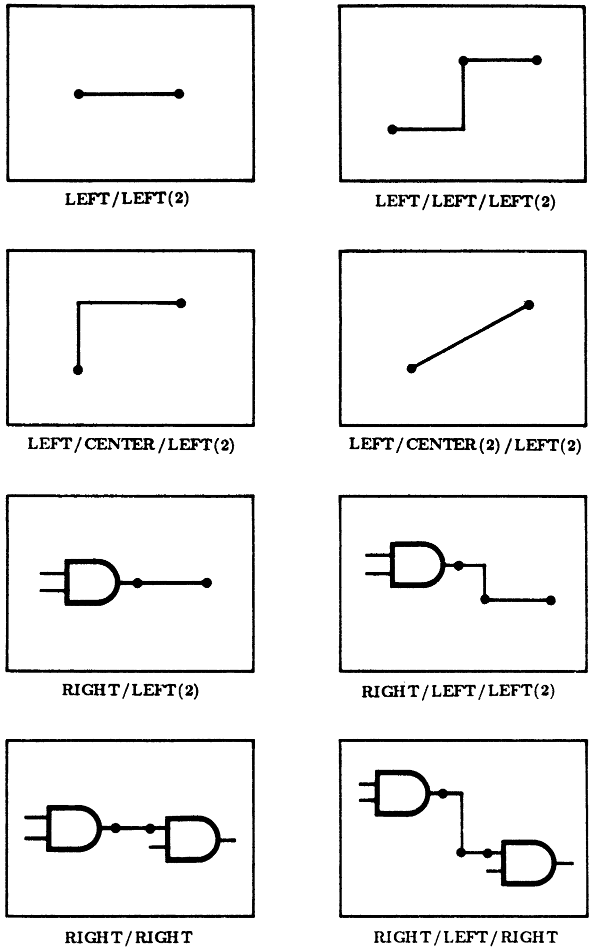
DRAWING WIRES

The WIRE command is used to draw lines to connect the components of a schematic. This command is used with the cursor controller buttons to begin, position, bend, and attach wires where required on the schematic.

LEFT	Starts and attaches the wire at the nearest grid point. This button is also used to place additional bends in a wire. Press this button twice to end a wire that is not attached to a pin.
RIGHT	Starts and attaches the wire at the nearest vertex. Press this button twice to end a wire that is not attached to a pin.
CENTER	Toggles between orthogonal and diagonal wire modes.

Because schematics most often use orthogonal (bent) wires, GED uses orthogonal wires by default. Direct or diagonal wires are also available. Press the center button to change the direction of the bend in the wire or to select the direct wire mode. You can also use the SET command to change the wire mode.

Figure 3-7 illustrates the most common wire shapes and the buttons used to draw them.



Notes:

1. Button click points are shown as a filled dot.
2. Click all buttons once unless indicated otherwise (2).
3. All wiring shown is left - to - right.

Figure 3-7. Wiring Reference Chart

Bus-through Pins

Bus-through pins are special pins placed on a body to make it easier to wire a group of the bodies together. For example, flip-flops can be defined with a bus-through pin on the body exactly opposite the clock input pin. The clock signal can be connected to the clock input pin and a second wire run from the clock bus-through pin to the clock input pin of another flip-flop.

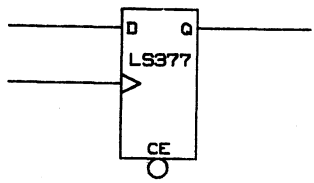


Figure 3-8. Wiring with Bus-through Pins

To connect a wire to a bus-through pin, issue the WIRE command, position the cursor across from the input pin and press the right button. The wire connects to the body. You can look at the drawing of the library part to determine if the part is defined with a bus-through pin. Use the EDIT command to display the .BODY drawing of the part.

DEFINING SIGNALS AND CONNECTIVITY

You can identify each signal of the circuit with a name. Signal names not only identify signals on the drawing, they allow you to enter other information that is interpreted by the SCALD design tools.

Signal name	The signal name is a string of characters that provides a descriptive or mnemonic reference for the signal. All signals with the same name are interpreted as the same signal. This is how signals are connected across pages of a multi-paged drawing.
Assertion level	The assertion level describes the active state of the signal when asserted. By convention, a signal is active high for positive logic and is active low for negative logic. Two signals with the same name, but different assertion levels are NOT the same signal.
Signal bits	Within the SCALDsystem, signals can represent a single bit (scalar signals) or multiple bits (vector signals). The bit portion of the signal name specifies the number of bits (and which ones) the vector signal represents.
Properties	Signals can be given properties that describe characteristics of the signal, control how the signal is interpreted by the Compiler, or convey physical information.

The names and values attached to the signals in a drawing are written into the CONNECTIVITY file that GED creates when you save the drawing. Refer to the Languages Manual for more information on signal syntax.

NAMING SIGNALS

To name the signals on a schematic, use the SIGNAME command. This command adds signal names to the drawing and associates each name with the required signal. After signal names are placed on the drawing, the text strings can be moved without affecting their attachments to signals.

The SIGNAME command allows you to type in several of the signal names for the drawing at one time. Then, point to each signal in turn to attach the required signal name. As you place each signal name, the next one is displayed at the cursor.

Alternately, you can first point to several wires with the cursor. An asterisk or cross marks each position. Then type the signal names in order. The signal names are automatically placed at the indicated positions.

For example, this procedure can be used to name the primary input and output signals of the schematic in Figure 3-9:

1. Select SIGNAME from the GED menu.

2. Type **A** and press the ENTER key. The **A** is positioned at the cursor.

3. Type:

   B <ENTER>
   C IN <ENTER>
   SUM <ENTER>
   C OUT <ENTER>

4. Move the cursor to signal B and press the left button.

5. Move the cursor to signal C IN and press the left button.

6. Move to SUM and press the left button.

7. Move to C OUT and press the left button.

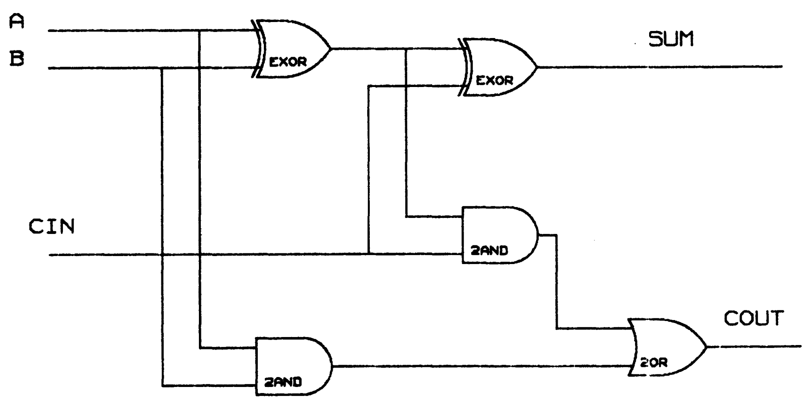


Figure 3-9. Sample Schematic with Signal Names

ADDING DOTS

You can use the DOT command to place dots on the drawing to clearly mark the connection of two wires. In the SCALD system, a T-junction is automatically a connection whether or not it is dotted. A four-way intersection (+) is not a connection unless it is dotted.

Dots are also used to represent the connection points on body drawings.

To use the DOT command:

1. Select the DOT command from the menu.

2. Move the cursor to the required wire junction and press the right button. A dot appears.

To automatically place dots on a complex circuit, use the following procedure with the AUTO DOTS command.

1. Type **SHOW CONNECTIONS** and press the ENTER key. This command places asterisks temporarily on the drawing to highlight each connection point.

2. Check the drawing to make sure that no connections have been made by mistake.

3. Use the refresh command (**WINDOW ;**) to remove the asterisks from the screen.

4. OPTIONAL: Type **SET DOTS_FILLED** to specify filled rather than open dots.

5. Type **AUTO DOTS** and press the ENTER key. All the junctions are automatically dotted.

USING NOT BODIES

The NOT body in the Standard Library supports the Bubble Checker feature of the Compiler. The Bubble Checker verifies that signals and pins are connected only to other signals and pins having the same assertion. The Bubble Checker flags each signal that is connected to another signal or pin of the opposite assertion, and that does not have a matching NOT body. This allows you to catch errors concerning assertion levels. When you intentionally connect a signal of one assertion to a pin of the opposite assertion, add one of the four versions of the NOT body so that the bubble on the NOT body connects with the bubbled (low-asserted) signal.

To add a NOT body to a drawing, type:

1. ADD NOT and press the ENTER key.

2. Place the body on the drawing.

3. Use the VERSION command to select the representation where the bubble on the NOT body faces the bubbled pin.

4. Use the WIRE command to connect the parts.

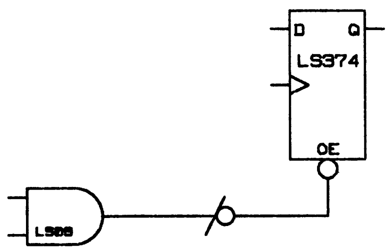


Figure 4-10. Using a NOT Body

The NOT body is seen only by the Bubble Checker; it does not change the assertion of a signal. If the Bubble Checker is turned off (by the Compiler directive), the signals on either side of the NOT body are synonymed together and the NOT body is ignored. See the Compiler Reference Manual for more information.

ADDING PROPERTIES

A property is a name and a value pair which conveys information about a design to the SCALD analysis tools. Properties can be attached to bodies, signals, signal names, and pins. Properties can also be attached to a drawing by attaching them to a special DEFINE or DRAWING body.

GED, in general, has no knowledge of rules about logic design or the ways in which components can be connected together. GED does interpret the following properties:

- Last_modified (on the DRAWING body)
- Pin_Name
- Sig_Name
- Properties added by the BACKANNOTATE, SECTION, and PINSWAP commands

GED uses Pin_Name and Sig_Name properties as part of its treatment of components. All other properties are passed to the other programs in the SCALD system.

The information represented by the properties in a drawing is interpreted by the Compiler. This information is then passed to the other SCALD system tools as well as user-developed programs.

You can use the properties that have been developed for the SCALD system or you can define your own properties. Refer to the Language Manual for more information about properties and property name syntax.

There are two ways of adding properties to a drawing. You can use the PROPERTY command, or you can include a property in a signal name. The meaning of the property is the same, regardless of the method used.

- Body properties are always added with the PROPERTY command.
- Signal properties are usually included in the signal name, but can be added with the PROPERTY command.
- Pin properties are usually included in the pin name, but can be added with the PROPERTY command. A pin property can also be inherited by a pin from a signal connected to the pin.

To add a property to a drawing:

1. Select PROPERTY from the menu.

2. Move the cursor to the object where the property is to be attached and press the left button. An asterisk is placed on the selected object.

3. Type the name of the property and the value of the property on the command line and press the ENTER key. Leave a space between the property and the value. The value appears at the cursor.

   Alternately, you can separate the name and value pairs with an equal sign (=) or by pressing the ENTER key between them.

4. Move the cursor to position the property and press the left button.

By default, only the value is displayed when you add a property to a drawing. The DISPLAY command controls how properties are displayed. After you change the form of a property with the DISPLAY command, it remains in effect until you issue another DISPLAY command.

DISPLAY NAME — This option displays only the name of the property.

DISPLAY VALUE — This command displays only the value of the property (default).

DISPLAY BOTH — This command displays both the name and the value of the selected property.

DISPLAY INVISIBLE — This command displays neither the name nor the value of the selected property.

To change the display of a property:

1. Type **DISPLAY** *option* and press the ENTER key. (Substitute the actual value for *option.*)

2. Point to each property to be changed and press the left button.

For example, you can attach the property ABBREV=SBT to a drawing of a subtractor circuit and display both the name and the value by issuing the **DISPLAY BOTH** command.

You can issue the SHOW command to temporarily display information about the properties on the drawing. After you issue the SHOW command, you can remove the information from the screen with the WINDOW ; command.

SHOW PROPERTIES — This command displays both the name and the value of all the properties (including signal names and invisible properties) on the drawing.

SHOW ATTACH — This command displays the connections between properties and the objects to which they are attached.

DRAWING ARCS AND CIRCLES

You can add both arcs and circles to a GED drawing. This is most commonly used for body drawings where a circle represents a bubbled pin.

To draw a circle:

1. Type **CIRCLE.**

2. Select a point as the center of the circle and press the left button.

3. Select a second point to determine the length of the radius and press the left button. A circle appears.

To draw an arc:

1. Type the CIRCLE command and specify the center point.

2. Select a second point for the length of the radius. A circle appears when you press the button.

3. Move the cursor counterclockwise from the radius point along the circumference of the circle and specify a point to determine the length of the arc.

4. Press the left button to draw the arc.

ADDING NOTES AND DOCUMENTATION

You add annotations and comments to your drawing with the NOTE and FILENOTE commands. The information placed on the drawing with these commands is ignored by the Compiler and the other SCALDsystem analysis programs.

The NOTE command is useful for adding a single line of text to the drawing. The FILENOTE command places the contents of a specified text file on the drawing.

Another way to add information about the drawing is to add a special body called the DRAWING body. The DRAWING body automatically displays the date and time when the drawing was last updated. You can also attach properties to the DRAWING body to add a title and abbreviation to the drawing.

The ABBREVIATION property allows you to specify an abbreviation for the drawing name. The SCALDsystem requires an abbreviation for a drawing in order to identify each signal and library part. If you do not specify an abbreviation, the system automatically creates one.

You can also attach the TITLE property to the DRAWING body to record the name of the drawing on the drawing itself. The drawing name on the status line does not appear on a printed copy of the drawing unless you add the TITLE property. The title must exactly match the GED drawing name.

To add a border around your drawing, you can add a PAGE body. There can be several sizes of PAGE bodies and PAGE bodies that incorporate company logos. Two types of PAGE bodies in the Valid Standard Library are:

A SIZE PAGE	This places a border around an A-size drawing.
B SIZE PAGE	This places a border around a B-size drawing.

3.2 DEFINING GROUPS

Use the GROUP command to define a group of objects. You draw a line around the required objects to identify a group. You can also use the FIND command to define a group of specified objects.

GED assigns a letter of the alphabet as the name of each group and displays the name and the contents of the group on the screen. You can define up to 26 groups in each drawing. Alternately, you can assign the group name.

After the group is defined, use the COPY, CUT and PASTE, DELETE, PAINT, REPLACE, VERSION, and MOVE commands to manipulate the group.

To define a group:

1. Issue the GROUP command.

2. OPTIONAL: Specify the one-letter name of the group. A default one-letter name is assigned if you do not enter one.

3. Use the mouse to draw a line around the required objects. Press the left button to place bends in the line.

4. Close the polygon by pressing the right button when the cursor is near the starting point.

All of the vertices within the polygon are included in the group.

You can also use the FIND command to group objects in the drawing. For example, you can color, replace, or version all occurrences of a specified body in the drawing.

To perform an editing operation on a group, you select the group by typing the group name or by pressing the center button when the cursor is near the group.

3.3 USING COLOR

The PAINT command allows you to specify the colors of the objects in your drawing. You can use up to 16 colors in your designs. These colors are preset and cannot be changed.

Red	Orange	Salmon	Aqua
Green	Purple	Violet	Peach
Blue	Gray	Skyblue	Brown
Yellow	White	Pink	Mono

The objects are drawn in the actual colors you specify. You can also use the SHOW COLOR command to display the names of the colors assigned to the objects in your drawing.

When you issue the PAINT command, the on-screen menu of commands is replaced by a list of the available colors. To assign a color to an object:

1. Issue the PAINT command.
2. Use the cursor to select a color from the paint menu.
3. Point to the object and press the left button.

You can also assign a color to a defined group with the command:

PAINT *color "groupname"*

or

PAINT *color pt*

Use the FIND command to define the group of objects.

You can also use the SET command to establish default colors for the objects in your drawings.

3.4 MAKING DUPLICATES

You can make copies of the bodies, wires, properties, and groups in a drawing. This feature allows you to work more quickly and efficiently. By positioning copies of wires, you can achieve consistency and uniformity in a drawing.

To copy an object in the drawing:

1. Select the COPY command from the menu.

2. Move the cursor to the object to be copied and press the appropriate button.

    - The left button picks up a copy of the object at the grid point nearest the cursor.

    - The right button picks up a copy of the object at the vertex nearest the cursor. (Useful for copying bodies.)

3. Move the copy to its location and press the appropriate button.

    - The left button places the copy on the grid point nearest the cursor.

    - The right button attaches the copy to the vertex nearest the cursor. This is useful for attaching copies of wires at new locations.

To make a copy of a group, use the center button to pick up the group of objects nearest the cursor. Alternately, specify the name of the group when you issue the copy command. Position the copy and then press the left button to place the copy down on the drawing.

You can also make copies of most properties on the drawing. Default properties and user-added body properties are automatically included in copies made of parts. Wire properties are not automatically included when you copy a wire.

You cannot copy default body properties, pin properties, and those properties generated by the SECTION, PINSWAP, and BACKANNOTATE commands.

When you copy a property, the COPY command provides a method allowing you to attach the property to its new location:

1. Issue the COPY command.

2. Move the cursor to the property to be copied and press the left button.

3. Use the cursor to position the copy and press the left button to place the property on the drawing.

   A rubber band line is drawn from the property to the cursor.

4. Move the cursor to the object where the property is to be attached and press the left button.

To copy objects or groups from one drawing to another drawing, you can use the CUT and PASTE commands.

The CUT command places the specified object or group in a buffer. Default body properties and user-added body properties are included in copies made of parts. Path properties, pin properties, and the properties generated by the PINSWAP, SECTION, and BACKANNOTATE commands are not included in the CUT buffer. Wire properties are copied when a wire is cut. This allows you to transfer signal names to the new drawing.

To add the contents of the CUT buffer to a new drawing:

1. Edit the required drawing.

2. Issue the PASTE command.

3. Use the cursor to select the point for the copied material and press the left button.

3.5 MAKING CHANGES

GED provides a full range of editing functions that allow you to correct, modify, and fine tune your design.

While you work on a design, GED maintains an UNDO and REDO log that records the changes you make to the current drawing. If you change your mind about a particular change or a series of changes, you can use the UNDO command to step back through your work. Using the UNDO command, you can back up to the last EDIT or WRITE command. The UNDO command does not undo screen operations and is reset when you edit another drawing.

The REDO command allows you to redo an UNDO operation if you back up too far with the UNDO command.

CHANGING DEFAULT VALUES

Many GED commands have pre-established default values that you can use to create your designs. For example, the DOT command draws open dots and the WIRE command draws orthogonal wires. Issue the SET and DISPLAY commands during an editing session to change the preset default values. Additionally, you can place SET commands in your startup.ged file. Each command has several options to allow you to tailor GED to your particular preferences and requirements.

The DISPLAY command changes the way particular objects appear on the drawing. You specify the object or group to be changed. Once you modify the drawing with the DISPLAY command, the change stays in effect until you issue another DISPLAY command. (The SHOW command has only temporary effects on the drawing.) Refer to the discussion of the DISPLAY command in Section 9 for more information.

The SET command allows you to change general GED default operations. If you issue the SET command during an editing session, you can change the value for that entire session (until you exit). For example, to increase the size of text for one particular drawing, issue the SET SIZE command and specify the size to be used.

You can also place SET commands in your startup.ged file to establish your own default values. For example, if you always use filled dots on your drawings, enter the command SET DOTS_FILLED into the startup.ged file. Or, if you prefer to draw with the grid displayed, enter the command SET GRID_ON into the startup.ged file. Refer to the discussion of the SET command in Section 9 for details.

EDITING TEXT ON A DRAWING

You can use the CHANGE command to edit the text on a drawing. This command allows you to change property names, signal names, and notes. You can use the GED line editor or you can place the text into a file and then use the system text editor to make changes.

GED observes some rules about performing operations on properties. For instance, default body properties cannot be deleted and their names cannot be changed. GED gives an error message if an illegal operation is attempted.

To use the line editor:

1. Issue the CHANGE command.

2. Point to each text string to be changed and press the left cursor button. You can select as many strings as necessary. The first text string is displayed on the command line.

3. Use the line editing functions to modify the text. When you make a change and then press the ENTER key, the text string is repositioned on the drawing and the next string is displayed on the edit line for modification.

You can also select and modify text one string at a time. When you point to a new text string, the current text is repositioned on the drawing and the next string is placed on the command line for editing.

Table 3-1. Line Editor Functions

Keys	Result
Control-F	Moves the cursor forward one character.
Control-B	Moves the cursor back one character.
Control-E	Moves the cursor to the end of the line.
Control-A	Moves the cursor to the beginning of the line.
Control-D	Deletes one character to the right of the cursor.
Control-K	Deletes the remainder of the line (right of the cursor).
Control-I	Displays the HELP file for the line editor.
Control-X	Repositions the text currently on the edit line and displays the next line of text to be edited.
Control-S *character* <cr>	Searches to the right of the cursor for the specified character.
Control-R *character* <cr>	Searches to the left of the cursor for the specified character.
Control-U *number command*	Repeats the command the specified number of times. If no number is given, the default is four.
Control-Z	Aborts changes to the text currently on the edit line and repositions the original back onto the drawing.

To insert text to the right of the cursor, type the characters to be inserted, then press the ENTER key.

To use the system text editor:

1. Issue the CHANGE command.

2. Select the text strings with the cursor.

3. Press the CONTROL-V key combination. This accesses the **vi** editor on UNIX.

   A file containing the text strings you selected is displayed. Use **vi** editing functions to make the required changes. Be sure not to add or delete andy lines from the file. Refer to the appropriate manual for information about using **vi**. When you end the editing session (SHIFT-ZZ), the drawing is redisplayed with the modified text.

SEARCHING FOR PATTERNS

The FIND and NEXT commands allow you to locate all the occurrences of a specified text string. This can help speed up the process of editing similar text strings such as property names and values, notes, and signal names on the drawing.

The FIND command places all the occurrences of the pattern into a group. GED labels the group and lists the number of drawing elements it contains. The NEXT command centers each item on the screen so it can be changed or deleted. Because all the items are placed in a group, you can perform an operation such as REPLACE, , DISPLAY, or DELETE on that group to make a global change to the drawing.

To specify a search pattern:

1. Type **FIND** *pattern* and press the ENTER key.

   The FIND command is not case-sensitive; it does not distinguish between capital and lower case letters. You can use wild card characters in the pattern. An asterisk (*) matches anything, and a question mark (?) matches any single character.

2. Type **NEXT** and press the ENTER key.

   The first occurrence of the pattern is displayed in the center of the screen. You can change the item or type **NEXT** to display the next occurrence of the pattern.

MOVING OBJECTS

There are several commands that let you move and manipulate the objects on your drawing. You can choose MOVE, SPLIT, SWAP, or REATTACH depending on the particular application.

The MOVE command allows you to reposition objects on the drawing. When you move an object or a group, all the connections and attachments on the drawing are maintained. This is a special feature of GED called ''dynamic drag.'' The MOVE command also operates on defined groups of objects. Properties are moved with the objects to which they are attached, or they can be moved independently.

To use the MOVE command:

1. Select or type the MOVE command.

2. Position the cursor at the object to be moved and press the appropriate button.

   - The left button picks up the object at the grid point nearest the cursor.

   - The right button picks up the object at the nearest vertex. This is useful for moving bodies.

   - The center button picks up groups. Alternately, you can specify the group name when you issue the command.

3. Move the object to its new location and press the appropriate cursor button.

   - The left button places the object on the grid point nearest the cursor.

   - The right button attaches the object to the nearest vertex.

Occasionally, items and wires become placed on top of each other when you are working on a design. The SPLIT command is useful for separating objects.

You can also use the SPLIT command to disconnect a wire from one pin and move it to another pin.

When you use the SPLIT command,

1. Issue the SPLIT command.

2. Point to the objects and press the right button.

   One of the objects is attached to the cursor so it can be moved about on the screen.

   To move one of the other objects, point to the objects and press the right button again.

   You can continue to select objects at that vertex until the correct item is selected.

3. Reposition the object and place it down by pressing the appropriate cursor button.

You can also use the SPLIT command to add more wire segments to an existing wire. This is useful for creating orthogonal wires from diagonal wires.

1. Issue the SPLIT command and identify a point along the wire.

   This adds a vertex at the specified point.

2. Move the vertex to its new location and press the left button.

3. Use the WIRE command to add a new section of wire.

The REATTACH command reattaches properties (including signal names) from one object to another:

1. Type **REATTACH** and press the ENTER key.

2. Select the property to be reattached. A line is drawn from the property to the current cursor position.

3. Specify the object that is the new attachment point for the property.

4. If necessary, use the MOVE command to relocate the property closer to its new attachment point.

Default body properties and those produced by the BACKANNOTATE, PINSWAP, and SECTION commands cannot be reattached.

Use the SWAP command to change the positions of two properties or notes. Only two properties or two notes can be exchanged, not a note and a property. To use SWAP:

1. Issue the SWAP command.

2. Point to the two notes or the two properties to be exchanged; press the left button each time.

DELETING OBJECTS

The DELETE command is used to remove unwanted objects, text, and wires from the drawing. You can also use the DELETE command to delete specified groups of objects.

1. Issue the DELETE command.

2. Point to the item to be deleted and press the left button. The object nearest the cursor is deleted.

Alternately, you can specify the group name or press the center button to delete a group.

You cannot delete default body properties or the pin number properties generated by the PINSWAP command.

3.6 VIEWING THE DRAWING

The Graphics Editor manages drawings that can be as large as 64 inches on one side if plotted in a single piece. If that much area is displayed on the screen, the objects on the drawing are so small that they are too difficult to manipulate. GED lets you view that large drawing area through a window. By positioning the window and changing the scale at which images are viewed, you can display anything from a very small portion of a drawing to the entire drawing on the CRT display.

The GED WINDOW command lets you zoom in on part of a drawing, zoom out, pan to different areas, and center the screen around a specified point. WINDOW also reduces and enlarges selected portions of the drawing or the entire drawing. Changing the view of the drawing on the screen does not affect the actual size of the drawing; it allows you to pan and zoom for visual convenience.

PANNING

Panning refers to the process of moving the window to view different portions of the drawing without changing the scale. To do this, issue the WINDOW command and then specify a point to be used as the new center of the viewing area. The drawing remains at the current size, but you see a different view of it.

Use these steps:

1. Select the WINDOW command from the menu.

2. Move the cursor to the place on the drawing to be centered on the screen.

3. Press the left button. An asterisk appears on the screen.

4. Select the semicolon (;) from the menu (or type ; and press the ENTER key).

   The location of the asterisk moves to the center of your screen and the asterisk disappears. The scale of the drawing does not change.

ZOOMING

Zooming in is the process of enlarging a portion of the circuitry to display more detail. It is especially useful for checking the wiring and connections on large drawings. Use these steps:

1. Select WINDOW from the menu.

2. Move the cursor to one corner of the area you want to enlarge and press any button.

3. Move the cursor diagonally to the opposite corner of the area you want to enlarge and press the left button.

4. Select the semicolon (;) from the menu (or type ; and press the ENTER key).

   The selected area enlarges to fill the entire screen.

You can issue another version of the WINDOW command to zoom into a portion of a design. This command allows you to control the amount of enlargement and pan on the display at the same time.

1. Select the WINDOW command from the menu.

2. Move the cursor to the place on the drawing to be centered on the screen and press any button. An asterisk appears on the screen. This first point is labeled C (for center) in Figure 3-11.

3. Move the cursor to the right an inch (or so) and press any button. A second asterisk appears. This second point is labeled 2 in Figure 3-11.

4. Move the cursor to the right for an equal distance (about an inch) and press any button. This third point is labeled 3 in Figure 3-11.

   When you specify the third point, the window is redisplayed with the first point as the new center and the portion of the drawing enlarged by about 200%.

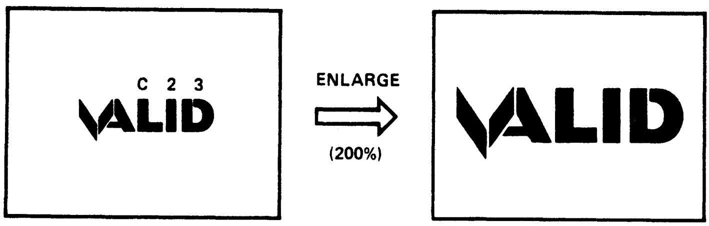


Figure 3-11. Zooming In and Enlarging

Because the distance between point C and point 3 is larger than the distance between point C and point 2, the drawing is enlarged. The enlargement factor is based on the ratio C-3 to C-2. If the distance between center and point 3 is twice as far as the distance between center and point 2, the drawing size is doubled. You can vary the enlargement factor by changing this ratio.

You can also use this form of the WINDOW command to zoom out or reduce the selected portion of the drawing.

1. Select the WINDOW command from the menu.

2. Move the cursor to the place on the drawing to be centered on the screen and press any button. An asterisk appears on the screen. This first point is labeled C (for center) in Figure 3-12.

3. Move the cursor to the right an inch (or so) and press any button. A second asterisk appears. This second point is labeled 2 in Figure 3-12.

4. Move the cursor back to the left to a point about halfway between points 1 and 2 and press any button. This third point is labeled 3 in Figure 3-12.

   When you specify the third point, the window is redisplayed with the first point as the new center and the portion of the drawing reduced by about 50%.

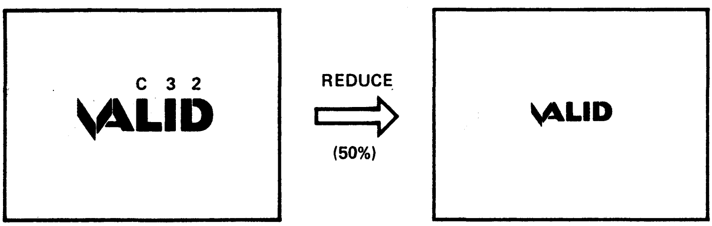


Figure 3-12. Zooming Out and Reducing

Because the distance between point C and point 3 is less than the distance between point C and point 2, the drawing is reduced. The reduction factor is based on the ratio C-3 to C-2. If the distance between center and point 3 is half as far as the distance between center and point 2, the drawing size is reduced by half. You can vary the reduction factor by changing this ratio.

When you reduce the drawing, rectangles are used to replace text that becomes too small to read. These rectangles remind you that there is text at the indicated positions. When you enlarge the drawing, the rectangles are replaced by the actual text.

SCALING THE DRAWING

You can use the WINDOW command to change the displayed size of the entire drawing.

Specify an integer or a real number to scale the view of the drawing by the entered amount. The center of the window remains the same.

For example:

WIN 2; <cr>	Makes the drawing appear twice as large.
WIN -2; <cr>	Makes the drawing appear half as large.
WIN 1.5; <cr>	Enlarges the drawing one and a half times.
WIN 0.5; <cr>	Has the same effect as WIN -2.

The minus sign (-) in front of the scale factor scales the drawing by the inverse of the specified scale factor.

VIEWING THE ENTIRE DRAWING

You can view the entire drawing with the WINDOW FIT command. This option of the WINDOW command scales the drawing to fit into the window area, providing a global picture of the design.

Type **WINDOW FIT** and press the ENTER key to fit the entire drawing into the window. The drawing is enlarged or reduced to fit the window.

3.7 CHECKING FOR ERRORS

After you complete a drawing, you can use the CHECK and ERROR commands to locate connectivity problems and other general errors. These problems are difficult to detect visually and can cause compilation errors.

The CHECK command assigns the path property to all the parts in the drawing and examines the drawing for these errors:

- Wires that are not connected, but appear to be connected
- Pins attached to more than two wire segments
- Duplicate parts in the same location
- Wires connected to only one pin and not named (NC wires)
- Nets that are named, but not connected to any pins

CHECK lists each detected error and its location. After you run the CHECK command, use the ERROR command to locate each error on the drawing.

The ERROR command steps through the errors located by CHECK, centers each error on the screen, draws an asterisk at the location of the error, and displays a message describing the error.

To use these commands:

1. Type **CHECK** and press the ENTER key.

   CHECK processes the drawing and displays the results in the upper left corner of the screen.

2. Type **ERROR** and press the ENTER key.

   ERROR centers the first error on the screen and displays an asterisk and an error message. You can edit the drawing to fix the problem.

3. Issue the ERROR command again to display the next error located by CHECK. You can step through each error to fix all the problems detected by the CHECK command.

SECTION 4
DESIGN TECHNIQUES

This section introduces the design techniques or methods supported by GED and the Valid analysis tools. Depending on your particular needs, one of the following three techniques can best meet your needs:

- Flat designs that can include several drawing pages.

  A flat design is an efficient method for creating a design that is small and does not reuse portions of the circuitry. Flat designs are required for complete backannotation of the design and are more convenient for troubleshooting.

- Structured designs that allow abbreviated bus structures and minimize the required number of parts and interconnections.

  Structured design techniques using the SIZE property support designs that use large bused signals, register depth, and memory depth.

- Hierarchical designs that make use of symbolic representations of circuitry for functions that are repeated throughout a design.

  Large designs that can be broken into functional modules or designs that reuse portions of circuitry can be efficiently created with the hierarchical technique.

Although all designs can be entered as flat drawings, choose the method most appropriate to your particular design. Valid tools are specially designed to operate efficiently with structured and hierarchical techniques.

4.1 FLAT DESIGNS

A flat drawing method is the most straightforward technique for creating a design on the Valid system. In a flat design, all the parts on the drawing come from Valid or user-defined libraries and are one-to-one logical representations for the physical parts. All of the interconnecting wiring within the design is entered pin-to-pin.

Flat designs are best suited for small designs that do not have sophisticated bus requirements or reuse portions of circuitry. Also, if the design must be completely backannotated with pin and physical location numbers, a flat drawing is required.

CREATING A FLAT DESIGN

Both single and multiple-page flat drawings can be created with GED and processed by the Valid design analysis programs.

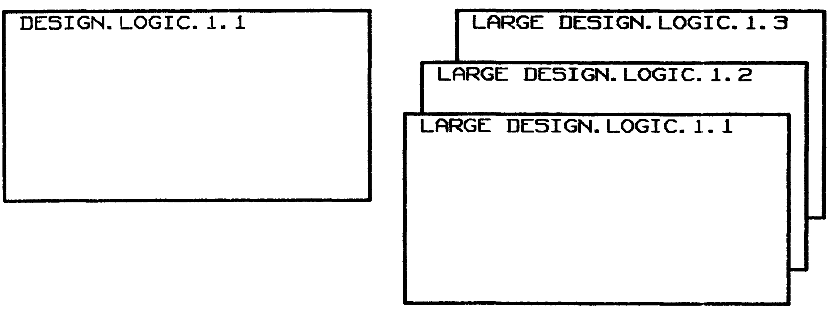


Figure 4-1. Single and Multiple Drawing Pages

Some designs are small enough to fit on one page of a drawing. To create a single-paged design:

1. Specify the drawing name with the EDIT command.

2. Use GED to draw the design on the screen.

3. Use the WRITE command to store the design on the disk.

4. Use the other Valid design programs to compile, simulate, analyze, and package the design.

If the drawing is too large to fit on one page, use the following procedure to create a multiple-paged drawing.

1. Specify the drawing name with the EDIT command and create page one of the design.

2. Use the WRITE command to save page one.

3. Type **EDIT ...2** to begin page two of the drawing.

   This command uses the drawing name, type, and version displayed on the status line.

4. Use the WRITE command to save page two.

5. Create subsequent pages of the drawing in the same way (EDIT...3, EDIT...4).

All pages of a multi-paged design have the same drawing name. The Compiler links all drawings with the same name. If the names are different, each page is treated as a separate drawing.

Give signals that cross page boundaries the same signal name on each page. Signals with the same name have an implicit connection, even if they appear on different pages. For example, the signal SYSTEM CLK on pages 1 and 3 has the same effect as being on the same page and wired together.

BENEFITS OF FLAT DESIGNS

Using a flat design technique has these advantages:

- This technique requires a minimum learning curve and there are few rules and restrictions.

- Since every part and signal is explicitly shown on a flat drawing, pin numbers and physical location designators can be fed back from a physical design system and backannotated onto the schematic. This produces flat print sets with all physical information noted. This is useful for design troubleshooting and is sometimes required by company standards.

CONSIDERATIONS OF FLAT DESIGNS

Keep these considerations in mind when you create a flat drawing:

- Flat designs take longer to create and process than structured and hierarchical designs.

- Flat designs tend to be cluttered and hard to read unless special care is taken to organize and lay out the material.

- Troubleshooting Compiler errors in a large, multi-paged flat design is time consuming and difficult.

4.2 STRUCTURED DESIGNS

The structured design method facilitates the entry and analysis of sophisticated designs that make use of bused signals and memory and register depth. A structured design minimizes the number of interconnections and parts in the design.

CREATING A STRUCTURED DESIGN

You use GED commands to enter and store your drawing. The main difference between a structured design and a flat design is the use of special library parts and the SIZE and TIMES properties.

SIZE Property

The SIZE property is attached to a body and used to specify the width of pin names, signal names, and to define size expansion.

For example, there are two versions of an LS374 octal register in the LSTTL library. The first version is a one-bit slice of the part. It accepts a one-bit D input and produces a one-bit Q output. The second version is the full chip representation of the LS374 with all eight input and output bits explicitly shown.

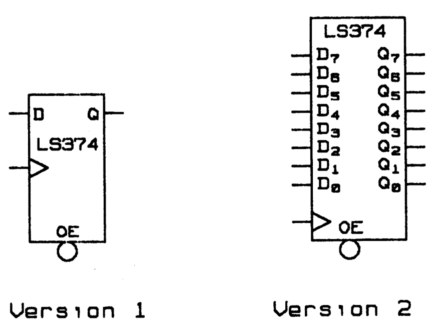


Figure 4-2. LS374 Body — Versions 1 and 2

The first version is ''sizeable;'' you can specify the number of bits the part can represent. Library parts are generally developed with the first version sizeable. The SHOW VECTORS command displays the pin names of a selected part, allowing you to verify that a part is sizeable.

You attach the SIZE property to version 1 of the LS374 part to define the number of bits pins D and Q represent. Valid's signal syntax for bus notation is used to specify a range of bits for the input and output signals. (See the Language Reference manual for more information about signal syntax.)

Figure 4-3 illustrates how you can use version one of the LS374 part in a structured design. In this example, the number of bits is set to 8 (SIZE = 8B); any number of bits can be specified to meet your requirements.

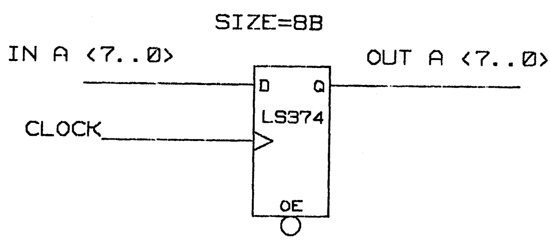


Figure 4-3. Using the SIZE Property to Structure LS374

Version 2 of LS374 is the flat representation of the part. Each pin on the drawing represents a pin on the physical package.

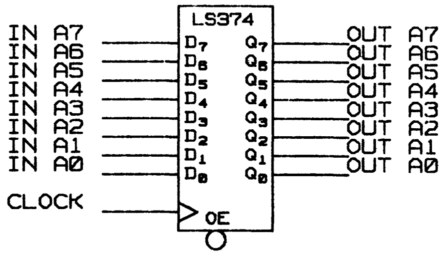


Figure 4-4. Using Version 2 of LS374

Figure 4-5 illustrates the difference between using structured and flat design techniques. Using the SIZE property can greatly minimize the number of parts and interconnections. Also, many possible entry errors are avoided.

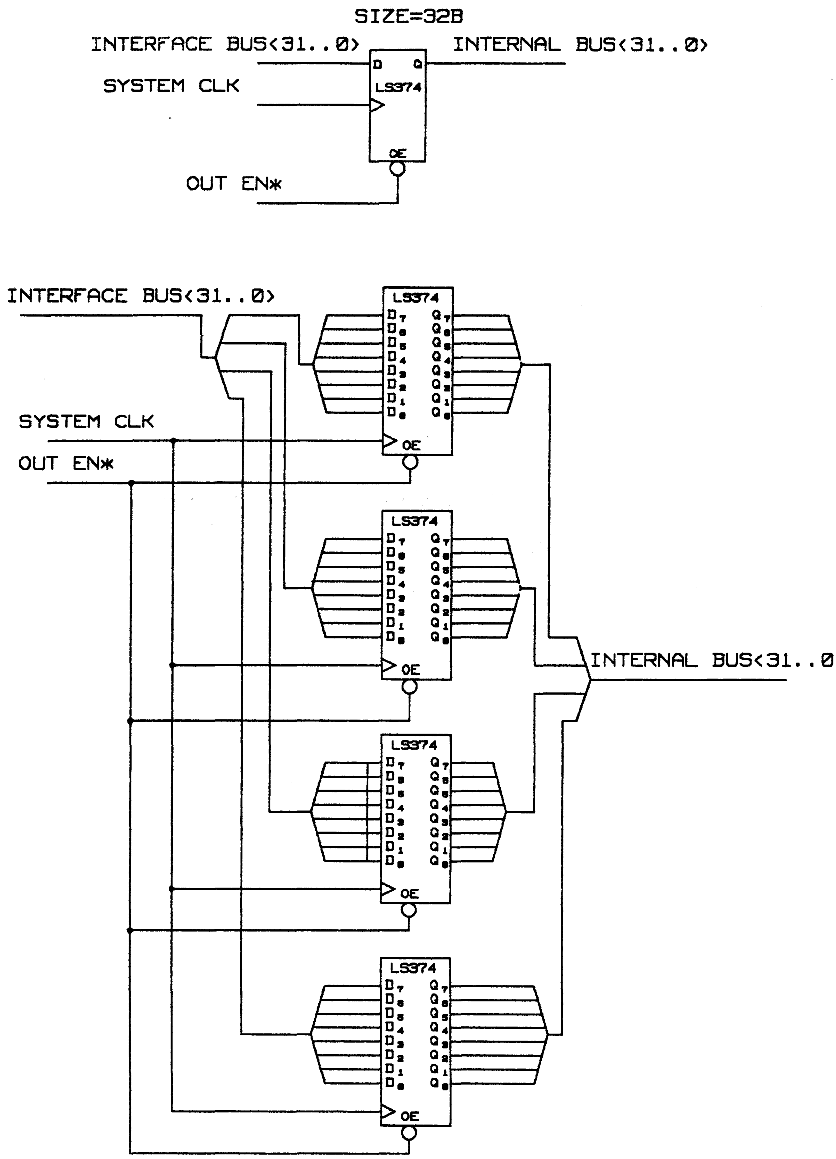


Figure 4-5. Stuctured and Flat Design Techniques

TIMES Property

The TIMES property is used with the SIZE property on structured designs. TIMES allows you to create your structured design to databook specifications. TIMES can be used in cases where the SIZE property causes loading errors. For example, in Figure 4-6, a single part is driving too many inputs on SIZE replicated parts.

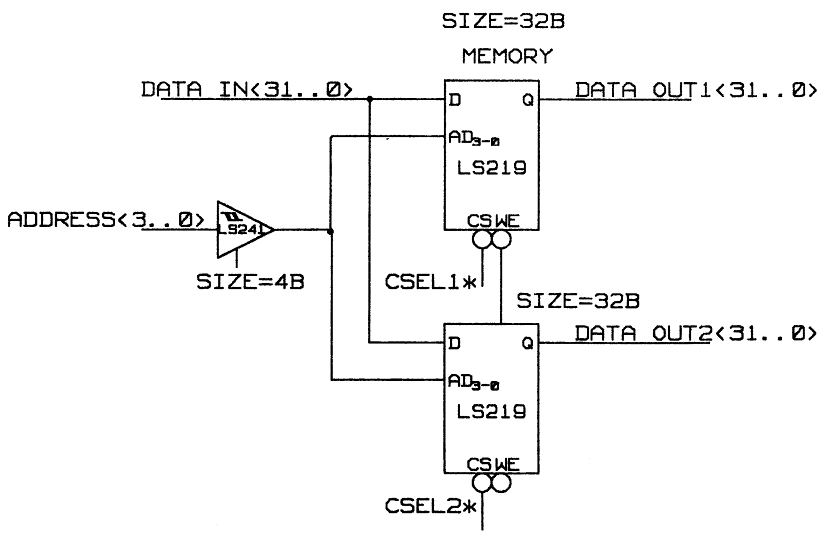


Figure 4-6. Structured Design with the SIZE Property

In this design, the 4-bit tri-state buffer is driving 64 bits of memory. Four sections of an LS241 do not have the drive capability to handle 16 memory packages. The Packager would report a loading error.

The TIMES property is used to correct loading violations in structured designs as illustrated by Figure 4-7.

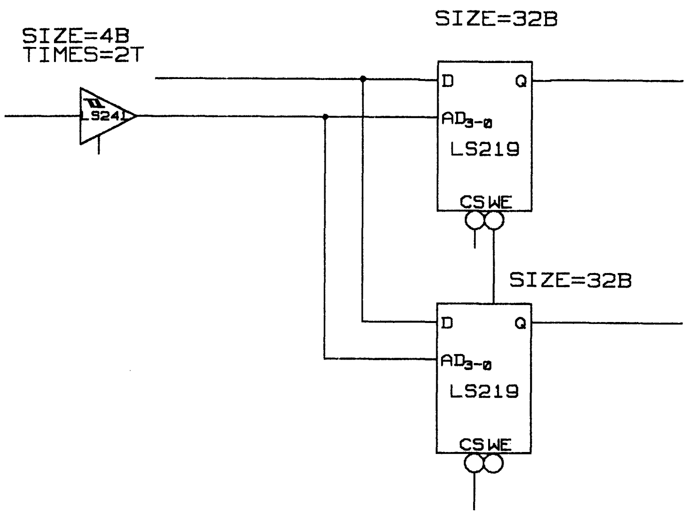


Figure 4-7. Using the TIMES Property

In this example, the TIMES property tells the Valid system that two instances of a 4-bit tri-state buffer are needed. The system checks the loading and balances the load between all the parts being driven. Using the TIMES property in this design is equivalent to adding another part and more interconnections as illustrated in Figure 4-8.

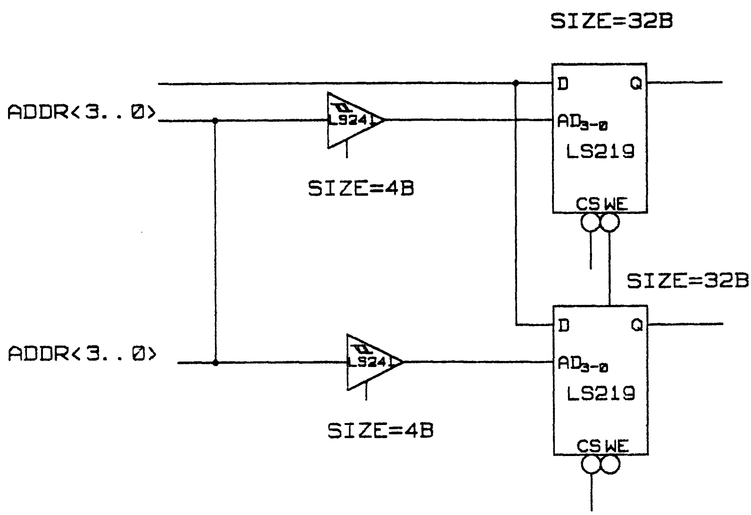


Figure 4-8. Manually Balancing Loads

Using the TIMES property eliminates the need to manually balance the load and enter more data.

The Standard Library

Valid provides a library of standard parts that allow you to define and manipulate signals in a structured design. The Standard library is automatically associated with the list of search directories so you can conveniently use these parts in your designs. Although the bodies in the Standard library can be used for any of the design techniques, many of them are created especially for structured designs.

The library contains A and B sized page borders, merge bodies for merging signals, and tap bodies for tapping bits from busses as well as several other special parts. See the Library Reference Manual for illustrations and descriptions of the bodies in the Standard Library.

BENEFITS OF STRUCTURED DESIGNS

Using a structured design technique has these advantages:

- Creating structured designs can dramatically decrease the design cycle time. The amount of data that is entered in the Graphics Editor is reduced, resulting in faster schematic entry. Also, the analysis tools run more efficiently on structured designs because they can process multiple bits in parallel.
- Errors in design entry are minimized because of the reduced number of parts and simplified interconnections.
- The resulting print is less cluttered, easier to read, and easier to understand.

CONSIDERATIONS OF STRUCTURED DESIGNS

Creating a structured design results in logical representations of parts that represent many physical packages. Therefore, a structured design cannot be entirely back annotated because there is no one-to-one correspondence between the logical and physical designs. Backannotation is performed wherever it is possible. Figure 4-9 illustrates this.

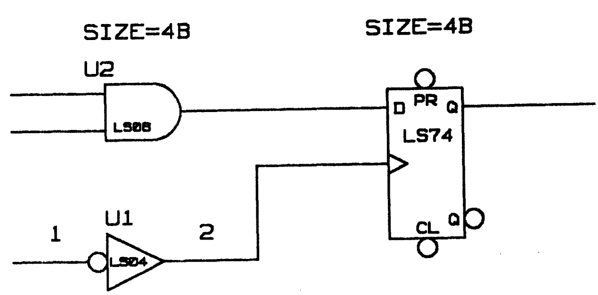


Figure 4-9. Backannotation of a Structured Drawing

Since the inverter represents a single section of a physical package, both the designator and pin numbers are back annotated. The LS08 has four sections per package, so the physical location designator (U2) is back annotated, but no pin numbers are annotated. Since the LS74 has only two sections to a physical package, neither the designator nor the pin numbers are displayed.

The Packager produces easy-to-read cross reference listing for the logical to physical mapping of the design data. These listings are used with the structured print set for design troubleshooting. Members of the design team responsible for troubleshooting the structured design must know how to read structured print sets and how to reference the physical information.

Valid's schematic flattener is an optional utility that automatically ''flattens'' structured drawings into individual wires and components for complete backannotation.

4.3 HIERARCHICAL DESIGNS

The hierarchical design technique is an efficient approach to developing complex designs that can be organized into modules. This method is useful for designs that re-use many of the same circuit functions and for isolating portions of the design for teamwork assignments.

A hierarchical design results in print sets that are easy to read and produces modules that can be effectively debugged. Hierarchical designs, like structured designs, reduce the amount of data entry and interconnections required by the design, thereby reducing the chance for error. Also, all the Valid design tools can be used to analyze partial designs (modules).

CREATING A HIERARCHICAL DESIGN

Creating a hierarchical design is a natural extension of the entire design process. If the design to be implemented is a computer, the design begins by planning the constituent parts of the computer.

The computer can be divided into CPU, MEMORY, and I/O modules. The CPU module can be further divided into ALU, MEMORY, and CONTROL modules. This represents three levels of hierarchy in the design. There are no limits to the number of levels in a hierarchical design.

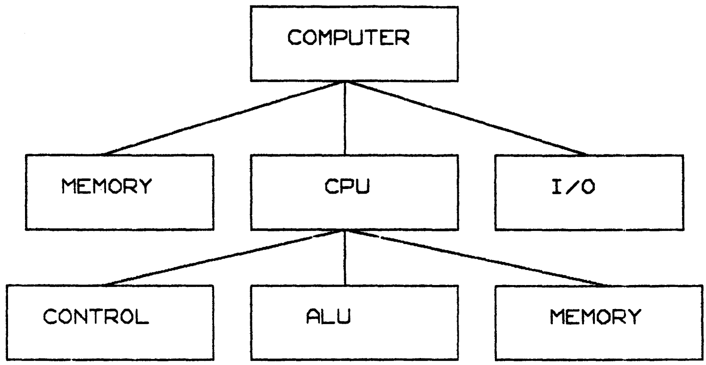


Figure 4-10. Levels of Hierarchy

After the modules of the design are planned, implement the design using the following basic procedure:

1. Create a LOGIC drawing that represents a functional portion (module) of your design (for example, counter, register file, memory unit, or control blocks of circuitry). You can start at the most detailed level of the design hierarchy.

2. Designate each interface signal on the LOGIC drawing with the interface property (\I).

3. Test that drawing, processing it with other Valid programs to check its timing and logic functions. You can efficiently debug each module of the design as you work.

4. Create a BODY drawing to represent the design module.

5. Create a new LOGIC drawing and add the required bodies to it, building a circuit using the modules. You have added symbols that represent the functional module created in Step 1. The BODY drawings act as ''pointers'' to the LOGIC definitions of the circuit.

6. Continue to create the corresponding LOGIC/BODY representations for each of the defined modules in the design, working up the levels of hierarchy.

Figure 4-11 illustrates LOGIC and BODY drawings defined for use in a hierarchical design. Instead of having to wire together the gates of the Full Adder circuit whenever it is needed, you add the Full Adder.BODY drawing in its place.

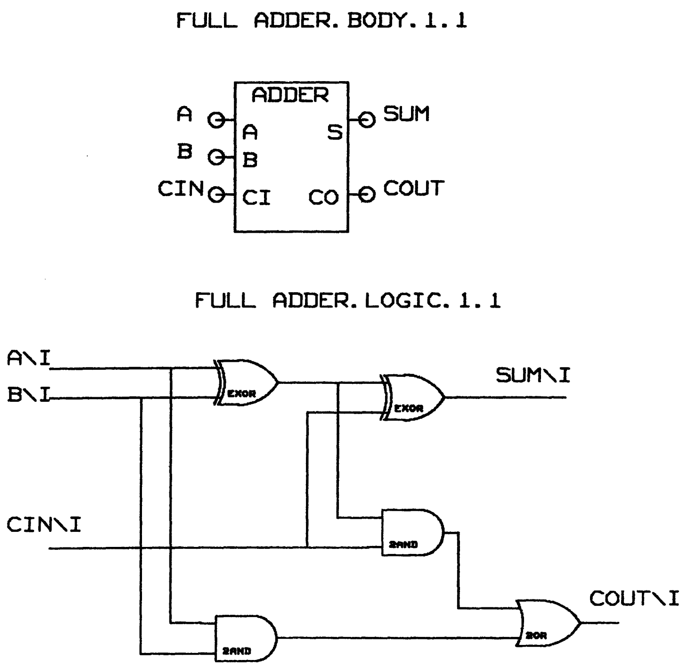


Figure 4-11. Full Adder Logic and Body Drawings

Every level of hierarchy (except the lowest level) is made up of a LOGIC and BODY drawing pair. The LOGIC drawing defines the functional circuitry for the design module. The BODY drawing is the picture or symbol that represents the logic function. The BODY points to the functional representation, but does not take up as much space in higher levels of the hierarchy. The result is a well-organized and understandable design print set.

Creating Bodies

In a hierarchical design, simple shapes, bodies, represent the specific logic for each element of the design. GED provides the tools for drawing bodies and establishing the relationships between the body drawings and the logic drawings they represent. For more information about creating bodies, see the Library Reference Manual.

The interface signals on the bodies are given the ''\I'' interface signal property.

This procedure describes how to use GED to create a body drawing.

1. Edit a drawing with the .BODY extension. For example, *drawing*.BODY.

   A grid is displayed, with an X to mark the origin of the body.

2. Split the name from the origin with the SPLIT command. The origin of the body becomes its vertex when the body is later added to LOGIC drawings.

3. Use the WIRE command to build the outline of the body symmetrically around the origin body. The grid is used as a guideline for the appropriate size and shape of the body. Make sure that the origin is not on a connection point.

4. Add wire stubs for the pins. They should be 0.1 inch (one grid segment) long. Be sure to place the pins on grid lines so that the body can be correctly wired on LOGIC drawings.

5. Use the DOT command to place an open dot at the end of each pin. Dots should be placed on displayed grid intersection points. Use the right button to ensure that the dot is placed at the end of the wire.

6. Use the SIGNAME command to add signal names (corresponding to the signal names in the related logic drawing) to each pin. The name must match the name in the logic drawing exactly except for the addition of the interface property (\I). Use the SHOW ATTACHMENTS command to ensure that all pin names are attached properly.

7. Use the NOTE command to place labels within the body drawing. This makes the purpose of the body and each pin clear.

8. Mark the CLOCK signal with a wedge. Use the WIRE command and press the green button to draw diagonal lines.

9. Use the WRITE command to save the BODY drawing.

Defining Low Asserted Pins

Use a circle instead of a wire to represent a low-asserted (bubbled) pin. The circle should be 0.1 inch in diameter. A dot is placed on the appropriate grid intersection point on the circle to mark the vertex. The signal name should also be low-asserted (*).

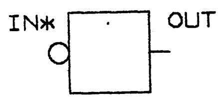


Figure 4-12. Using a Low-Asserted Pin

To define a pin that can be either bubbled or unbubbled, draw a body and represent the pins with both wires and circles. There must be a line that goes across the diameter of the circle.

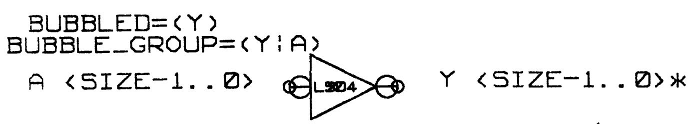


Figure 4-13. Using a Bubbleable Pin

When you add the body to a drawing, use the BUBBLE command to toggle the pin from bubbled to unbubbled.

You can also define groups of pins that automatically change state when one of the pins in the group is bubbled. These are called bubble groups. See the Library Reference Manual for more information.

Creating Body Versions

You can create different versions of your body drawings similar to the versions supported in Valid libraries. The BODY drawings refer to the same LOGIC drawing. Use the EDIT command to begin a drawing for another version of a body: **EDIT CIRCUIT.BODY.2.** Multiple versions can be useful when different sizes of the body are required on the logic drawing or when different, but equal, representations of the body are required.

GED and the SCALD language have several features for creating versions of parts.

- You can make a version that is sizeable.
- You can make a version with bubbled pins.
- You can make a version with pins assigned to bubble groups so that certain pins are automatically bubbled when one pin from the group is bubbled.

See the Library Reference Manual for more information about creating bodies.

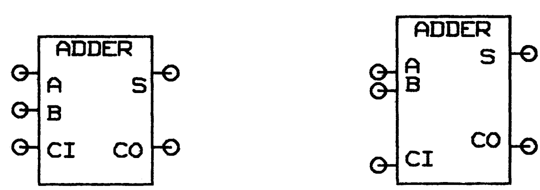


Figure 4-14. Versions 1 and 2 of FULL ADDER.BODY

When the body is added to a drawing, you can use the VERSION command to display the required representation of the body.

BENEFITS OF HIERARCHICAL DESIGNS

The benefits of creating hierarchical designs are similar to those of structured designs:

- Creating hierarchical designs can dramatically decrease the design cycle. Since the BODY drawings act as pointers to the LOGIC drawings, a large amount of repetitious data entry is eliminated.

- The Compiler is optimized to operate on hierarchical designs. The functional LOGIC drawing that a BODY represents can be compiled once and linked to all locations where that body is used.

- Because the amount of schematic entry is reduced, the number of entry errors is minimized.

- Since functional modules are created when defining hierarchy, each module can be fully tested before it is incorporated into higher levels of the design. Testing can be performed incrementally rather than at the end of the design process.

- Hierarchical designs result in designs that are well organized and easy to read and understand.

CONSIDERATIONS OF USING HIERARCHY

Keep these considerations in mind when planning a hierarchical design:

- Hierarchical designs do not have a one-to-one relationship between logical and physical parts. Therefore, a hierarchical print cannot be entirely backannotated.

  The cross reference listings generated by the Packager contain physical information for every part and in your design. The listings and print sets are used for design troubleshooting.

- Members of the design team responsible for troubleshooting the design have to be taught how to read hierarchical print sets and how to reference the physical information cross reference listings.

Valid's schematic flattener is an optional utility that automatically ''flattens'' structured drawings into individual wires and components for complete backannotation.

4.4 COMPARING DESIGN TECHNIQUES

The design methodologies discussed in this section (flat, structured, and hierarchical) are all appropriate for solving different kinds of design problems. The benefits and considerations of each technique must be weighed before you decide which method to use.

There is no restriction against combining these techniques within a single design. Hierarchical and structured design techniques are often used together to provide maximum flexibility and efficiency for the design engineer.

Table 4-1. Comparing Design Techniques

FLAT DESIGNS

Best Suited For:	Small designs Designs that do not reuse modules Designs that do not use busses
Benefits:	Fully back annotated print sets Short learning curve
Considerations:	Long design cycle time Cluttered print sets

STRUCTURED DESIGNS

Best Suited For:	Designs that use sophisticated bus structures
Benefits:	Shortened design cycle time Fewer errors during data entry Less cluttered print sets Print sets organized in logical flow of design Cross reference listings
Considerations:	Partially back annotated print sets Additional training required for design troubleshooters

HIERARCHICAL DESIGNS

Best Suited For:	Designs that reuse modules Large designs that can be organized into separate components
Benefits:	Shortened design cycle Fewer data entry errors Easy-to-read print sets Print sets organized in logical (top down) flow of design Cross reference listings Effective debugging capability
Considerations:	Partially back annotated print sets Additional training required for design troubleshooters

SECTION 5
PRODUCING A HARDCOPY

The HARDCOPY command is used to send drawings to a plotter to produce a hardcopy of a drawing.

Several brands and sizes of plotters are supported:

Versatec 11 — 42 inches

Hewlett-Packard 7475 and 7580

Benson

CALCOMP 1043 and 5744

Epson LQ1500 (3 resolutions modes supported: Low, Medium, and High)

Epson FX80 and FX100

5.1 USING THE HARDCOPY COMMAND

The HARDCOPY command plots files on the specified printer as long as the network and your system have been properly installed. See the *Valid Guide to Operations* for information about setting up and configuring your equipment.

Issue the SET command to specify the device where the drawings are to be sent. Use the SET PLOTTER command to specify an Epson device. If you are plotting on an EPSON LQ1500, use the SET PLOTTER command to select low resolution (LR), medium resolution (MR), or high resolution (HR) mode. The SET and SET PLOTTER commands can be issued during the editing session or placed in the startup.ged file. For example:

SET PLOTTER EPSONLQLR

Specifies a low resolution density (60 dots per inch) on the EPSON LQ1500 printer. This mode produces draft quality plots and is faster than the medium or high resolution mode.

SET PLOTTER EPSONLQMR

Specifies a medium resolution density (90 dots per inch) on the EPSON LQ1500 printer.

SET PLOTTER EPSONLQHR

Specifies high resolution density (180 dots per inch) on the EPSON LQ1500 printer.

SET PLOTTER EPSONFX80

Selects the Epson FX80 as the required plotter.

SET HP7475 or SET HP7580

Selects the specified Hewlett-Packard plotter.

SET W11versatec

Selects the 11 inch versatec plotter. (Must have Ethernet option installed or be in SET SPOOLED_PLOT mode.)

SET B9429

Selects a Benson plotter. (Must have Ethernet option installed or be in SET SPOOLED_PLOT mode.)

SET CALCOMP1043

Selects a Calcomp plotter. (Must have Ethernet option installed or be in SET SPOOLED_PLOT mode.)

Refer to the discussion of the SET command in Section 9 for more information.

After you tell the system which plotter is being used, issue the HARDCOPY command with the required scale factor option and drawing names.

scale factor The scale factor allows you to determine the size at which the drawing is plotted. The default scale factor is 1. You can enter a number or a page size to vary the size of the plotted drawing. If a page size is given (letters A through E), the plot is adjusted to that size. If a real number is entered, the plot is scaled from the normal size.

If you specify a drawing name other than the current drawing, you MUST specify a scale factor (a real number or A - E).

drawing name If no drawing name is given, the current drawing is assumed. The drawing name does not have to be the currently edited drawing nor in the current working SCALD directory. If the drawing is from a SCALD directory other than the current working SCALD directory, the directory name must be given (<dir.wrk>drawing name.logic*).

For example:

HA <cr> Plots the current drawing at the drawing's default scale.

HA A Plots the current drawing on an A size page.

HA C *.logic* Plots all LOGIC drawings in the current directory on C size pages.

HA 1 <100k>100112.body*
Plots the 100112 part from the 100k library with the default drawing size.

HA .5 hyper mux	Plots all drawing types for the drawing ''hyper mux'' (BODY, LOGIC, SIM, etc.) at half size.

5.2 CREATING PLOT FILES

GED provides facilities which allow you to create a plot file that can be printed at a later time or transported to another system that is not on your network.

To create a plot file:

1. Use the SET command to change to SPOOLED_PLOT mode from the default LOCAL_PLOT. (These terms are not descriptive of the actual processes being performed, but have been retained for compatibilty with previous versions of GED.)

2. Use the SET or SET PLOTTER command to specify the device on which the plot file is to be plotted.

3. Issue the HARDCOPY command to specify the drawing to be plotted and any scale or page size options.

This process creates a file called **vw.spool** that is specific to the device specified with the SET or SET PLOTTER command. You can then transfer the file to a diskette or tape to move it to a machine that supports the specified plotter or use the UNIX **lp** command to print the file at another time.

The UNIX **lp** command can be used to plot the vw.spool file. Your system must be configured properly for the command to work. See the *Valid Guide to Operations* for more information.

The syntax for the **lp** command is:

lp -d*devicename* vw.spool

The device names to be used with the **lp** command are:

vers11	11 inch Versatec
vers22	22 inch Versatec
vers36	36 inch Versatec
vers42	42 inch Versatec
Cvers42	42 inch Color Versatec
B9424	24 inch Benson
hp7580	D size HP pen plotter
hp7475	B size HP pen plotter
calcomp1043	E size CalComp pen plotter
calcomp5744	E size CalComp electrostatic plotter

Refer to AT&T's *System Administration Reference Manual* for more information about the **lp** command and the lp spooling system.

SECTION 6
ADDING PHYSICAL INFORMATION

After you complete the logical design, you can add information about the physical part assignments to the drawing. You can include both part reference numbers (U-numbers) and pin numbers on the drawing. The BACKANNOTATE command automatically adds information generated by the Packager and the physical design system. You can also manually add physical information with GED commands.

6.1 BACK ANNOTATING THE DESIGN

Back annotation brings physical design information from the Packager and adds it to the logical design drawings. Generally, you back annotate the design after the first error-free run of the Packager and then again after the design has been processed by a physical design system.

The typical steps in the design and processing of a drawing are:

GED — Schematic capture: The logical design is created.

COMPILE — The design is checked for errors and prepared for other analysis tools.

PACKAGE — The design is prepared for use by a physical design system.

BACKANNOTATE — Physical design information generated by the Packager is added to the design.

physical interface program
: The design is formatted for the physical design system.

physical design system
: The circuit is produced from the design and prepared for manufacturing.

physical feedback program
: The Feedback files from the physical design system are formatted for the Packager.

PACKAGE
: Physical parts are reassigned based on feedback files from the physical design system.

BACKANNOTATE
: The GED drawing is updated to reflect the actual physical design of the circuit.

Physical information is added to a drawing through back annotation as soft properties. This information is interpreted differently than properties added manually to a drawing. GED considers back annotated information to be 'That is, back annotated information does not appear in the SCALDsystem data base; it only resides in the drawings. Back annotated information cannot be manually changed or reattached, but it can be moved or deleted.

Properties added by the BACKANNOTATE command begin with the dollar sign character ($). For instance, a LOCATION property added by the BACKANNOTATE command is represented as $LOCATION. Back annotation adds the $LOCATION, $PN, and $SEC properties to the drawing. The next time you backannotate the drawing, the updated information replaces the existing values (a new $LOCATION value replaces an old $LOCATION value).

One of the files created by the Packager is the backannotation file, called pstback.dat. This file contains the physical part assignments the Packager made in a format that GED can understand. The BACKANNOTATE command automatically adds this information to the drawing. This ensures that your drawing accurately reflects the physical part assignments and saves time and tedious work.

To use the BACKANNOTATE command, you rename the Packager backannotation file (pstback.dat) to backann.cmd, which GED recognizes, and then go into GED and issue the BACKANNOTATE command. Use this procedure:

1. From the UNIX prompt, issue the command to copy a file:

   cp pstback.dat backann.cmd

2. Enter GED and edit the required drawing.

3. Type **BACKANNOTATE** and press the ENTER key.

GED reads the file containing the physical part assignments and automatically adds the information to the drawing.

6.2 MANUAL PHYSICAL PART ASSIGNMENTS

When you use GED commands to make physical part assignments, you are adding hard properties to the drawing. These properties are not altered by the Packager and are not changed when you backannotate the drawing. For example, you can attach the LOCATION property to a body to ensure that the part is assigned to a specific physical designator. The hard properties used to add physical information to the drawing include:

LOCATION	Assigns a particular physical part to a logical body on a design. The LOCATION property can be attached only to bodies that represent physical parts.
GROUP	Groups logical parts to be in the same set of physical packages when you are not concerned about the actual physical designator. Isolates parts of different groups.
SEC	Assigns a logical body to a particular section within a physical part. This is accomplished with the SECTION command.

When you assign physical part information to the logical parts of a GED drawing, you most commonly use the SECTION command and the LOCATION property. When you use the LOCATION property and the SECTION command to make physical part assignments, the Packager and the physical design system do not override the assignments. To change a manual assignment, you have to delete the information or manually assign new information.

Use the PROPERTY command to attach LOCATION properties to the bodies on a drawing. The LOCATION property is not restricted to U-numbers; it can be any alphanumeric string. You can attach a LOCATION property later in the design process, but you must recompile and repackage the design for the Packager to make use of the specified information.

Use the SECTION command to select which section of a physical part is assigned to a particular logical part. To use the SECTION command, edit the drawing and follow these steps:

1. Use the WINDOW command to enlarge the appropriate part of the drawing so that the part is clearly visible.

2. Type **SECTION** and press the ENTER key.

3. Point to the origin of the body you want to assign a section and press the left button. To assign the sections of a full-chip representation of a part, point to a particular pin (not to the origin of the part).

   Each time you press the button, you select a different section of the physical part. The pin numbers on the entire body change accordingly.

Alternately, you can issue the SECTION command, point to a pin, and then type in the required pin number. This can save the time of cycling through the various sections, if you know exactly which section is required.

You can use the SECTION command on a body either before or after you compile and package the design. When you change section assignments after back annotation, you assign just the sections you want to force and leave the others. The schematic may then have some duplicate section numbers. When you recompile and repackage the drawing, the Packager reassigns the remaining sections.

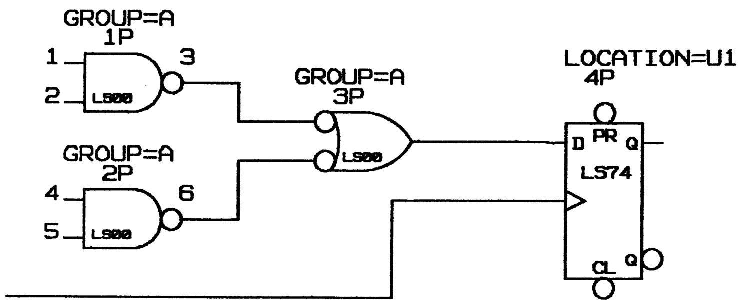


Figure 6-1. Adding Physical Information

In this figure, the properties that have been added have the following effects.

LOCATION The LS74 has been given the physical location designator, U1. Other LS74s that might occur in this drawing would also be placed in this package to fill it.

GROUP The LS00s are all placed in the same physical location designator because they have the same group name. The designator name is assigned by the Packager. Another LS00 in the drawing that has a different group name or no group name is not placed into the same package. You can, however, use the FREE GROUPING directive of the Packager to allow parts with no group name to be placed in a package with parts that have a group name. See the Packager Reference Manual for more information.

SEC The SECTION command was used on two of the LS00s to assign the exact section to those logical parts. The LS00 at 3P is assigned to a section by the Packager.

THE CHIPS_PRT FILE

Only parts having a chips_prt file can be sectioned and pinswapped. The CHIPS_PRT files for each part in a library are made by dividing the library chips file (*lib.prt*) into individual files for each part.

To divide a chips file, use the UNIX command:

% /usr/bin/makechipsfiles *chips_file library_name*

For example, use the following commands to divide the 100K library,

```
% cd /u0/lib/100k
% /usr/bin/makechipsfiles 100k.prt 100k.lib
```

The new files are stored in the subdirectory for each part. For example:

/u0/lib/100k/100171/chips_prt

is the individual chips file for the device 100171.

To use the SECTION and PINSWAP commands, installations must have the program:

/usr/bin/section/section

This is the program that figures out the section and pin assignments for the various parts. Installations must also include the following files:

/usr/bin/secassign
/usr/bin/makechipsfiles
/usr/bin/makedrawingnames
/usr/bin/maketextfile
/usr/bin/makewritefiles

SECTION 7
MIXING TEXT AND GRAPHICS

You can create mixed text and graphics documents interactively using the Graphics Editor's set of graphics tools. You can also add graphics to existing text. The need to physically cut and paste drawings into text is eliminated.

7.1 ADDING DRAWINGS TO EXISTING TEXT FILES

The FORMAT command is used to add drawings to an existing text file. By adding the drawing names to the existing text files and then formatting with GED, you eliminate the need to physically cut and paste illustrations into the document.

Documents made using the FORMAT command are .DOC drawings in the Graphics Editor. Editing a .DOC drawing is different from editing a regular schematic (.LOGIC, .TIME, .BODY). First, the grid is initially set up so that there are 6 grid spaces per inch on the final plot (the grid is set to 0.166). In addition, when a DOC drawing is written, only the ASCII and binary representations are saved. There is no need to create a connectivity representation because .DOC drawings are not read by the SCALD Compiler.

The UNIX text file should be created with a text editor such as vi using the troff macros. Refer to the appropriate UNIX reference documentation for more information. However, the default page length is too long for the font created by the Graphics Editor. To create pages with 59 lines, use the command **.pl -7** at the beginning of the file.

To add a GED drawing to a text file, you include 2 special formatting commands and then insert the specified number of blank lines at the location in the text where the drawing is to appear.

1. To specify the drawing, place an ampersand (&) in the first column and then type the name of the SCALD drawing (name.type.version.page).

2. In the first column of the next line, type the number of lines required for the drawing (6 lines = 1 inch).

3. Then insert the number of blank lines you specified. This allows GED to properly format each page.

   For example:

   & AN EXAMPLE.LOGIC.1.1 3
   .sp 3

4. Save your text file and then issue the FORMAT command:

   FORMAT *filename* <cr> *drawingname* <cr>

GED accesses the drawings specified in the UNIX text file, smashes them, and scales them to fit into their appropriate spaces. GED saves this new text and graphics document in the drawing name you specified on the command line.

The arguments to the FORMAT command are the name of the UNIX ASCII text file, a carriage return, and the name of the drawing the document is to be called. FORMAT then creates a SCALD .DOC document called *drawingname*.DOC. The UNIX file is a text file that has been formatted by the nroff text formatting program (under UNIX). Each page of the text file is turned into a page in a SCALD drawing. A page ends with the 60th line or a user-inserted formfeed (CONTROL-L).

Each page created by FORMAT is 8 1/2 by 11 inches, with 6 lines per inch. The characters are slightly larger than GED's default character font (1.29 times the default) for easier readability. The FORMAT command also adds tick marks on the corners of the document pages to facilitate cutting the Versatec output to the correct size.

7.2 CREATING DOCUMENTS INTERACTIVELY

Creating a document with GED requires being able to add both text and drawings. To add text lines, use either the NOTE command or create a file of text using a text editor and then add it to the GED document using the FILENOTE command. To add a figure, create the drawing with GED and add it to the document using the SCALE command.

For example, Use the following procedure to create Figure 7-1:

1. In UNIX, use vi to create a file called **mux.txt**.

2. Enter the text:

   The 2 to 1 MUX. If S is high, the output, /Y, is I1. If S is low, the output is I0.

3. Enter GED and create the MUX.BODY drawing.

4. Use GED to edit a drawing called EXAMPLE.DOC.

5. Type **FI**lenote mux.txt and point to the spot where the note should go.

6. Use the SCALE command to add the drawing. Type **SCALE MUX.BODY** and point to the opposite corners of the rectangle where the figure should go.

The SCALE command causes all bodies to be smashed into their primitive pieces. The BODY definition is not maintained.

The 2 to 1 MUX. If S is high, the output,
Y is I1. If S is low, the output is I0.

*

(SCALE)

*

becomes

The 2 to 1 MUX. If S is high, the output,
Y is I1. If S is low, the output is I0.

BUBBLE_GROUP=<I1:I0:Y>

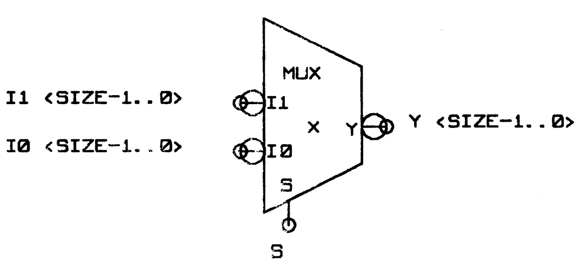
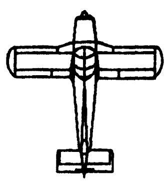


Figure 7-1: EXAMPLE.DOC

7.3 CHANGING AN EXISTING DOCUMENT

Once a document is created, either using the FORMAT command or interactively with the FILENOTE and SCALE commands, it can be edited using the Graphics Editor. You may want to, for instance, rescale figures or make simple changes to lines of text. Modifications can be made using regular Graphics Editor commands such as MOVE, COPY, CHANGE, WIRE, and GROUP.

For example, Figure 7-2, is an existing document. Several changes need to be made.



Aerotec 122 Uirapuru (Brazil)
Frst Flight 1965
Data: Standard t-23 military version
Cruising speed (Model A-122A) is 155mph
Powered by 108hp Lycoming Engine
Wing Span: 27 ft 10 3/4 in (8.5m)
Accommodation: Two seats side by side, with dual controls. Baggage compartment, capacity 66 lb (33 kg), aft of seats.

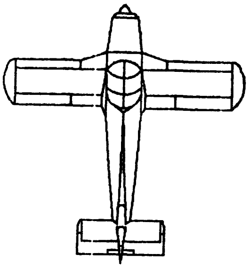
Figure 7-2: An Existing Document to Edit

1. The word ''Frst'' is misspelled in the second line. This can be corrected with the CHANGE command.

2. Use the WIRE command to underline the words Data, Wing Span, and Accommodation.

3. The scale of the drawing is too small. Define a group around the drawing and then delete the group. Use the SCALE command to add the drawing at a larger scale.

The results are in Figure 7-3.



Aerotec 122 Uirapuru (Brazil)
First Flight 1966
Data: Standard t-23 military version
Cruising speed (Model A-122A) is 155mph
Powered by 108hp Lycoming Engine
Wing Span: 27 ft 10 3/4 in (8.5m)
Accommodation: Two seats side by side, with dual controls. Baggage compartment, capacity 66 lb (33 kg), aft of seats.

Figure 7-3: The Existing Document Changed

7.4 PRINTING A DOCUMENT WITH GED

When a document is created using the FORMAT command, cut marks are placed at the corners of the page. These cut marks serve 2 purposes. First, they allow you to hardcopy at the default scale (HA 1) and get an 8 1/2 by 11 inch page. Second, the cut marks are used to cut the plotter paper to the correct size. Therefore, to ensure that the document is correctly plotted, do not delete the cut marks on the sides of the document or place text outside the cut marks.

If the marks are deleted or the page is created manually, type the following to create the cut marks:

```
DOT (410,4864) ;
DOT (410,-50) ;
DOT (4648,4864) ;
DOT (4648,-50) ;
```

Make sure that all the text and graphics lie within the box created by the dots to ensure that the page is plotted properly.

7.5 USING FONT STYLES

GED supports a variety of printing font styles which you can use to print your documents. See Appendix B for a complete list and character sets.

To specify a font, issue the SET FONT command with the required character set name and then use the HARDCOPY command to plot the document.

SECTION 8
DRAWING MAINTENANCE

GED provides facilities for you to recover from system failures and for updating drawings.

8.1 TEMPORARY FILES AND RECOVERY FROM CRASH

When you edit a second drawing without writing out the first one, the Graphics Editor saves a copy of the first drawing. The saved drawing is in a temporary binary file and is named a00aaaa?.xyz. The two digits (00) refer to the SCALD station window. The ? is a letter designating the number of the temporary file stored in the UNIX directory. Temporary files are not written into the SCALD directory.

These temporary files are deleted from UNIX if the Graphics Editor terminates normally. However, if the Graphics Editor exits abnormally, all drawings except for the last one can be restored to the version saved in the temporary files.

If GED or UNIX crashes, it is possible to recover the drawings that were being edited while GED was running. In the event of a crash, you can recover files by answering ''yes'' to the query about recovering files. Every time GED is called to the screen after a system crash, this query appears the first messages from the editor. For normal operations, it does not appear; in the event of a crash, it is used to recover files.

If you elect to recover drawings, they are all placed in a SCALD directory called restore.wrk. The recovered drawings are called RESTORED1, RESTORED2, ... If a restore.wrk SCALD directory already exists, it is overwritten. A warning message is printed about this, and it is possible to elect not to recover. To access your recovered drawings:

1. Type **USE RESTORE.WRK.**

2. EDIT the drawings in the reverse order. If there are drawings RESTORED1, 2, 3, 4, and 5, start editing RESTORED5 and work back to RESTORED1.

3. WRITE the edited drawings to the appropriate SCALD directory with the correct drawing names.

8.2 UPDATING OUT-OF-DATE DRAWINGS

If library parts change, it is often difficult and time consuming to look through a SCALD directory to determine if any drawings are affected. An update facility is provided with the Graphics Editor to make this process easier.

This update facility allows you to ask which drawings are out of date and then remake them, using the new parts. This is done from UNIX in a batch mode.

When a drawing is updated, several processes are performed. First, a list of all the parts used by a drawing is compiled. This list is then used to determine whether any of the parts in the library are newer than those in the drawing. Second, changed properties on parts are handled correctly. For instance, if a property on the part has been added or deleted, that property is added or deleted on the drawing. In addition, any modified part property values over-ride any default values.

DEPENDENCY FILES

The Dependency file lists the bodies used by a drawing and UNIX directory from which the parts came. The Graphics Editor writes a dependency file in addition to the ASCII, binary, and connectivity files for each drawing. When you run the update facility, the date on the Dependency file containing each part is compared to the date of the last write for the drawing. If any of the parts are newer than the drawing, the drawing needs to be updated.

UPDATING A DRAWING

To update the drawings in your directory, use the UNIX command

/u0/editor/update *argument*

The arguments are:

<cr> (No parameter) Find all drawings in the current directory that need updating and remake them. This option deletes the Binary file and then reads in the ASCII version of the drawing so that changes to properties are handled properly.

-n Find all the drawings in the current directory that need updating and produce a list of them.

-b Find all drawings in the current directory that need updating and remake them. It does not delete the binary versions first; consequently, if a binary version exists, the property changes are not handled correctly. This option is faster than the first (no parameter) option, and it is preferable if you know that only the body shapes have changed, not the properties.

-a ''*drawing name*''
This option remakes the named drawing whether or not it is out of date. The drawing name should be in quotes and fully specified; wildcards are not allowed. For example:

/u0/editor/update -a "SIZE SHIFTER.LOGIC.1.1"

-f ''*part name*''

This option finds and lists all the drawings that use the named part. Quotation marks surround the part name on the command line, and the name itself follows the same form as the ADD command. For example, to find all drawings that use the part 3 MERGE, type:

/u0/editor/update -f "3 MERGE"

If you type in a parameter that is different from the ones listed above, the screen displays a help message that lists the parameters for /u0/editor/update and the meaning of each.

The search stack used to find the components in the drawings is defined in the startup.ged file. The drawings updated are those in the current UNIX directory.

SECTION 9
COMMAND REFERENCE

This section provides an alphabetical reference to all the commands used by the Graphics Editor.

Commands can be selected from the menu, typed at the keyboard, or both. You can select the command name from the menu, type options at the keyboard, and then select a point with the cursor to execute the command.

The command syntax has the following sequence:

command name [operands...] [points...]

Lower-case words are replaced with appropriate commands, filenames, arguments, and options and by pressing the appropriate cursor control buttons.

- The command name is either selected from the menu with the cursor or typed on the keyboard.
- Operands, where they are appropriate, are specified from the keyboard.
- Points are selected with the mouse or can be entered from the keyboard as x,y coordinates.

These syntax conventions are used to describe the GED commands.

- Bold-faced type indicates the required portion of the command line. The bold text is the smallest unique portion of a command the program recognizes. GED also recognizes complete command names and both upper and lower case letters.

For example, the command DELETE is recognized if the two letters **de** (or **DE**) are entered. The rest of the command name (-LETE) can be entered or omitted.

- Italics indicate variables. These variables require the substitution of an actual filename or value.

- Optional fields are indicated by square brackets.

  For example,

  SECTION [*pin_number*] *pt*

  means that the argument for the pin_number can be used, but is not required. You do not type the square brackets.

- These abbreviations appear in the command syntax:

dwg	Drawing name
<*dir*>	SCALD directory name; angle brackets are required.
pt	Point. A point is specified by pressing the appropriate cursor control button. The x,y coordinates for a point can also be specified from the keyboard.
lib	Library
<cr>	Carriage return.
ver	Version

- The use of ellipsis (...) indicates that the preceeding fields on the command line can be repeated any number of times.

- If a sequence of items is enclosed in parentheses, (), followed by ellipsis, only the enclosed sequence can be repeated.

ADD

ADD Adds a specified body to a drawing.

Syntax ADD [<*dir*>] *body_name*[.[BODY][.[*version*]]] *pt*[*pt*...] ...

Examples

AD LS74	Adds version one of the part LS74 to the drawing.
ADD addr..2	Adds version 2 of the part ADDR to the drawing.

Description

The ADD command is used to add bodies to logic drawings. The *body_name* refers to the name of the body drawing to be added. The type can be specified, but is not required. The drawing type defaults to BODY since adding a logic drawing to another drawing is not permitted. The version defaults to 1, but any existing version of a body can be added.

Bodies refer to library components as well as .BODY drawings created for hierarchical designs. To add library parts to a drawing, specify the required library with the LIBRARY command. To add bodies for user-created design modules, specify the required SCALD directory with the USE command.

If no directory is specified (with the USE command), each SCALD directory in the current search stack is searched until the drawing with name *bodyname*.body is found. GED does not allow time, simulator, or SPICE parts in logic drawings. Similarly, sim parts cannot be added to time drawings, etc. The DIR <*> command tells whether any of the SCALD directories in a user's list are of the wrong type for the currently edited drawing.

After a body is added to a drawing, copies of the body can be made:

1. Press the left cursor control button.

   Another copy of the same body is attached to the cursor.

2. Move the cursor to the required place and press the button to position the copy.

ADD remains active until another command or the semi-colon is selected. Additional bodies can be added to the drawing without reselecting the ADD command. To add a new body, type the name of the body and press the ENTER key.

The ADD command can cause the following error message to appear.

Could not find device of name: *body_name*

This message indicates that GED could not find the specified body or that you tried to add a part from an illegal library. For example, you cannot add bodies from the TIME or SIM library to a LOGIC drawing.

- Check the spelling, to make sure that you typed the name of the body correctly.

- Make sure that the required component library is specified. Use the LIBRARY command.

See Also

REPLACE	Substitutes one body for another
VERSION	Selects an alternate version of a body, if available.

ASSIGN

ASSIGN Assigns a GED command or operation to a program function key.

Syntax ASSIGN *function_key "command text"*
ASSIGN *key_code "command text"*

Examples

ASSIGN (press F6) "DIS 2.0"

This example assigns the command, DISPLAY 2.0, to the F6 function key. This command key assignment remains active only during the current editing session.

ASSIGN ~@! "DIS 2.0"

This example assigns the command, DISPLAY 2.0, to the F2 function key. This version uses a code number to identify the function key. You add this ASSIGN command to your startup.ged file so that the softkey assignment is made automatically whenever you use GED.

Description

The ASSIGN command assigns a GED command to a function key allowing you to press the specified key instead of typing the command. This can save time when a command is used often or requires several variables and options on the command line.

Default function key assignments for commonly used GED commands are supplied with the Graphics Editor. These values are stored in the file, /u0/editor/softkeyassign. The current assignments can be displayed with the SHOW KEYS command. The softkey assignments can be tailored for the entire system by editing the values in /u0/editor/softkeyassign.

The recommended method for changing the function key assignments is to place ASSIGN commands in individual startup.ged files. These values take effect whenever you use GED. Alternatively, you can issue the ASSIGN command during an editing session to make softkey assignments that have effect only during that session.

1. Type: **AS**SIGN

2. Press the required function key.

3. Type the GED command to be assigned.

   The command text is enclosed by quotation marks and can contain a maximum of 60 characters, including spaces.

4. Press the ENTER key.

You can also define softkeys by putting ASSIGN statements in your startup.ged file. You identify the function key by typing the code for the key. The command text, up to 60 characters, is enclosed in quotation marks. A RETURN is automatically appended to the end of the assigned string. The values for function keys are given in Table 9-1.

Table 9-1. Key Code Values

KEY	CODE
F1	~@ (space)
F2	~@ !
F3	~@ ”
F4	~@ #
F5	~@ $
F6	~@ %
F7	~@ &
F8	~@ ’
F9	~@ (
F10	~@)

AUTO

AUTO	Performs the global addition of certain objects to a drawing. The DOTS option automatically inserts a dot at each wire junction. The PATH option automatically assigns the path property where required.
Syntax	**AUTO DOTS** **AUTO PATH**

Description

The AUTO command automatically adds dots or path properties to your drawing.

The DOT option places a dot at each connection point in the current drawing. Open dots are the default value. Before using the AUTO DOT command, you can issue the SET DOTS_FILLED command to specify that filled dots be displayed.

The PATH option of the AUTO command is used to make bodies with the same name unique by assigning the PATH property. AUTO PATH assigns a unique path number (PATH = *n*P) to each body without a path property.

Path properties are automatically assigned when a drawing is written. The AUTO PATH command allows you to assign path properties before you write the drawing.

See Also

SET	Allows you to specify the style of dots to be displayed (open or filled).
Section 3	Contains additional information about the path property.

BACKANNOTATE

BACKANNOTATE Annotates designs with physical information from the Packager.

Syntax BACKANNOTATE

Description

GED reads a schematics annotation file produced by the Packager and includes physical information such as location designators, pin numbers, and physical net names on the design.

The annotated properties added by GED are soft properties. Soft property names begin with a dollar sign ($) and are not written into the connectivity file. This allows the Packager to reassign the physical information each time the design is repackaged.

You can move and delete soft properties, but you cannot use GED to change existing soft properties. You can, however, change a soft property into a hard property, by using the PROPERTY command and adding a property with the same property name, minus the dollar sign.

For example, if a component has a $LOCATION property, add a LOCATION property.

To generate a back annotation file for GED, perform the following steps.

1. Run the Packager with the following directive:

   output backannotation;

   There are options for backannotating location designators, pin numbers, and physical net names. See the Packager Reference Manual.

2. Rename pstback.dat to backann.cmd. The file pstback.dat is generated by the Packager. The backann.cmd file must be in the current UNIX directory.

3. Enter GED and type the BACKANNOTATE command.

GED reads the file, edits each named drawing in turn, adds the appropriate physical information, and writes the drawing.

See Also

PROPERTY	Adds a property to a design.
Section 4	Contains additional information about properties.
Section 6	Contains information about backannotation and adding physical information to a design.

BUBBLE

BUBBLE Toggles the state of a pin between bubbled and unbubbled.

Syntax BUBBLE *pt..*

The BUBBLE command toggles the state of a pin between ''bubbled'' and ''unbubbled'' if the body is defined to permit this conversion. If the pins are established as part of a BUBBLE GROUP, the BUBBLE command can be used to convert the body from one form to another.

The Compiler Bubble Check feature can verify that the bubble states of connected pins are matched. When Bubble Checking is on, connections with mismatched bubble states are reported as errors.

CHANGE

CHANGE Allows you to use a line editor or screen editor to modify selected lines of text.

Syntax CHANGE *pt*..

Description

The CHANGE command allows you to use a line editor or screen editor to modify selected lines of text such as notes and signal names in a design.

The text strings are chosen by pointing with the cursor and pressing the left button. You can select as many strings as necessary. If you use a screen editor, the text is placed in a file. If you use the line editor, the selected text strings are displayed, one at a time, on the bottom left of the screen.

To access the VI screen editor, type:

CHANGE *pt*.. Control-V.

All the text strings you selected are placed in a file, one string to a line. Use the vi editing functions to move around the file and make required changes. You cannot change the number of lines in the file while you are editing it.

When you are finished, type ZZ (SHIFT-zz) to exit from the file. The changed text is repositioned on the drawing. Refer to the appropriate manual for more information about the vi screen editor.

Using the GED Line Editor

The line editor uses a vertical-line cursor. Table 9-2 contains the key combinations and the resulting operations you can perform in the line editor.

Table 9-2. Line Editor Functions

Keys	Result
Control-F	Moves the cursor forward one character.
Control-B	Moves the cursor back one character.
Control-E	Moves the cursor to the end of the line.
Control-A	Moves the cursor to the beginning of the line.
Control-D	Deletes one character to the right of the cursor.
Control-K	Deletes the remainder of the line (right of the cursor).
Control-I	Displays the HELP file for the line editor.
Control-X	Exits from the line editor and repositions the text.
Control-S *character* <cr>	Searches to the right of the cursor for the specified character.
Control-R *character* <cr>	Searches to the left of the cursor for the specified character.
Control-U *number command*	Repeats the command the specified number of times. If no number is given, the default is four.
Control-Z	Aborts changes to the text currently on the edit line and repositions the original back onto the drawing.

To insert text to the right of the cursor, type the characters to be inserted.

To select a new line of text, point to the text string and press the left button.

CHECK

CHECK	Checks for connectivity problems and general errors on the current drawing.
Syntax	CHECK

Description

The CHECK command adds path properties (P-numbers) and examines a drawing for connectivity problems and other general errors. These problems are difficult to detect by looking at the drawing and cause compilation errors. These errors include:

- Wires that are not connected, but appear to be connected
- Pins attached to more than two wire segments
- Duplicate components in the same location
- Wires connected to only one pin and not named (NC wires)
- Nets that are named but not connected to any pins

CHECK lists each detected error and its location. After you run the CHECK command, you can use the ERROR command to locate each error on the drawing.

See Also

ERROR	Locates and displays each error detected by CHECK.

CIRCLE

CIRCLE Adds circles and arcs to a drawing.

Syntax CIRCLE *center_pt radius_pt* [*arc_pt*]...

Description

The CIRCLE command is used to create both circles and arcs. Although circles and arcs are rarely necessary on logic designs, they are commonly used for creating body drawings.

To place a circle on the drawing:

1. Type **CIRCLE**.

2. Select a point as the center of the circle.

3. Select a second point to determine the length of the radius. The circle appears.

An arc is defined by three points: the center, a point marking the termination of the radius, and a third point along the circumference of a circle.

To draw an arc:

1. Type the CIRCLE command followed by a center point.

2. Select a second point to determine the length of the radius and the starting point of the arc. The completed circle appears as soon as the radius point is specified.

3. Move the cursor counterclockwise from the radius point along the circumference of the circle and specify a point to determine the ending point for the length of the arc.

COPY

COPY	Copies objects, properties, and groups in the current drawing.
Syntax	COPY *source-pt destination-pt...* COPY *property-pt destination-pt attach-pt...* COPY *group_name destination-pt...*

Definition

The COPY command is used to copy objects, properties, and groups of objects on the same drawing. The first point identifies the object to be copied; the second point identifies the position of the new copy. When you copy a property, a third point attaches the property to an object (body, pin, or wire).

To copy an object such as a body or a wire:

1. Select or type the COPY command.

2. Position the cursor on the object and press the appropriate button.

   The left button picks up a copy of the object at the grid point nearest the cursor.

   The right-hand button picks up a copy of the object at the vertex nearest the cursor. (The vertex of the copy snaps to the cursor.) This operation is useful for copying component bodies.

3. Move the copy to its location and press the appropriate button.

   The left button places the copy on the grid point nearest the cursor.

   The right-hand button attaches the copy to the nearest vertex. This is useful for attaching copies of wires at new locations.

To copy a group:

1. Use the GROUP command to define a group.

2. Select or type the COPY command.

3. Move the cursor to the group to be copied, and press the center button.

   or

   Type the single-letter *group_name* and press the ENTER key.

4. Move the cursor to the location for the copy and press the left button.

To copy properties:

1. Select or type the COPY command.

2. Move the cursor to the property to be copied and press the left button.

3. Move the cursor to the location for the copy and press the left button.

   A rubber band line is drawn from the property to the cursor.

4. Move the cursor to the object where the property is to be attached and press the left button.

You can attach the property to a part, wire, pin, or signal name.

You cannot copy default body properties, pin properties, or those properties generated by the SECTION, PINSWAP, and BACKANNOTATE commands.

Default properties and user-added body properties are included in copies made of parts. Wire properties are not included when you copy a wire.

See Also

CUT and PASTE	These commands allow you to copy objects or groups from one drawing to another.
GROUP	Defines a group of objects, which can be moved.

CUT

CUT Copies an object or a group from the drawing to a buffer.

Syntax CUT *pt*
CUT *group_name*

Description

The CUT command, along with the PASTE command, copies objects and groups from one drawing to another. Use the CUT command to place the specified object or group into a ''cutting'' buffer. The cutting buffer can contain one group or object.

1. Type CUT.

2. Select the object to be cut by pointing with the cursor and pressing the left button.

   or

   Select the group to be cut by typing the group_name or pointing with the cursor and pressing the center button.

The CUT command highlights the selected object or group and also displays the number of bodies, wires, dots, circles, and notes that have been put into the buffer.

Default body properties and user-added body properties are included in copies made of parts. Properties that are not copied with the body include PATH, properties generated by the PINSWAP, SECTION, and BACKANNOTATE commands, and pin properties. Wire properties are copied when a wire is cut. This allows signal names to be transferred to the new drawing.

See Also

PASTE	Transfers objects from the cut buffer to specified locations in the current drawing.

DELETE

DELETE	Removes objects from a drawing.
Syntax	DELETE *pt* ... DELETE *groupname* ...

Description

The DELETE command is used to remove objects from a drawing.

- To delete a part, issue the command, point to the part, and press the left button.
- To delete a wire or an arc, issue the command, point to any point on the line or arc, and press the left button.
- To delete a text string, issue the command, point to the vertex of the string, and press the left button. The vertex of a left-justified string is on the lower left corner; the vertex of a right-justified string is on the lower right corner.
- To delete a group, issue the command, point to the group, and press the center cursor control button. The group nearest the cursor is deleted.

  or

  Issue the command and type the single-letter name of the group.

Default properties on bodies and pin number properties (PN) generated by PINSWAP cannot be deleted by the user.

See Also

UNDO	If a group or object is deleted by mistake, use the UNDO command to retrieve it.
GROUP	Defines a group of objects, which can be deleted.

DIAGRAM

DIAGRAM	Changes the name of the current drawing
Syntax	DIAGRAM [<*dir*>] *dwg*[.[*type*][.[*version*][.[*page*]]]]
Default Values	<current> *drawing*.logic.1.1.1

Description

The DIAGRAM command is used to change the name of the current drawing. This allows you to use an existing drawing as a pattern for a new drawing or to save a copy of a drawing under a different name before making changes to it.

To rename a drawing:

1. Edit the drawing to be changed.
2. Type the DIAGRAM command followed by the new name of the drawing.
3. Type the WRITE command to save a copy of the drawing under its new name.

See Also

USE	Specifies a working directory on the active search list.
DIRECTORY	Lists the drawings in the SCALD directory.
IGNORE	Excludes the specified directory or library from the active search list.

DIRECTORY

DIRECTORY Lists the contents of SCALD directories.

Syntax **DIRECTORY**
[<*dir*>] [*dwg*] [.[*type*] [.[*version*] [.[*page*]]]]

Examples

DIR	Lists all drawing names in the current directory.
DIR *	Same as DIR.
DIR <*>	Lists all active directories (but no drawing names).
DIR <time>*	Lists all drawing names in the TIME directory.
DIR ls*	Lists all drawing names beginning with LS in the current directory.
DIR *.body*	Lists all bodies in the current directory.
DIR <*>*	Lists all drawing names in all active directories.
DIR *.*.*	Lists the name, type, and version of each drawing in the current directory.

Description

The DIRECTORY command lists the names and contents of the SCALD directories in the current directory list. There is no limit to the number of SCALD directories you can use at one time. The DIRECTORY command displays the contents in the order the directories are searched, with the current working directory displayed first.

You can use wild card characters in directory names and drawing names. An asterisk (*) matches anything, and a question mark (?) matches any single character.

Unless you specify type, version, or page parameters, the DIRECTORY command displays just the drawing name.

See Also

IGNORE	Excludes the specified directory or library from the active search list.
USE	Specifies the current working directory on the active search list.
LIBRARY	Specifies the component library to be accessed.
Section 2	Describes SCALD directories and their operation.

DISPLAY

DISPLAY	Changes the way objects are displayed on a drawing.
Syntax	DISPLAY *option* [*"groupname"*] DISPLAY *option* pt

Examples

DIS INVISIBLE ''A''	Makes all properties in group A invisible. The quotation marks are required.
DIS BOTH *pt*	Displays the name and the value for the selected property.
DIS 2 *pt*	Enlarges the selected text by two times.
DIS .5 *pt*	Makes the selected text half as large.
DIS FILLED *pt*	Makes the selected dot solid.

Description

The DISPLAY command changes the way objects are displayed on a drawing. DISPLAY can be used with the cursor control buttons to specify either a single item or a group. DISPLAY can also be used with a group name. The group name specified must be quoted. The group can contain any type of object. The DISPLAY command selects the correct objects to change. Several options can be specified with the DISPLAY command.

The options NAME, VALUE, BOTH, and INVISIBLE deal with the way property values are displayed on the drawing. Although properties consist of name and value pairs, only the value is displayed when a property is added to a drawing. These options allow you to display the name alone, the value alone, both, or neither.

NAME	Displays only the name of the property.
VALUE	Displays only the value of the property.
BOTH	Displays both the name and the value of the property.
INVISIBLE	Displays neither the name nor the value of the property.

To change the display, type the command DISPLAY, the required option, and then select one or more properties with the cursor. After the form of a property has been changed, that change remains in effect until another DISPLAY command is used to change it again.

For example, you can define a body with a default property SIZE = 1B and suppress the display of the property with the DISPLAY I (INVISIBLE) command. When that body is added to a logic drawing, the property SIZE = 1B does not appear. Use the DISPLAY V (VALUE) option and point to the location of the property to make it appear on the drawing. The SHOW PROPERTIES command displays the name and value of all properties (including invisible ones) on the drawing.

These options determine the size of text displayed on the drawing.

DEFAULT	**Displays text on the drawing at the default size, 12 characters per inch.**
real_number	**Enlarges or reduces the size of the text on the drawing by the amount specified by real_number.**

When a text string is added to a drawing, it is defined by a vertex at the lower left corner of the text string. Text is added to a drawing at 12 characters per inch. This size of text is legible on a hard copy of the drawing without taking up more space than necessary.

To change the size of a string of text, type the command **DISPLAY** *real_number* to indicate the factor by which the size of the currently displayed text is to be multiplied.

Then, using the cursor, select the string of text to be changed.

To return the text to the default scale, use the DEFAULT option with the DISPLAY command and point with the cursor to specify the text.

The next three options change the way an existing wire looks.

HEAVY	Makes the wire thicker so it looks like a bus.
THIN	Returns a heavy line back to the default wire thickness.
PATTERN *number*	Changes a wire to one of six patterned lines. Pattern 1 is a filled line (the default); patterns 2-6 are a variety of dotted and dashed lines.

In a LOGIC drawing, the entire wire changes. In a BODY or DOC drawing, only the wire segment specified by the cursor is changed.

The next two options provide for the change of the display of dots already added to the design.

FILLED	Displays solid dots.
OPEN	Displays open dots (default).

Open dots scale when the WINDOW command is used; filled dots do not.

The last two options alter the justification of text on the screen. By default, all user-added text is left justified.

LEFT	Left justifies selected text strings.
RIGHT	Right justifies selected text strings.

When a right-justified string is moved, the cursor attaches to the right end of the string.

See Also

FIND	Allows you to define a group of information to be affected by the DISPLAY command.
GROUP	Allows you to define a group of objects to be modified by the DISPLAY command.
SET	Allows you to change the default options to be used by GED. The SIZE, DOTS_FILLED, DOTS_OPEN, LEFT_JUSTIFICATION, and RIGHT_JUSTIFICATION options of the SET command affect some of the same drawing elements as the DISPLAY command.
SHOW	Temporarily displays drawing information. Several SHOW command options affect the same drawing elements as the DISPLAY command. The SHOW command is useful for viewing the current values on the drawing before making changes with DISPLAY.

DOT

DOT Adds dots to drawings to indicate connection points.

Syntax **DOT** *pt...*

Description

The DOT command is used to add dots to drawings. Dots are used in logic drawings to indicate that wires crossing one another are connected. (By default, wires crossing are not connected unless dotted. Wires joining at a ''tee'' are connected, even without a dot.) Dots are used in body drawings to indicate pin connection points. Dots can be filled or open. By default, all added dots are open.

See Also

SET The DOTS_FILLED and DOTS_OPEN options change the default dot type.

DISPLAY The FILLED and OPEN options change the style of dots displayed on the drawing.

AUTO DOTS Places a dot at all the connection points in a logic drawing.

SHOW CONNECTIONS
Temporarily displays all the connection points in a logic drawing.

EDIT

EDIT Displays an existing drawing to be edited or allows you to create a new drawing.

Syntax EDIT [< *dir*>] *dwg*[.[*type*] [.[*ver*] [.[*page*]]]]
EDIT *pt*

Examples

ED test — Displays drawing, Test.logic.1.1

ED size shifter.time — Displays the time drawing, Size Shifter.time.1.1

ED circuit...2 — Displays the second page of the drawing, Circuit, with the type and version of the drawing being edited.

ED ...2 — Displays the second page of the current drawing (named on the status line).

Description

EDIT is the basic command for calling a drawing onto the screen. If no SCALD directory is given, each directory in the list is searched until a drawing of that name is found. If the specified drawing is found, it is displayed on the screen. If it is not found, the system creates a drawing by that name in the current SCALD directory.

If the drawing type is not specified, GED assumes that the drawing is a logic drawing. To edit another type of drawing, include the drawing type in the command.

The default value for both version and page is 1. Page specifications for body drawings are ignored, but each body can have multiple versions. Other drawings, such as logic, time, and sim, can have multiple versions and pages.

You can use the EDIT command to edit a second drawing without writing the current drawing. EDIT saves the first drawing, along with any changes, in a temporary file before bringing in the new drawing. If you re-edit the first drawing, EDIT displays the modified version from temporary storage.

The EDIT command also allows you to examine the logic definition of a component. This is used in hierarchical designs, not for library components. For example, to edit the logic associated with a body (for example, SUBTRACTOR) in the current drawing, type the EDIT command, point to the body with the cursor, and then press the left button. The current drawing is placed in temporary storage, and drawing, SUBTRACTOR.LOGIC, is displayed and can be edited.

See Also

SHOW HISTORY	Lists the drawings that have been edited during the current GED session.
GET	Replaces the current drawing with the version stored on the disk.
RETURN	Returns to the previously edited drawing.

ERROR

ERROR Locates and displays each error detected by the CHECK command.

Syntax ERROR

Description

The ERROR command steps through the errors found by CHECK. It centers each error on the screen, draws an asterisk at the location of the error, and displays a message describing the error.

After you correct an error, proceed to the next error by retyping the ERROR command or selecting it from the last box on the menu.

EXIT

EXIT	Allows you to leave the editor. Same as QUIT.
Syntax	EXIT

Description

The EXIT command allows you to leave the Graphics Editor. After you issue the EXIT command, GED displays a message if there are unwritten changes to the drawings in the current editing session. If you issue the EXIT command again, any changes to drawings are lost.

See Also

QUIT	Allows you to end the editing session. Same as EXIT.
WRITE	Writes the current drawing to the disk.

FILENOTE

FILENOTE Includes a named text file in a drawing at a specified point.

Syntax FILENOTE *filename pt*

Description

The FILENOTE command adds the named text file to a drawing at the point specified. When the file is added, each line in the file is converted into a note and can be individually moved, copied, deleted, and changed.

FIND

FIND	Searches the current drawing and places all data that matches a specified pattern into a group.
Syntax	**FIND** *pattern*

Examples

FIN *PATH	Locates all path properties.
FIN LS00	Locates all LS00 components on the drawing.

Description

The FIND command searches through the current drawing for all notes, property names, property values, and body names that match a given pattern.

You can use wild card characters in the pattern. An asterisk (*) matches any number of characters, and a question mark (?) matches any single character. The FIND command is not case-sensitive; it does not distinguish between upper and lower-case alphabetic characters in the pattern.

The command classifies all matching items as a group. This group has a one-letter name (A-Z). The number of items in the group is displayed on the screen. GED operations such as SHOW, DELETE, and DISPLAY can also be performed on the entire group.

All items found with the command are placed on a list. You can step though the list items using the NEXT command. This command centers each item on the display where it can be changed or deleted.

See Also

NEXT	Centers each item located by FIND on the screen.

FORMAT

FORMAT Combines a text file with referenced drawings into a new drawing.

Syntax FORMAT *text_file* <cr> *new_dwg*

Example

FORMAT design.dat <cr> specification

Processes the text file design.dat and creates a GED drawing called specification.doc

Description

The FORMAT command creates a DOC drawing file by merging specified drawings with a processed ASCII text file. To use the FORMAT command:

1. Type FORMAT and the name of the text file.
2. Press the ENTER key.
3. Type the name of the new DOC drawing.

The text file must contain references to the drawings to be included in the final document. The drawing name, preceded by an ampersand (&), and the number of lines required by the drawing are placed in the text file, marking the location of each drawing. Before you issue the FORMAT command, the text file must be processed by a text processor, such as TROFF or RUNOFF.

The Graphics Editor reads the named drawing, SMASHes it, and then scales it to fit into the allotted space. Each page of the text file is turned into a page of a drawing DOC file. The pages created by FORMAT are 8-1/2 by 11 inches, with 6 lines per inch. A page ends automatically at line 60 or a user-specified formfeed (Control-L). For easier readability, the characters are 1.29 times larger than the default character size.

See Also

Section 7	Contains more information about creating mixed text and graphics documents.
SMASH	Breaks a body into the separate objects that define it.
FILENOTE	Allows you to add a text file to a GED drawing.

GET

GET Replaces the current copy of a drawing with the version stored on the disk.

Syntax GET [<*dir*>] *dwg*[.[*type*]].[*version*][.[*page*]]]]
GET <cr>

Description

The GET command retrieves and displays the copy of the drawing stored on disk. This fresh copy of the drawing replaces any previously read and modified version inside the Graphics Editor. GET is useful if, while editing a drawing, you want to discard current work and go back to the previous version.

If no SCALD directory is given, each directory in the list is searched until the specified drawing is found. If the drawing is not found, the new drawing is assumed to belong to the current SCALD directory. GET followed by a carriage return re-reads the current drawing.

GRID

GRID Alters the way the grid is displayed.

Syntax **GRID** *option*;

Description

The GRID command is used to specify the way the grid is displayed. The current values of the grid multiple are displayed on the status line at the top of the screen. These options perform the following functions:

GRID ON; Displays the grid lines on the screen.

GRID OFF; Turns off displayed grid lines.

GRID <cr> Toggles the grid lines on and off.

GRID *grid_size*;

Defines, in inches, the separation of the grid lines. The default size for editing logic, time, and sim drawings is 0.1 or 1/10 of an inch. The grid_size (a real number) must be a multiple of 0.002 inches, which is the smallest possible grid separation.

Be extremely careful when changing the grid size. Bodies could be placed off grid and then, if the grid size is again changed, wires would not be connected.

Default Grid Sizes				
Body	Logic	Time	Size	Doc
0.05	0.1	0.1	0.1	0.166

GRID *grid_size grid_multiple*;

Specifies the grid size and multiple to be displayed.
The multiple indicates how many lines of the grid are skipped before the next line is displayed. The default value for Logic drawings is 5. You can specify a positive integer to change the default grid multiple. Specify 1 to display every line; 2 to display every other line, etc.

GRID DOTS; Displays the grid as dotted lines.

GRID LINES; Displays the grid as solid lines.

The GRID command must be terminated by a semicolon (;) or by any command from the menu. This does not include GRID <cr>, which toggles the grid on or off.

GED uses 500 internal units per physical inch. The grid multiple displayed on the status line of the display is in grids per inch.

If you use the SET METRIC command to base plots on the metric system, the Graphics Editor then uses 512 internal units per physical inch or 20 internal units per physical millimeter. The grid multiple displayed on the status line is expressed in grids per millimeter. Metric users can use standard Valid libraries since pins are on 2.5 mm centers.

With 500 internal units per inch, you cannot use a 1/8 inch grid (the grid can be set to .124 or .126 but not .125). If you use the SET FRACTIONAL command, GED resets the internal units to 400 per inch. (This allows the Valid library components to remain compatible with the drawing.) In this case, the bodies appear to be 25% larger, and the pins are placed on 1/8 inch centers.

See Also

SET	Changes the default values used by GED.

GROUP

GROUP	Combines selected objects into a group.
Syntax	**GROUP** [*group_name*] *pt pt pt...*

Description

The GROUP command creates a group of objects upon which you can perform operations.

To define a group:

1. Issue the GROUP command.

2. OPTIONAL: Specify the name of the group. A default, single-letter name is assigned if you do not enter one.

3. Use the cursor controller to draw a polygon around the required objects. Press the left button to change the direction of the line.

4. Close the polygon by pressing the right-hand button when the cursor is near the starting point.

   All of the vertices within the polygon are included in the group just defined.

The screen displays the group name and the number of bodies, arcs, properties, notes, dots, and wires that are in the group.

You can perform these operations on the entire group:

- COPY
- CUT
- DELETE
- FIND
- PASTE
- MOVE
- REPLACE
- VERSION

See Also

FIND	Allows you to define a group of objects.

HARDCOPY

HARDCOPY Sends drawings to a plotter to produce a printed output.

Syntax HARDCOPY [*scale_option* [<*dir*>] *dwg*[.[*type*] [.[*ver*] [.[*pg*]]]]]

Examples

HA	Plots the current drawing at the default scale of 1.
HA A	Scales the current drawing onto an A size page.
HA C *.logic*	Scales all LOGIC drawings in the current directory onto C size pages.
HA 1 <100k>100112.body*	Plots all versions of the 100112 part from the 100k library.
HA 1 hyper mux	Plots all drawing types for the drawing Hyper Mux (BODY, LOGIC, SIM) in the current directory.

Description

The HARDCOPY command is used to send drawings to a plotter to produce a hardcopy. Plots can be made on dot matrix, electrostatic, or pen plotters, including Epson, Versatec, Hewlett Packard, and Benson models. The SET command specifies what type of plotter is to be used.

HARDCOPY scale options:

A -- E	You can enter a page size specification to scale the drawing.
scale_factor	You can specify a number to scale the drawing from the normal size.

If a drawing name is given, a scale factor (number or pagesize A-E) MUST be given.

You can use wild card characters to specify drawings to be plotted. An asterisk (*) matches anything. This allows you to print several drawings with a single HARDCOPY command.

See Also

Section 5	Contains information about plotting drawings with your SCALD system.
SET	Allows you to specify the type of plotter to be used with the HARDCOPY command.

HELP

HELP Displays the on-line documentation for a specified GED command.

Syntax HELP *command_name*

Description

This command displays the contents of a specified help file. The help file briefly describes the syntax and the semantics of the selected command.

The command **HELP help** or **HELP <cr>** displays the list of topics on which help is available.

You can use the WINDOW command to enlarge the size of the text, if that is necessary. To exit from HELP, select another command from the menu.

IGNORE

IGNORE	Causes a specified directory or library to be deleted from the active list.
Syntax	IGNORE *directory_name* IGNORE *library_name*

Examples

IG <cr>	Ignores the current SCALD directory.
IG lsttl <cr>	Ignores the lsttl library.
IG practice.wrk	Removes the SCALD directory, practice.wrk, from the active search list.

Description

When you issue the IGNORE command, the system prompts you to answer YES or NO before proceding with the operation. When you type YES, this command causes the specified SCALD directory or library to be deleted from the active search list. The argument specified can have wild cards. If more than one directory matches the pattern, each one is ignored.

When you ignore a directory or a library, any bodies used in the drawing from the ignored directory or library are deleted from the screen. The body name is displayed to remind you to replace the body. The other active SCALD directories and libraries are searched for bodies with the same name and version. If one is found, the missing body is automatically replaced by the body from the other directory. If another body is not found, issue the USE or LIBRARY command to specify a directory of library with an equivalent part. This is used for using various versions or user-defined bodies in a hierarchical design.

See Also

USE	Specifies the working directory on the active search list.
LIBRARY	Adds a library to the active search list.
Section 2	Explains the file and directory structures in GED and includes a discussion of the active search list.

LIBRARY

LIBRARY	Adds a specified library to a search list.
Syntax	**LI**BRARY [*library_name*]

Examples

LIB sttl	Adds the sttl library of parts to your active search list of directories.
LIB <cr>	Lists all possible libraries that you can reference with the LIBRARY command.

Description

The LIBRARY command adds a specified library to the active search list. This allows you to reference parts from the library and add them to your design. To list the available library names, type: **LIBRARY <cr>**.

MIRROR

MIRROR Creates a mirrored version of a selected body.

Syntax MIRROR *pt*..
MIRROR <cr>

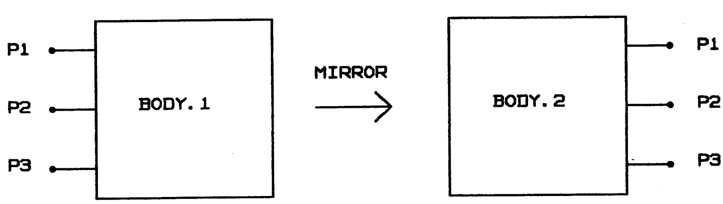
Description

The MIRROR command creates a mirrored version of a body, as opposed to a rotated version. This command mirrors, about the Y axis, all lines, arcs, and text in a body drawing. Justified text is shifted from left to right or right to left in the mirrored version. No other rotation is done.

To create a mirrored version of a body included in a logic drawing, issue the MIRROR command and then press the left cursor button.

In a body drawing, the MIRROR command does not require a point; the entire body definition is flipped over. This procedure is useful for creating other versions of a body.

For instance, two versions might resemble the following drawings:



The MIRROR command should be used with caution, especially with bodies with unmarked pins, such as the Valid-supplied merge bodies. Reversing the bits causes subtle, hard-to-find errors in the design.

See Also

ROTATE	Rotates a body or text string 90 degrees with mirrors at 180 degrees and 270 degrees.
SPIN	Provides true rotations, not mirrors, of a body.
VERSION	Displays alternate representations of a body.
Section 3	Contains additional information about creating body drawings.
Section 4	Contains additional information about adding bodies to a drawing.

MOVE

MOVE Moves objects from one position to another.

Syntax MOVE *group_name destination_pt..*
MOVE *source_pt destination_pt..*

Description

The MOVE command is used to move objects from one position to another on the drawing. The first argument identifies the object or group to be moved; the second point identifies its new position.

Properties (including signal names) are moved with the objects to which they are attached. Also, properties can be moved independently of objects.

If you move an object or group of objects that has electrical connections (wires), the MOVE command preserves the electrical connectivity and keeps the wires orthogonal (drags).

To use the MOVE command to move single objects:

1. Select or type the MOVE command.

2. Position the cursor on the object that is to be moved and press the appropriate button.

   The left button picks up the object at the grid point nearest the cursor.

   The right button picks up the object at the vertex neares the cursor. (The vertex of the object snaps to the cursor.) This operation is useful for moving bodies.

3. Move the object to its new location and press the appropriate cursor button.

   The left button places the object on the grid point nearest the cursor.

   The right-hand button attaches the object to the nearest vertex.

The MOVE command also operates on groups. If the center button is used to pick up a group, the nearest group is selected. If the name of the group is typed, followed by a carriage return, the group is snapped to the current cursor position.

NEXT

NEXT Displays the items located by FIND.

Syntax NEXT

Description

The NEXT command steps through the items found by the FIND command. It centers each item on the screen and draws an asterisk at the location of its vertex. You can perform an operation on the object and then issue the NEXT command to proceed to the next item. You can only step through a list once.

See Also

FIND Defines a group of items that match a specified pattern.

NOTE

NOTE Adds text strings to a drawing.

Syntax NOTE (*text_line... pt...*)... ;

Description

The NOTE command allows you to add notes to document your drawings. Notes are text strings that appear on the drawing; they do not affect the evaluation of the drawing by the SCALDsystem.

To add notes to a drawing you can:

- Specify the points on the drawing where the notes are to be located and then type in the lines of text. Press the ENTER key after each note to position each line of text on the drawing. As long as there are points remaining, GED interprets the text you enter as notes to the drawing.

  or

- Type in each line of text and press the ENTER key. Then use the cursor and press the left button to indicate where each note is to appear on the drawing.

Unless there are remaining points or text, GED recognizes command words you enter from the keyboard.

Notes beginning with an open parenthesis must be quoted. Also, any note which is enclosed in double quotation marks is not evaluated as a potential GED command.

See Also

FILENOTE	Allows you to add a text file to a GED drawing.

PAINT

PAINT Assigns selected colors to specified objects.

Syntax PAINT *colorname* ''*groupname*''
PAINT *colorname pt*

Description

The PAINT command allows you to assign colors to the objects in your drawing. You can use up to 16 colors in a drawing:

Red	Orange	Salmon	Aqua
Green	Purple	Violet	Peach
Blue	Gray	Skyblue	Brown
Yellow	White	Pink	Mono

When you issue the PAINT command, the on-screen menu of commands is replaced by a list of the available colors. To assign a color to an object, use the cursor to select a color from the menu and then point to the required object and press the cursor button again. You can also define a group and then assign a color to the group. For example, to change the color of the LS00 bodies in the design:

1. Type: **FIND ls00**

2. Type: **PAINT skyblue "a"**

A group can be selected with a quoted group name or by pointing to the required group and pressing the center button.

If you have a color monitor, the colors are displayed on the screen.

You can establish default colors for the objects in your drawings with the SET COLOR commands. You can issue these commands during an editing session or place SET command statements in your startup.ged file.

See Also

SHOW	The SHOW COLOR command lists the color of the specified object.
SET	The SET COLOR commands allow you to specify a default color to be used for each type of object in the drawing.

PASTE

PASTE	Copies the contents of a CUT buffer to the current drawing.
Syntax	PASTE *pt ...*

Description

The PASTE command, in conjunction with the CUT command, allows objects to be copied from one drawing to another. To copy a group or object that has been CUT, type PASTE, press the ENTER key, and then select the point to position the group or object.

To add more copies of the cut buffer, press the left cursor button, position the copy, and then press the left button again.

See Also

CUT	Copies an object or a group from a drawing to a buffer.

PINSWAP

PINSWAP	Swaps the pin numbers defined to be in the same pin group on a body.
Syntax	PINSWAP (*pt pt*)... PINSWAP *pinnumber pt*..

Description

The PINSWAP command swaps the pin numbers belonging to the same pin group on a body. This command can only be used after section assignment has occurred for the part. Also, pin swapping can only occur between pins that have been defined in the library as swappable. For example, it may be legal to swap the two input pins of a NAND gate, but not the input and output pins of the gate.

To swap pins:

1. Type PINSWAP.

2. Point to the two pins to be swapped,

   or

   Type in a new pin number and then point to the selected pin. The selected pin is swapped with the pin having the pin number you specified.

The properties attached by the PINSWAP command cannot be changed, only deleted and moved. Once pins on a part have been swapped, the part cannot be resectioned using the SECTION command.

See Also

SECTION	Displays different sections of a body with pin numbers.

PROPERTY

PROPERTY Attaches a property name and value to a specified vertex of an object.

Syntax PROPERTY
(*attach_pt* (*name value location_pt*)...)...

Description

Properties allow you to associate information with selected objects on a drawing. The information is passed to other design programs for processing and analysis. A property consists of a name-value pair that is attached to an object: a body, pin, wire, or a signal name.

Property names can be any string of alphanumeric characters and underscores, provided that the first character is an alphabetic character. A property name cannot contain any spaces or punctuation except for the underscore.

The property value can be any string of text, including spaces and marks of punctuation. As is described in the Language Reference Manual, there are no restrictions on the use, names, or values of properties. Certain kinds of properties, such as SIZE, are known to the Graphics Editor and are handled in a consistent manner. Properties that are not known to GED are passed to other processors such as the Compiler and Timing Verifier.

Each property attached to a given object must have a unique name. If a newly entered property has the same name as a property currently attached to a vertex, the new property value replaces the old property value.

To specify a property:

1. Select or type the PROPERTY command.

2. Specify the object (vertex) where the property is to be attached.

3. Type the name and value of the property. The name and value can be separated by a space, an equal sign (=), or typed on separate lines.

4. Specify the location on the drawing where the text of the property value should appear.

When a property is added to a drawing, only the property value appears. The SHOW PROPERTIES command temporarily displays the names and values of all properties on the drawing. The DISPLAY command changes the permanent display of property name and value pairs.

You can manipulate the properties you add to a drawing with the SWAP, REATTACH, COPY, MOVE, and DELETE commands. Default body properties and the properties produced by the BACKANNOTATE, PINSWAP, and SECTION commands cannot be manipulated.

See Also

SHOW ATTACHMENTS	Displays the connections between properties and the objects to which they are attached.
DISPLAY	Changes the way objects are displayed on a drawing.
SWAP	Swaps the position of two lines of text.
REATTACH	Reattaches properties from one object to another.
COPY	Copies objects, properties, and groups in the current drawing.
MOVE	Moves objects from one position to another.
DELETE	Removes objects from a drawing.

QUIT

QUIT	Allows you to end the editing session. Same as EXIT.
Syntax	QUIT

Description

This command terminates an editing session. GED displays a message if there are unwritten changes to the drawings in the current editing session. Issue the QUIT command twice in succession to override the warning, discard all changes, and terminate the session.

See Also

EXIT	Allows you to end the current editing session. (Same as QUIT.)
WRITE	Writes the current drawing to the disk.

REATTACH

REATTACH Reattaches properties from one object to another.

Syntax REATTACH *pt pt..*

Description

The REATTACH command reattaches properties (including signal names) from one object to another. For example, you can use the REATTACH command to attach a property from the input pin of a part to the output pin.

To use the REATTACH command:

1. Type REATTACH.
2. Select the property to be moved. A line is drawn from the property to the current cursor position.
3. Specify the new attachment point for the property.
4. Use the MOVE command to position the property closer to its new attachment point.

Default body properties and those produced by the BACKANNOTATE, PINSWAP, and SECTION commands cannot be reattached. An error message is displayed when you attempt to reattach one of these properties.

See Also

SHOW ATTACHMENTS
Allows you to verify that the properties you reattached are attached to the correct objects.

MOVE Allows you to reposition objects on a drawing.

REDO

REDO Reverses the last UNDO command.

Syntax **REDO**

Description

The REDO command undoes the previously issued UNDO operation. The SCALDsystem keeps a list of operations performed during the current editing session in a log. The UNDO and REDO commands perform their functions according to the log.

See Also

UNDO Undoes the previous graphics operation.

REMOVE

REMOVE	Deletes a drawing from a SCALD directory.
Syntax	REMOVE [<*dir*>] *dwg*[.[*type*] [.[*version*] [.[*page*]]]]
Default	<*dir*> *drawing*.*.*.* (Deletes all types and versions of the specified drawing in the SCALD directory.

Examples

REMOVE *drawing1*

> Deletes all drawing types (SIM, LOGIC, BODY) of *drawing1*.

REMOVE *drawing2*.logic.*

> Deletes only the LOGIC type of *drawing2*.

REMOVE *drawing3*.logic.*.1

> Deletes only the first page of LOGIC drawing, *drawing3*.

Description

This command deletes a drawing from a SCALD directory. To delete a drawing:

1. Type REMOVE and the name of the drawing to be deleted. Wild cards can be used in the drawing name.

2. Press the ENTER key.

   GED displays the names of the files to be deleted. Because you can specify only one argument with the REMOVE command, repeat Steps 1 and 2 to delete additional drawings.

3. Type a semicolon on the command line and press the ENTER key.

   or

   Select the semicolon from the GED menu.

   The directory entries are deleted and the files are purged.

To cancel the REMOVE command, type: ABORT <cr>, or select any command except semicolon from the menu. The message ''Nothing done'' is displayed.

Wild cards are allowed in the file names specified in the REMOVE command. An asterisk (*) matches anything, and a question mark (?) matches any single character.

If just the name of the drawing is given with no wild cards, all drawing types (BODY, LOGIC, SIM) and files (ASCII, binary, dependency, connectivity) are deleted.

If no SCALD directory is given, REMOVE searches for the specified drawing in the currently active SCALD directory.

REPLACE

REPLACE Substitutes one part for another.

Syntax REPLACE [<*dir*>] *body_name*[.*ver*] *pt..*
REPLACE [<*dir*>] *body_name*[.*ver*] *group-name* or *bodyname...*

Description

The REPLACE command is used to substitute one part for another. There are many ways to use the REPLACE command.

You can enter the name of the replacement part and then use the cursor to point to the body or bodies to be replaced.

You can select the bodies to be replaced with the cursor. Then enter the name of the replacement body at the keyboard. Each body you selected with the cursor is replaced by the specified body.

You can also use the FIND command to group all the occurrences of a body to be replaced. Then use the groupname with the REPLACE command to globally change all the occurrences of the body. A message displays the number of bodies that are replaced.

Pin properties are reattached if a pin name on the new part is the same as a pin name on the first part. If the pin names do not match, the pin property becomes a body property.

All properties except those generated by the BACKANNOTATE, SECTION, and PINSWAP commands are kept. All default properties that have a value of ''?'' receive the value of the property with the same name on the replaced body (if one exists). Wire connections to the original part are kept only if the pins are in the same place. The rotation of the original body is observed when the body is replaced.

RETURN

RETURN	Returns to the previously edited drawing.
Syntax	RETURN

Description

This command causes GED to return to a previously edited drawing. If the current drawing is modified but not written, the system saves a copy of that drawing before returning to the previous drawing.

See Also

SHOW HISTORY	Lists the drawings that you edited during the current session.

ROTATE

ROTATE	Rotates a body or text string 90 degrees with mirrors at 180 and 270 degrees.
Syntax	ROTATE *pt..*

Description

The ROTATE command creates rotated and mirrored versions of a selected body. When a body is rotated, all notes and properties are also rotated. Text strings can be also rotated or justified independently.

To rotate a body, type ROTATE and then point to the part or text string to be rotated. Each time you press the button, the part rotates 90 degrees.

Rotating some parts 180 degrees reverses the order of the pins. This can cause subtle errors in your designs if pins become incorrectly wired. Therefore, a 180 degree rotation of a part becomes a mirror of a 0 degree rotation (about the Y axis). A 270 degree rotation of a part is a mirror of a 90 degree rotation (about the X axis).

In the 90 degree rotation, body notes are rotated 90 degrees and left in their original justification.

For the mirrors, justified text is shifted from left to right, right to left, and no further rotation is done. Text rotations (properties and drawing notes) are actually rotated, not mirrored.

See Also

MIRROR	Creates a mirrored version of the selected body.
SPIN	Provides true rotations, not mirrors, of a body.

SCALE

SCALE	Smashes a drawing and includes it in the current drawing.
Syntax	SCALE (*pt pt*) *drawing_name*

Description

The SCALE command adds a specified drawing to the current drawing in the rectangle indicated by two points. The drawing is smashed (all bodies turned into wires, arcs, and text). SCALE is useful for doing documentation drawings.

When a drawing is smashed, all connectivity information is lost. The drawing can no longer be interpreted by the Compiler.

See Also

FORMAT	Combines a text file with referenced drawings.
Section 7	Contains more information about creating mixed text and graphics documents.

SCRIPT

SCRIPT Performs the GED commands listed in the specified text file.

Syntax SCRIPT *file_name*

Description

The SCRIPT command allows you to specify GED commands in a script file. This is most frequently, but not always, used to initialize the list of working directories.

An example of a script file is ''startup.ged.'' This special file is expected by GED as an initialization script. If that file does not exist, a warning message is displayed when GED begins.

SECTION

SECTION Displays different sections of a body with pin numbers and assigns path properties.

Syntax SECTION [*pin_number*] *pt...*

Description

The SECTION command allows you to assign a physical part section to a selected logical part. As you step through the different sections of a body, the pin numbers of each section are displayed on the drawing. Sectioning a part automatically assigns path properties to the drawing.

If the part selected can be assigned to a section, the pin numbers for the selected section are displayed on the drawing. If the same part is selected again, the next section is selected and the new pin numbers are displayed. Thus, by pointing to the same part, you can step through all the different possible sections.

To assign a specific section directly, type in a pin number that uniquely defines the section and then point at the part. This allows you to avoid stepping through each section individually.

Currently, you can only section parts with SIZE = 1 or HAS_FIXED_SIZE parts. Assigning sections to a HAS_FIXED_SIZE part is accomplished by pointing to the pin of the section to be assigned. It is an error to point to the body of a HAS_FIXED_SIZE part.

The SECTION command uses the information in the library's chips file to display the pin numbers. You can only perform SECTION on parts from libraries with this file. See Section 6 for more information about the chips_prt file.

See Also

PINSWAP	Swaps pin numbers on a body that are defined to be in the same pin group.
BACKANNOTATE	Annotates the design with physical information from the Packager.

SET

SET Establishes the default options for GED.

Syntax **SET** *option*

Description

The SET command establishes the default options used by GED. The commands can be issued during an editing session or placed in the startup.ged file. To see a list of the current settings and options, type **SET** and press the ENTER key.

SET has many options that allow you to tailor GED to your particular requirements.

SET <**cr**>	Lists the current settings.
SIZE *scale_factor*	Changes the default size of entered text. The default text size is 0.082 inches (size 1). The maximum height is 2 inches or SIZE 24.
ASCII **NOAS**CII **BI**NARY **NOBI**NARY **CO**NN **NOCO**NN **DEP**ENDENCY **NODE**PENDENCY	Specifies the types of files that are written when a drawing is saved. Default files are: ASCII, binary, connectivity, and dependency. Unwanted files can be turned off temporarily and then reset.
ORTHOG_WIRE	Sets the wiring mode to orthogonal (bended) wires (default).
DIRECT_WIRE	Sets the wiring mode to non-orthogonal (diagonal) wires.

STOP_AT_PIN	Ends a wire when it reaches a pin (default).
GO_AT_PIN	Continues the wire from a pin. You must press the left cursor button twice to end the wire.
MOVE_ORTHOG	Uses orthogonal (bended) wires when an object is moved (default).
MOVE_DIRECT	Uses diagonal wires when an object is moved.
DOTS_OPEN	Displays dots as small open circles (default).
DOTS_FILLED	Displays filled (solid) dots.
STICKY_OFF	If a default property is deleted from a BODY drawing, this option deletes the property from a LOGIC drawing when it is read into GED (default).
STICKY_ON	Converts default body properties into non-default properties on a LOGIC drawing.
CAPSLOCK_OFF	Allows GED to interpret all input as it is entered at the keyboard (default).
CAPSLOCK_ON	Allows GED to interpret all input as upper case regardless of how it is entered at the keyboard.
GRID_OFF **GRID_ON**	Controls whether or not the grid is displayed when GED is entered and a drawing is edited. The default is OFF.

DECIMAL	Bases plots on the decimal system with 500 internal units per physical inch on the Versatec plotter. The grid multiple on the status line of the display is grids per inch.
METRIC	Bases plots on the metric system with 512 internal units per inch or 20 internal units per millimeter. The grid multiple on the status line of the display is grids per millimeter. This remains compatible with standard Valid libraries since pins are on 2.5 mm centers.
FRACTIONAL	Sets the default internal divison of 500 units per inch to 400 units per inch. Valid libraries remain compatible with bodies 25% larger and pins on 1/8 inch centers.
DEFAULT_GRID *gridsize*	Changes the default grid size for drawings other than BODY and DOC. The default is 0.01.
FONT *FONT*	Allows you to specify the text font to be used in the hard copy of the drawing. Seven fonts are available: VECTOR_FONT (default), MILSPEC_FONT, GOTHIC_FONT, CURSIVE_FONT, GREEK_FONT, SYMBOL_FONT. All text in the drawing is set in the same font. Non-default fonts are available only in HPR mode.

V11VERSATEC **V22**VERSATEC **V36**VERSATEC **V42**VERSATEC MONO_HPPLOT **B9424** **CALCOMP1043** **CALCOMP5744** **HP7475** **HP7580**	Allows you to set your system for the appropriate Versatec, Benson, CalComp, or Hewlett Packard plotter. The default is an 11-inch Versatec.
LOCAL_PLOT SPOOLED_PLOT	Determines whether spooling is immediate (local) or delayed (spooled). The SPOOLED option creates a plot file, **vw.spool**, for the type of plotter specified with the SET command. To plot the files stored in vw.spool on a local printer, use the UNIX **lp** command. The default is LOCAL_PLOT (See Section 5).
PLOTTER EPSONLQLR **PLOTTER EPSONLQMR** **PLOTTER EPSONLQHR**	Specifies the EPSON printer attached to the system. LR = low resolution; MR = medium resolution; HR = high resolution.
DOUBLE_WIDTH SINGLE_WIDTH	Governs the darkness of plotted lines. The double width option (default) prints two pixels instead of one.
LEFT_JUSTIFICATION RIGHT_JUSTIFICATION	Sets the justification of text strings (properties and notes). The default is left justified.

USER_SIM *simfile*	Allows you to specify the UNIX pathname of an alternate Simulator file. The default is /u0/scald/simulator/sim.
COLOR_ARC *colorlabel* **COLOR_BODY** *colorlabel* **COLOR_DOT** *colorlabel* **COLOR_NOTE** *colorlabel* **COLOR_PROP** *colorlabel* **COLOR_WIRE** *colorlabel*	Sets the color for the specified object. The default for all objects is monochromatic. Enter the name of the *colorlabel.* Valid selections are: aqua, blue, brown, peach, green, gray, mono, orange, pink, purple, red, salmon, skyblue, violet, white, and yellow.

SHOW

SHOW	Temporarily displays the specified drawing information.
Syntax	SHOW [*option*]

Description

The SHOW command displays classes of objects. The effect of the SHOW command is temporary; information displayed with this command disappears when the drawing is written to the disk file or when the screen is redrawn.

To see a list of all the SHOW options, type SHOW followed by a carriage return.

ATTACHMENTS	Displays the connections between properties and the objects to which they are attached.
BODY *pt*	Displays the name, version, and the SCALD directory of the indicated part.
COLOR *pt*	Displays the color of the specified object.
COORDINATE *pt*	Displays the GED coordinates of an indicated point.
CONNECTIONS	Displays wire connections in the drawing.

GROUP [*pt*] or [*''group_name''*]

Causes the specified group to be highlighted. You can either point to the required group with the cursor or type the name of the group. Specifying a point with the cursor displays the group name and causes the closest group to be highlighted.

Specifying a group name highlights the group. Also, the SHOW GROUP command lists the number of bodies, notes, properties, dots, arcs, and wires that the group contains.

HISTORY

Lists all the drawings you edited and shows which are modified but unwritten. SHOW HISTORY also lists the drawing the RETURN command returns to.

KEYS

Lists the function keys and the corresponding text string that has been assigned to each key.

NET [*net_name*] or [*pt*]

Lists the name of the indicated net and highlights its segments. The net can be specified by name or by pointing to a net with the cursor.

ORIGINS

Displays the origins of bodies on the drawing.

PINS

Displays the pin connection points on bodies.

PROPERTIES

Shows both the name and value of all of the properties on the drawing. Since signal names are handled internally as properties attached to the wire, the use of SHOW PROPERTIES displays the text "SIG_NAME =" with each signal name.

PWD

Lists the UNIX directory from which the current GED session originated.

RELEASE

Displays the release number of GED.

SIZE *pt*	Shows the amount by which the display size of the characters in the indicated text string has been modified. This size is the multiple of the default text size (0.082, unless a SET option has been used to change the default).
VECTORS *pt*	Displays the pin names from the body definition of the indicated part.

SIGNAME

SIGNAME Attaches signal names to wires or pins.

Syntax SIGNAME (*signal_name ... pt...)...*

Description

The SIGNAME command allows you to attach signal names to wires or pins. To attach a signal name:

1. Select or type the SIGNAME command.

2. Use the cursor to identify the location for each signal name. An asterisk is drawn at each location.

3. Type the text for the signal name. You can enter up to 80 characters, including spaces.

4. Press the ENTER key. As long as there are points remaining, the text lines you enter are interpreted as a signal names.

Alternately, you can issue the command, type in one or more signal names, and then specify points to place the signal names on the drawing.

The signal name is attached to the wire or pin that is closest to the specified point.

Internally, GED handles signal names as properties. For example, attaching a signal called ''BUS ENABLE'' to a wire is equivalent to attaching a property ''SIG_NAME=BUS ENABLE'' to that wire.

See Also

Section 4	Contains more information about properties and signal names.
Language Reference Manual	Contains details about signal name syntax.
PROPERTY	Attaches a property name and value to an object.

SIMULATE

SIMULATE Allows you to run the simulator program for the current drawing.

Syntax SIMULATE

Description

The SIMULATE command is used to invoke the simulator. The SIMULATE command creates a Simulator window in the lower portion of the screen and establishes communication with the GED window. To exit from the simulator, type EXIT in the simulator window or ENDSIM in the GED window.

The Simulator is optional software that may not be included with your system.

See Also

Simulator Reference Manual
Contains a thorough discussion of the Simulator program.

SMASH

SMASH	Breaks a body into the objects that define it.
Syntax	SMASH *pt*..

Description

The SMASH command breaks a body into separate lines, arcs, and notes. Any properties attached to the body are deleted. The SMASH command is useful for creating library body drawings. You can only use this command on bodies.

For example, you can create N-input AND gates from a 2-input AND gate with the following commands:

1. **edit** N AND.body
2. **add** 2 AND *pt*
3. **smash** *pt*
4. Attach the N inputs and write the drawing.

Normally, you cannot add a body to a body drawing. Using the SMASH command changes the 2 AND body into its separate elements so that GED does not interpret it as a body when the N AND drawing is written.

See Also

SCALE	Adds a drawing to a text file.
ADD	Adds a BODY to a drawing.

SPIN

SPIN	Provides true rotations, not mirrors, of a body.
Syntax	**SPIN** *pt..*

Description

The SPIN command is used when a true rotation of a body is needed. This command rotates the body 0, 90, 180, and 270 degrees without mirroring any of the four representations. This reverses the pins on bodies, which can cause errors in the drawing.

See Also

MIRROR	Creates a mirrored version of a selected body.
ROTATE	Rotates a body or text string 90 degrees with mirrors at 180 and 270 degrees.

SPLIT

SPLIT Adds a segment to an existing wire and separates objects with common vertices.

Syntax SPLIT *pt pt..*

Description

The SPLIT command can be used to perform two functions:

- Split a single wire into two wires by adding a vertex along that wire.
- Separate objects that have been placed at the same vertex (co-located objects).

  For example, the SPLIT command can be used to disconnect a wire from one pin and move it to a different pin.

To split a single wire into two wires:

1. Type or select the SPLIT command and select a point along the wire.

   This procedure adds a vertex along the wire between the original two vertices.

2. Move the vertex to the new location.

3. Place the new vertex by specifying a second point with the cursor.

To disconnect items that are co-located:

1. Type or select the SPLIT command and select the desired vertex with the cursor (press the right button).

   The selection attaches one of the objects to the cursor so it can be moved about on the screen.

2. To move another object, select the original vertex again and pull off the second object.

   You can continue to select objects at that vertex until the correct item has been selected.

3. Place the object at a new location by moving the cursor and pressing the appropriate button.

   When all the objects have been split off the vertex, select the vertex one more time to place down the last item. One more selection begins the cycle again, splitting off each item in turn.

Whenever possible, the SPLIT command attempts to operate on wires.

SWAP

SWAP	Exchanges the position of two lines of text (properties or notes).
Syntax	SWAP *pt pt..*

Description

The SWAP command is used to swap two properties or two notes. Only two notes or two properties can be swapped, not a note and a property. Default properties and those generated by the PINSWAP, SECTION, and BACKANNOTATE commands cannot be swapped.

See Also

NOTE	Adds textual information to a drawing.
PROPERTY	Attaches properties to the objects in a drawing.

UNDO

UNDO	Undoes the operation of the previous command affecting the drawing.
Syntax	UNDO

Description

The UNDO command undoes the previous operation affecting the drawing. GED keeps a list of operations performed during the current editing session. Repeated applications of UNDO reverses the effects of events according to this list. Each read or write of a drawing causes the UNDO log to be reset; therefore, UNDO cannot undo operations on drawings earlier than the last read or write.

See Also

REDO	Reverses the last UNDO command.

UNIX

UNIX Allows you to access the UNIX shell.

Syntax UNIX <cr>
UNIX *command*

Description

This command allows you to access the UNIX shell on your system where you can issue UNIX commands. Type **Control-D** or **exit** to exit from the UNIX shell and return to GED.

You can also execute a particular UNIX command from GED by specifying the command following UNIX. A prompt instructing you to press the ENTER key is displayed, unless the command is executed from a script.

USE

USE	Specifies a working directory.
Syntax	USE *directory_name*

Examples

USE /u0/job/common.wrk

> Specifies the SCALD directory common.wrk in the UNIX directory /u0/job.

USE project1.wrk

> Specifies the SCALD directory project1.wrk in the current UNIX directory.

Description

The USE command allows you to specify a SCALD directory from which you can retrieve and store drawings. This directory is placed at the top of the active search list and becomes your current working directory. There is no limit to the number of directories that can be in use at one time.

To USE a SCALD directory other than one in the current UNIX directory, the UNIX pathname must be given.

See Also

IGNORE	Deletes the specified directory or library from the active list.
LIBRARY	Specifies the component library to be accessed.
Section 2	Explains the file and directory structures of GED.

VECTORIZE

VECTORIZE	Creates a file in vector plot format for the current drawing.
Syntax	VECTORIZE

Description

The VECTORIZE command creates a file called vector.dat that contains the current drawing in vector format. This file can be used to transmit files to other machines or drive a pen plotter (with the aid of a format conversion program).

See Also

Appendix A	Contains more information about vector plot format.

VERSION

VERSION Selects an alternate version of a body.

Syntax VERSION *pt..*

Description

The VERSION command allows you to select alternate versions of appropriate bodies. Some bodies can be created with several different symbolic representations. For example, the NAND gate is equivalent to an INVERT-OR gate by DeMorgan's Theorem. Similarly, a NOR gate is equivalent to an INVERT-AND gate. The versions of a body all refer to the same logic drawing.

To step from one representation of a body to another, issue the VERSION command and then select the body with the cursor. GED determines which version of that body is currently displayed and replaces it with the next version in the sequence. Continue to press the appropriate button to cycle through all the available versions. After the last version of the sequence is displayed, the first version is redisplayed.

You can also use the FIND command to group all the occurrences of a specified body. Then issue the VERSION command with the groupname to globally change the drawing. The center button versions the bodies in the group closest to the cursor.

See Also

REPLACE Substitutes one device for another.

ADD Allows you to add a specific version of a body directly to a drawing.

WINDOW

WINDOW	Changes the view of the current drawing.
Syntax	WINDOW [*option*] ;

Examples

WIN ;	Redraws the screen and clears error and status messages.
WIN FIT	Fits the entire drawing to the screen.
WIN *pt*;	Defines a point on the drawing as the new center of the screen, allowing you to pan across the drawing.
WIN *pt pt*;	Fills the screen with the area of a rectangle defined by the two points, allowing you to zoom in on parts of the drawing.

WIN *pt1 pt2 pt3*

Changes the center and scale of the drawing. Pt1 is the new center, and the drawing is either enlarged or reduced based on the ratio of the distance between point 1 and points 2 and 3.

WIN *scale_factor*

Scales the window by the value of the scale factor. For example, WIN 1.5 enlarges the window one and a half times.

WIN *-scale_factor*

Reduces the drawing by the value of the scale factor. For example, WIN -2 reduces the drawing by one half. You can specify a scale factor of 0.5 to achieve the same result.

Description

The WINDOW command is used to change the view of the drawing on the screen. This command can be used with up to three arguments. If there are fewer than three arguments, the command must be terminated with a semicolon. You can either select the semicolon box on the on-screen menu with the cursor or type a semicolon followed by the ENTER key.

If you issue the WINDOW command followed by a semicolon, GED redraws the image without changing the center or the scale. This option is useful if error messages cover part of the drawing.

The WINDOW command with an argument of one point causes that point to become the center of a new screen display of the drawing. The scaling of the drawing remains the same.

The WINDOW command with an argument of two points defines a rectangle with the specified points at opposite corners. The rectangle expands to fill the screen providing a close-up view of the specified portion of the drawing.

You can issue the WINDOW command with three points. The first point defines the new center of the drawing and the display becomes either larger or smaller, depending on the ratios of the distances between the other points. If the distance between point 1 and point 3 is greater than the distance between point 1 and point 2, the items appear larger; if the distance is smaller, items appear smaller.

Type **WINDOW FIT** and press the ENTER key to fit the entire drawing to the screen. Alternatively, this command can be issued by selecting WINDOW from the menu and typing **FIT** <cr> on the keyboard.

You can specify an integer or a real number as the argument to the WINDOW command to scale the view of the drawing by the amount entered. The center of the window remains the same. For example:

WIN 2; <cr>	Makes the drawing appear twice as large.
WIN -2; <cr>	Reduces the drawing by a factor of two.
WIN 1.5; <cr>	Enlarges the drawing one and a half times.
WIN 0.5; <cr>	Has the same effect as WIN -2.

See Also

Section 4	Contains more information about window and display functions.

WIRE

WIRE Adds wires to a drawing.

Syntax WIRE (*pt pt..*)...

Description

The WIRE command is used to add wires to a drawing. The wire begins at the first point specified and runs to the second. Additional points are specified to draw a wire with one or more segments.

Because schematics almost exclusively use orthogonal wires, the default wire mode is orthogonal (bent). Once the wire is started and the cursor changes direction, the attached wire remains orthogonal, whether the cursor is moved horizontally, vertically, or diagonally. Press the center button to change the orientation of the bend. If the center button is pressed a second time, the wire becomes diagonal. A third press returns the wire to the first orthogonal position.

To end the wire, a zero length segment is specified by pressing one of the cursor buttons twice at the final point.

To bend a wire, press the left button.

To snap the wire to the nearest pin, press the right-hand button.

The **SET DIRECT_WIRE** command can be typed at the keyboard or added to your startup.ged file. In this mode, all wires are diagonal until they are placed down. Also, finishing a wire with the left or right-hand button creates a diagonal wire. Ending a wire with the center cursor button creates orthogonal wire segments to the nearest grid point.

You can return to the automatic orthogonal wiring mode by typing the **SET ORTHOG_WIRE** command.

To indicate wire connections, you can use the DOT or AUTO DOT command. In GED, a T-junction is automatically a connection whether or not it is dotted. A four-way intersection (+) is not a connection unless it is dotted.

See Also

Section 4	Contains more information about adding wires to a drawing.
SET	Allows you to set default wire options.
DISPLAY	Allows the display of wires to include buses and patterned lines.
SHOW CONNECTIONS	Temporarily highlights all wire connections in your design.

WRITE

WRITE Writes the current drawing onto the disk.

Syntax **WRITE** [< *dir*>] *dwg*[.[*type*][.[*version*][.[*page*]]]]

Examples

WRITE — Saves the current drawing named on the status line in the current SCALD directory.

WR newname — Stores the current drawing in the current SCALD disk directory. If a drawing named ''newname'' already exists, a message is displayed. Type or select a semicolon (;) to overwrite the existing drawing, or type ABORT or select any other command to cancel WRITE.

WR <project2.wrk> — Writes the current drawing into the SCALD directory, project2.wrk.

Description

This command saves the current drawing on disk. If no drawing name is specified, the drawing is given the drawing name specified on the status line at the top of the display.

You can specify a different drawing name with the WRITE command. If a drawing with that name is already in a SCALD directory, a warning message is displayed. Select WRITE again to overwrite the existing drawing with the new drawing. Select any other command to cancel the WRITE command.

If no directory is given, the drawing is written to the SCALD directory from which it was retrieved. If the drawing is a newly created drawing and no directory is given, the drawing is written to the current directory.

See Also

EXIT	Leaves the editor.
QUIT	Leaves the editor.
DIAGRAM	Allows you to rename a drawing.
USE	Specifies a working directory.
LIBRARY	Specifies the component library to be accessed.

APPENDIX A
GED FILES

This section contains information about the files and file structures used by GED.

A.1 SYSTEM INTIALIZATION FILES

Several system files initialize the Graphics Editor when you enter the system.

/u0/editor/startup.ged

This is the system wide initialization file. It defines some of the directories referenced by GED, and it is referenced by all users of the system. Currently, /u0/editor/startup.ged also defines the keys on the workstation keyboard.

/u0/editor/softkeyassign

This file contains the default values of the function keys. See the ASSIGN command for more information about programming the function keys.

/u0/scald/config.dat

This file is used by GED for the creation of unnamed signal names. It defines the characters that are used for signal name definitions (such as the asterisk to indicate a low asserted signal).

/u0/lib/master.lib

This file contains the name translations for the Valid part libraries. The file entries contain the abbreviated names for the GED LIBRARY command and the UNIX pathname to the location of the libraries.

For example, an entry in master.lib appears:

```
sttl /u0/lib/sttl/sttl.lib;
```

This entry makes a permanent ''alias'' for sttl so that it always refers to the pathname listed above. Consequently, instead of typing **USE /u0/lib/sttl/sttl.lib**, you can type **LIBRARY sttl.**

A.2 ASCII FILES

One of the design data base files that GED creates when a drawing is written is an ASCII file. An ASCII file is a script file that can be used to generate any drawing except for BODY drawings. It is a specific type of text file that consists of commands to add each object in a drawing.

Points are represented in ASCII files by their coordinates, enclosed in parenthesis. Thus, the point x=100, y=200 is represented by (100 200).

Angles are represented by a number from zero (0) through seven (7):

0: 0 degrees

1: 90 degrees

2: Mirror of 0 degrees

3: Mirror of 270 degrees

4: 180 degrees

5: 270 degrees

6: Mirror of 180 degrees

7: Mirror of 90 degrees

ASCII files basically consist of an identification line, commands to represent the type and location of each object in the drawing, and a QUIT statement.

FILE IDENTIFICATION AND END STATEMENTS

Each ASCII logic file starts with this line to identify the type of file to the system:

```
FILE_TYPE = MACRO_DRAWING;
```

The file ends with a line containing:

```
QUIT
```

BODY DEFINITIONS

Body definitions use up to four lines in the ASCII logic files. The description of the body in the ASCII logic file follows the form listed below:

FORCEADD *name*
[R *angle*]
pt ;
[PAINT color *pt*]

The name includes the version. The angle and paint definitions are optional. FORCEADD is used so that a placeholder is created if the body is not found.

WIRE DEFINITIONS

Descriptions of wires in the ASCII logic files consist of a single line that follows the form:

WIRE *linetype pattern pt pt* ;

The *linetype* includes both the color information and whether the wire is thin or heavy. If the number is converted to binary, the least significant bit is the thin/heavy bit (0 = thin, 1 = heavy). The remaining seven bits specify the color.

-16384 = *pattern* 16384. If the pattern = -1, the line is filled. There are six defined patterns in GED:

1: -1

2: 273

3: 682

4: 2175

5: 3135

6: 4383

DOT DEFINITIONS

DOTS are written out as:

DOT *type pt* ;

Type = 0 if the dot is open and 1 if it is filled. (If the type is not 0 or 1, the dot is assumed to be open.) If the dot is colored, it is followed by a PAINT command.

CIRCLE AND ARC DEFINITIONS

CIRCLES and ARCS are written out as:

CIRCLE *pt1 pt2* ;

or

CIRCLE *pt1 pt2 pt3* ; (for arcs)

Points are represented by the x,y coordinates that describe a location on the drawing. If the circle is colored, it is followed by a PAINT command.

NOTE DEFINITIONS

NOTES are represented as follows:

FORCENOTE
contents
pt angle;

The FORCENOTE command is similar to the NOTE command in the editor except that the FORCENOTE command terminates after reading one note. If the note is not the default size, the following line is added:

DISPLAY *size pt*;

This command makes the text the correct size. If the note is colored, it is also followed by a PAINT command.

PROPERTY DEFINITIONS

Property definitions occur directly after the object they are attached to. The format is:

FORCEPROP *default_status* LAST *name value*
[R *angle*]
J *justification_type*
pt;

The FORCEPROP command is similar to the PROPERTY command in the editor except it takes a *default_status* flag. This flag is necessary for the correct handling of changes to properties on library bodies. The *default_status* flag can have three values:

2 The property is known to be non-default (one that a user added to the ASCII logic drawing)

1 The property is known to be default (one that comes from the body definition)

0 (zero) The status of the property is unknown (an undefined variable whose status is determined when the body definition is searched).

LAST is a keyword indicating the property is to be attached to the last object or wire entered.

If the property is a PIN property, then the keyword LAST is replaced by the keyword LASTPIN followed by a *pt* describing the location of the pin in absolute coordinates.

If the property is attached to another property, then the keyword LAST is replaced by the keyword LASTPROP.

The angle is optional.

The next line describes the text justification.

0 Means the text is left justified

2 Means the text is right justified.

If no justification is given, the property is created with the current default justification. If an illegal justification is given, the system uses left justification as a default.

If the property does not have the standard visibility, it is followed by a DISPLAY command to set the visibility of the name and the value.

If the property is colored, it is followed by a PAINT command.

BUBBLED PIN DEFINITIONS

Bubbled pins for an object are written out using the format:

FORCEBUBBLE *pt* ....

All pins that are not in their default bubbled state are listed.

A.3 BINARY FILES

Binary files contain the same information as the ASCII file described above, but in a binary format that is quicker to read and write. This format is proprietary and is not described in this document.

A.4 BODY FILES

Body files contain descriptions of bodies in ASCII format. Body files are written out in an abbreviated format; they are not tolerant of errors. Bodies are composed of seven elements: Lines, arcs, text, connections, bubble groups, body properties, and pin properties. As in the ASCII logic files, all coordinates are in 0.002 inch units.

LINE DEFINITIONS

Lines require 1 line each in the body file. A thin line has the format:

L *x1 y1 x2 y2 pattern color*

A thick line has the format

M *x1 y1 x2 y2 pattern color*

The pattern is optional and -16384 = *pattern* 16384. If the pattern is -1, the line is filled. See the pattern definition listed with the description of wires in the ASCII file description. Color is the GED color number.

ARC DEFINITIONS

Arcs require one line each in the body file. The line has the format:

A *Xcenter Ycenter Radius Start_angle Stop_angle color*

The center and radius are in integer units, and the start and stop angles are measured in degrees counterclockwise from the X axis. They are in floating point. *Color* is the GED color number.

TEXT STRING DEFINITIONS

Text strings require two lines each in the body file. The first line contains the specification of the text; the second contains the contents. The first line has the format:

T *x y angle slant size over inv just font Nch color*

x,y	integer	reference point for text
angle	real	0.00, 90.00, 180.00, 270.00
slant	real	(not implemented)
size	integer	height of characters
over	0-1	(not implemented)
inv	0-1	(not implemented)
just	0-2	0=left justified, 2=right justified
font	0-4	(not implemented)
Nch	integer	number of characters

The next line consists only of the characters of the text string.

CONNECTION DEFINITIONS

Connections require one line each in the body file. The contents of the line depend on whether the pin is bubble-able or not. The format is:

C *x y Name dispx dispy bubbleable [default_state x2 y2 x3 y3]* f size angle just

The portion in brackets is present only if the pin is bubble-able.

x and *y* are the location of the connection.

Name is a quoted string containing the name.

dispx and *dispy* are the locations of the name.

Bubbleable and *default_state* are both integers, 1 if TRUE and 0 if FALSE.

x2,y2 and *x3,y3* are the endpoints of the bubbleable pins.

f is 1 if the dot on the connection is filled, 0 if open.

Size is the size of the pin name string (41 is GED's default).

Angle is the angle of the pin name string attached to the connection (0 = 0 degrees, 1 = 90 degrees, 2 = 180 degrees, 3 = 270 degrees).

Just is the justification of the string (R = right, L = left).

For a bubbleable pin, the entry might look like:

C 50 50 "Y<0>" 100 50 1 0 100 50 50 50 0 41 0 R

For a non-bubbleable pin, the entry might look like:

C 50 50 "Y<0>" 100 50 0 0 41 2 L

BODY PROPERTY DEFINITIONS

Body properties require one line each in the body file. The format is:

P *name value x y angle slant size over inv just font NV VV IP color*

name	quoted string	name of property
value	quoted string	default value of property
NV	0-1	name is visible by default
VV	0-1	value is visible by default
IP	0-1	1 if property is a parameter

The other numbers describe the text in the same manner as the T (text) command. Color is the GED color number.

PIN PROPERTY DEFINITIONS

Pin properties require one line each. They are identical to body properties except they start with an X, rather than a P, and occur directly after the connection with which they are associated.

BUBBLE GROUP DEFINITIONS

Bubble groups require several lines each in the body file. They start with a line beginning with B and end with a line containing only the word END. Each bubble group is on a line by itself, with the format:

[*name1,name2,name3,...*]

All the names are quoted strings. If the bubble group is asymmetrical, the first comma is replaced by a colon.

A.5 CONNECTIVITY FILES

Connectivity files contain information that includes the names of the bodies, the names of the signals tied to their pins (with bubble state), and the properties that belong to the body. Connectivity files, which are in ASCII format, are the only files used by the Compiler.

There are four types of items in a connectivity file:

The header
The NET section
INVOKE commands
Comments.

Each connectivity file has the form:

FILE_TYPE = CONNECTIVITY;
{GED_Release: date and number}
[*expr property*]
[*nets*]
[*invokes*]
END.

The first line is the header, the second line is a comment, the third is the EXPR (expression) property from the drawing body, the fourth line begins the net section, the fifth line begins a section for the invoke commands, and the sixth is the END statement. The EXPR, net, and invoke sections are optional. The continuation character for lines in a connectivity file is a tilde (~). This character can occur anywhere in the line, even in the middle of words, but must be followed by <LF>. GED limits line length to 80 characters, so it automatically inserts a continuation character in lines longer that 80 characters.

COMMENTS

Comments begin with an open brace ({) and end with a close brace (}). They can appear anywhere in a connectivity file except in the middle of identifiers or quoted strings and can cross lines.

EXPRESSION PROPERTY ON A DRAWING BODY

The expression string is the expression property value from the drawing body. The format of the expression string is :

expr property ::= EXPR=*expression string*;

For example:

EXPR="SIZE=10";

NETS

Each time GED writes a connectivity file, it numbers all the nets. The NC net is always net 0 (zero). Unnamed signals are also numbered. The net numbers are not the same each time the connectivity file is written. The format is:

nets ::= *constant net_name_string* [*property_list*] ;

The *constant* is the net number.

The *net_name_string* is either the signal name for the net or the unnamed signal string created by GED. The *net_name_string* is always quoted.

The *property_list* is optional.

property_list ::= { *identifier string* }

The *identifier* is the property name; it can only contain letters, digits, and underscore (_), and must begin with a letter. There are two reserved identifiers: FILE_TYPE and END.

The *string* is quoted.

An example of several net entries is:

2"UN$1$2P$A";
3"ANWC"LOAD"37"CONNECTED_TO"PAGE 4";

INVOCATION OF COMPONENTS

Each component in the drawing is described as follows in the connectivity file:

invokes ::=
%*invoke_name_string*
version_str, *xy_str*, *rotation*, *directory_str*, *path_str*;
[*parameter_property_list*] ;
[*property_list*] ;
{ *pin_name_string* [*property_list*] *constant*; }

The *invoke_name_string* is the name of the component and is quoted.

The next line contains body properties that are always output: The body version number, in quotes, the coordinates of the body on the page, the rotation of the body, in quotes, the name of the directory where the body came from (not rooted, so /u0/lib/lsttl/lsttl.lib is shortened to lsttl.lib), and the path property. If any of these properties doesn't exist, the null string ("") is used. The rotation string is:

0: 0 degree rotation

1: 90 degree rotation

2: Mirror of 0 degrees

3: Mirror of 90 degrees

4: 180 degree rotation

5: 270 degree rotation

6: Mirror of 180 degrees

7 Mirror of 90 degrees

The *property_list* is optional, but the semicolon is not.

The *parameter_property_list* is optional and is in the same format as a property list.

The *pin_name_string* is quoted and the *constant* is the number of the net attached to it. The net numbers are assigned in the net section.

An example of an invoke:

```
%"LS00"                                  {body name}
"1","(100,345)","0","lsttl.lib","2P";    {body information}
SIZE"SIZE";                              {parameter property list}
COLOR"RED"SECTION"U32";                  {body property list}
"A"23;                                   {pin names}
"B"5;
"Y"OUTPUT_LOAD"(50.0,-50.0)"3;
```

An example with no path property string and no body property list:

```
%"LS02"                                  {body name}
"2","(500,1234)","3","lsttl.lib","";     {body information}
;                                        {parameter property list}
COLOR"RED";                              {body property list}
"A"23;                                   {pin names}
"B"5;
"Y"OUTPUT_LOAD"(50.0,-50.0)"3;
```

A.6 DEPENDENCY FILES

Dependency files list the UNIX directories that are the source of all bodies added to a drawing. These files are used by the update procedure, which allows drawings to be updated if any bodies are out of date. There are DEPENDENCY files for all drawings except BODY drawings.

The first line of a dependency file is the drawing's logic file name, followed by a colon (:), followed by a blank-separated list of body file names. The names are all UNIX file names with paths.

For example, a sample dependency file for the logic drawing MY EXAMPLE.LOGIC.1.1 is:

```
myexample/logic.1.1 : \<cr>
    /u0/lib/lsttl/ls00/body.1.1 \<cr>
    /u0/lib/lsttl/ls03/body.1.1 \<cr>
    adder/body.1.1 \<cr>
    shifter/body.1.1
```

The \<cr> is used to continue across lines. Lines continued in this way must begin with <tab>. It is assumed that files are referenced from the UNIX directory containing the SCALD directory file that holds the logic drawing. For example, from the directory /u0/class, you need only to say shifter/logic.1.1, not /u0/class/shifter/logic.1.1. However, parts that are added from SCALD directories not in the current UNIX directory must be given a full path, /u0/lib/lsttl/ls00/body.1.1. The entire path name must be written out, no wild cards are allowed.

The last line in the dependency file is:

/u0/editor/MakeAddToList
"*drawing_name.extension.version.page*"

The drawing identifier is quoted and all four parts of the name must be given.

A.7 BACK ANNOTATION FILE

This information discusses the format for the file read by the Graphics Editor BACKANNOTATE command and generated by the Packager. If you do not use the backannotation file generated by the Packager, there is no guarantee that the information is consistent with the physical design.

The back annotation file contains physical information grouped by drawing. The file must be called backann.cmd.

The first line is:

```
FILE_TYPE = BACK_ANNOTATION;
```

The last line in the file is:

```
END.
```

The information in the file includes drawing names, body names, pin names, and net names.

The back annotation file should not contain information for bodies with SIZE and/or TIMES properties except as follows:

- A LOCATION property for the body should be output only if ALL SIZE replicated logical sections of the body are allocated to the same physical part.
- Pin numbers for pins of SIZE replicated components should be output only if the pin is common to all sections and appears on the same pin for all.

There should be no information for a body that is used more than once in the design.

DRAWING NAMES

The line with the drawing name looks like:

DRAWING = "SCALD *drawing_name_extension.version.page*";

The drawing name must be quoted.

BODY NAMES

This is specified by giving the body's name and path property and any information to be attached to the body. If there is no information to be attached, the line should be:

BODY = "*name*","*path_property*";

If properties are to be attached, the above statement ends with a colon and is followed by property name/value pairs, separated by commas. For instance:

BODY = "*name*","*path_property*": *prop1* = "*value1*";

Property values are quoted, but not property names.

There MUST be spaces around any equals sign (=).

The only property that should be attached to a body is the LOCATION designator.

PIN NAMES

This includes the name of a pin on the body, as well as any information to be attached to the pin. Vectored pins cannot be annotated. The pin name should be quoted. For instance:

"*pin_name*": *prop1* = "*value1*";

Property names must be 15 characters or less. Property values are quoted, but not property names. There MUST be spaces around any equals sign (=). The only information given should be the pin number (PN property). If a pin does not have any properties, the pin should not be listed.

NET NAMES

Net names include the name of a net, in user syntax form, and any information attached to the net. Only scalar nets can be annotated. The form is:

NET = "*net_name*": *prop1* = "*value1*", ... *propN* = "*valueN*";

AN EXAMPLE

```
FILE_TYPE = BACK_ANNOTATION;
DRAWING = "C C.LOGIC.1.1";
BODY = "LS74","6P": LOCATION = "U32";
"CLOCK*": PN = "1";
"D": PN = "2";
BODY = "LS08","5P": LOCATION = "U34";
"Y<0>": PN = "1";
NET = "XOUT":
DRAWING = "C C 2.LOGIC.1.1";
BODY = "LS74","6P": LOCATION = "U34";
"CLOCK*": PN = "3";
"D": PN = "2";
BODY = "LS08","5P": LOCATION = "U32";
"Y<0>": PN = "7";
NET = "XOUT":
END.
```

A.8 VECTOR PLOT FORMAT

This information describes the format of the plot file produced with the Graphics Editor's VECTORIZE command. This command produces an ASCII plot file that can be used to transmit drawings to other machines or that can be used to drive a pen plotter (with the aid of a format conversion program).

The vector output is a plot of the entire drawing, not just the portion showing on the screen.

There are three different types of primitives in the plot file: LINES, ARCS, and TEXT_STRINGS. Each primitive is contained on one line of the file. The first character of the line specifies the type of the primitive. All units are nominally 0.002 inches.

LINE PRIMITIVE

A first character of L identifies a line primitive. The L is followed by six integers separated by spaces. The first four are the line's endpoint coordinates, in order, x1, y1, x2, y2; the line runs from (x1,y1) to (x2,y2).

Next is the line pattern number, an integer between 1 and 6. These correspond to the 6 patterns available in GED. If no pattern is given, 1 (solid line) is assumed.

The last integer represents the line type. The line type describes both the color and thickness of the line. When the integer is converted to a binary value, bit 0 (zero) defines the thickness (0 = thin), and the seven most significant bits define the color.

The format of the line primitive is:

L *x1 y1 x2 y2* [*pattern line_type*]]

ARC PRIMITIVE

The arc primitive is identified by an A as the first character. It is followed by 5 numbers separated by spaces. These are, in order, X,Y, RADIUS, START_ANGLE, and STOP_ANGLE. The angles, which are in degrees, are in floating point. The rest of the numbers are integers. X and Y are the coordinates of the center of the circle. The angles are measured counter-clockwise from the X axis.

The format of the arc primitive is:

A *x y radius start_angle stop_angle*

TEXT STRING PRIMITIVE

A text string primitive is identified by a ''T'' as the first character in the line. Each text string primitve consists of the following lines; each line is terminated by a line feed character.

T X Y *angle sland size overbar inverse_video*
justification font
string

The individual parameters within each line are:

X,Y	Origin point of text string
angle	0, 1, 2, 3 (for 0, 90, 180, and 270 degrees, respectively)
slant	Not implemented
overbar	Not implemented

inverse_video	Not implemented
justification	0 = left, 2 = right
font	Not implemented
string	The text string (not quoted)

For an example of how to convert the Valid Vector Plot Format to the HPGL (Hewlett Packard Graphics Language) format for display on an HP pen plotter, see the source in /u0/editor/lib/hpfilter.pas.

APPENDIX B
HARDCOPY FONTS

The following fonts are supported; the font name in parentheses is the *font* argument for the SET command's FONT option (**SET FONT** *font*).

- Vector font (VECTOR_FONT)
- Valid Font (VALID_FONT)
- Milspec Font (MILSPEC_FONT)
- Gothic Font (GOTHIC_FONT)
- Cursive Font (CURSIVE_FONT)
- Greek Font (GREEK_FONT)
- Symbol Font (SYMBOL_FONT)

To use these fonts on a drawing, issue the SET command to specify the required font and then HARDCOPY your drawing. One font style can be active at one time, and the active font affects all drawings plotted while it is active.

Figure B-1. Vector Font (Default)

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
-	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	C	D
69	70	71	72	73	74	75	76	77	78	79	80
E	F	G	H	I	J	K	L	M	N	O	P
81	82	83	84	85	86	87	88	89	90	91	92
Q	R	S	T	U	V	W	X	Y	Z	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	^	_	`	a	b	c	d	e	f	g	h
105	106	107	108	109	110	111	112	113	114	115	116
i	j	k	l	m	n	o	p	q	r	s	t
117	118	119	120	121	122	123	124	125	126		
u	v	w	x	y	z	{	¦	}	~		

Figure B-2. Vector Font ASCII Codes

! ″ # $ % & ´ ( ) * + , – . / 0 1 2 3 4 5 6 7 8

9 : ; < = > ? @ A B C D E F G H I J K L M N O P

Q R S T U V W X Y Z [\ ] ↑ _ ` a b c d e f g h

i j k l m n o p q r s t u v w x y z { | } ~

Figure B-3. Valid Font

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
-	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	C	D
69	70	71	72	73	74	75	76	77	78	79	80
E	F	G	H	I	J	K	L	M	N	O	P
81	82	83	84	85	86	87	88	89	90	91	92
Q	R	S	T	U	V	W	X	Y	Z	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	↑	_	`	a	b	c	d	e	f	g	h
105	106	107	108	109	110	111	112	113	114	115	116
i	j	k	l	m	n	o	p	q	r	s	t
117	118	119	120	121	122	123	124	125	126		
u	v	w	x	y	z	{	|	}	~		

Figure B-4. Valid Font ASCII Codes

! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8

9 : ; < = > ? @ A B C D E F G H I J K L M N O P

Q R S T U V W X Y Z [\ ] ↑ _ ` a b c d e f g h

i j k l m n o p q r s t u v w x y z { | } ~

Figure B-5. Milspec Font

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
–	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	C	D
69	70	71	72	73	74	75	76	77	78	79	80
E	F	G	H	I	J	K	L	M	N	O	P
81	82	83	84	85	86	87	88	89	90	91	92
Q	R	S	T	U	V	W	X	Y	Z	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	↑	_	`	a	b	c	d	e	f	g	h
105	106	107	108	109	110	111	112	113	114	115	116
i	j	k	l	m	n	o	p	q	r	s	t
117	118	119	120	121	122	123	124	125	126		
u	v	w	x	y	z	{	|	}	~		

Figure B-6. Milspec Font ASCII Codes

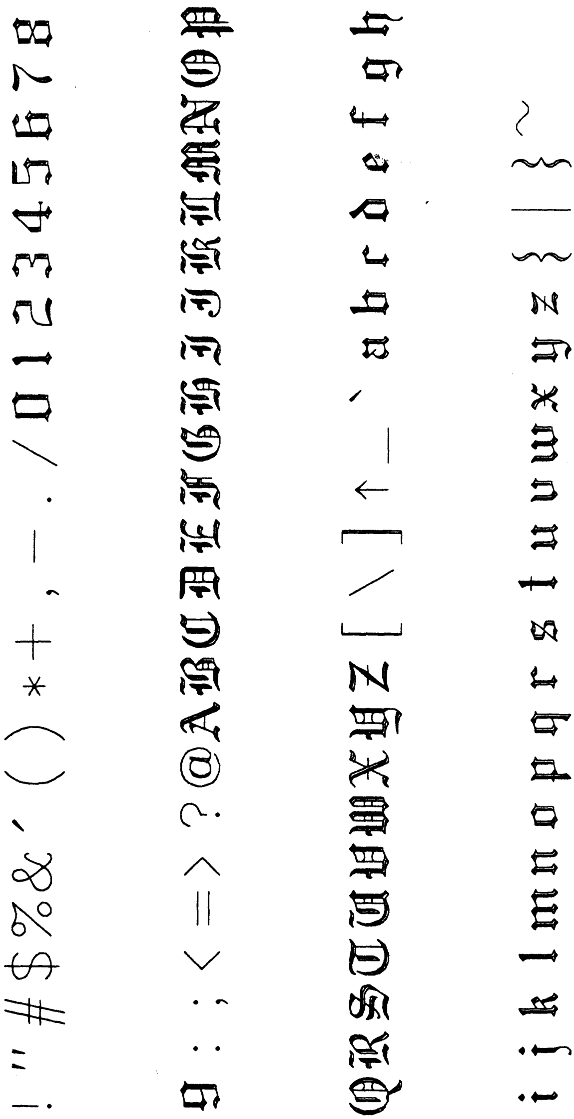


Figure B-7. Gothic Font

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
–	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	C	D
69	70	71	72	73	74	75	76	77	78	79	80
E	F	G	H	I	J	K	L	M	N	O	P
81	82	83	84	85	86	87	88	89	90	91	92
Q	R	S	T	U	V	W	X	Y	Z	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	↑	_	`	a	b	c	d	e	f	g	h
105	106	107	108	109	110	111	112	113	114	115	116
i	j	k	l	m	n	o	p	q	r	s	t
117	118	119	120	121	122	123	124	125	126		
u	v	w	x	y	z	{	|	}	~		

Figure B-8. Gothic Font ASCII Codes

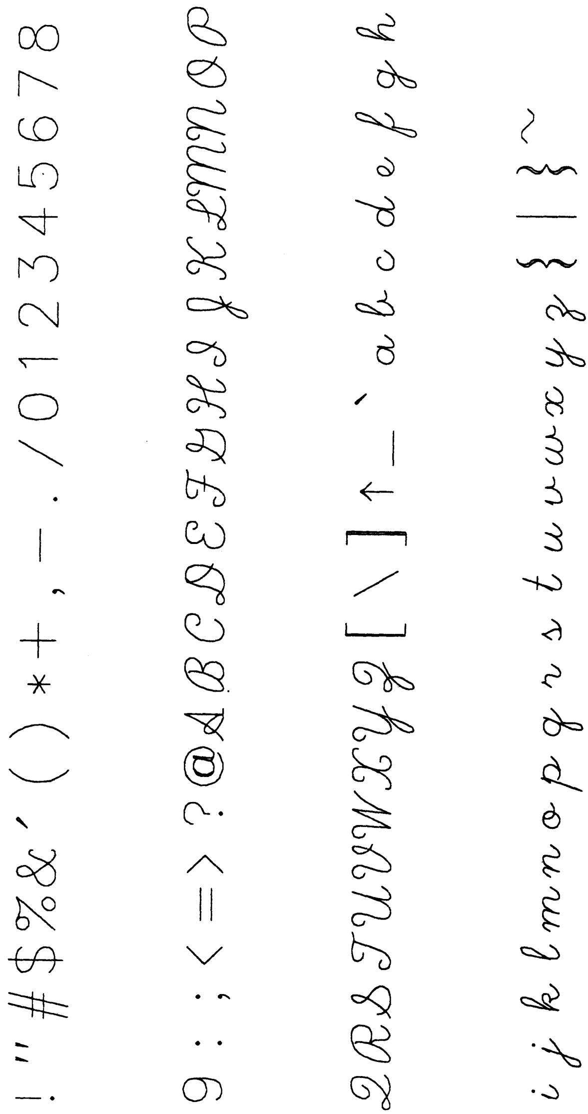


Figure B-9. Cursive Font

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
-	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	C	D
69	70	71	72	73	74	75	76	77	78	79	80
E	F	G	H	I	J	K	L	M	N	O	P
81	82	83	84	85	86	87	88	89	90	91	92
Q	R	S	T	U	V	W	X	Y	Z	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	↑	_	`	a	b	c	d	e	f	g	h
105	106	107	108	109	110	111	112	113	114	115	116
i	j	k	l	m	n	o	p	q	r	s	t
117	118	119	120	121	122	123	124	125	126		
u	v	w	x	y	z	{	|	}	~		

Figure B-10. Cursive Font ASCII Codes

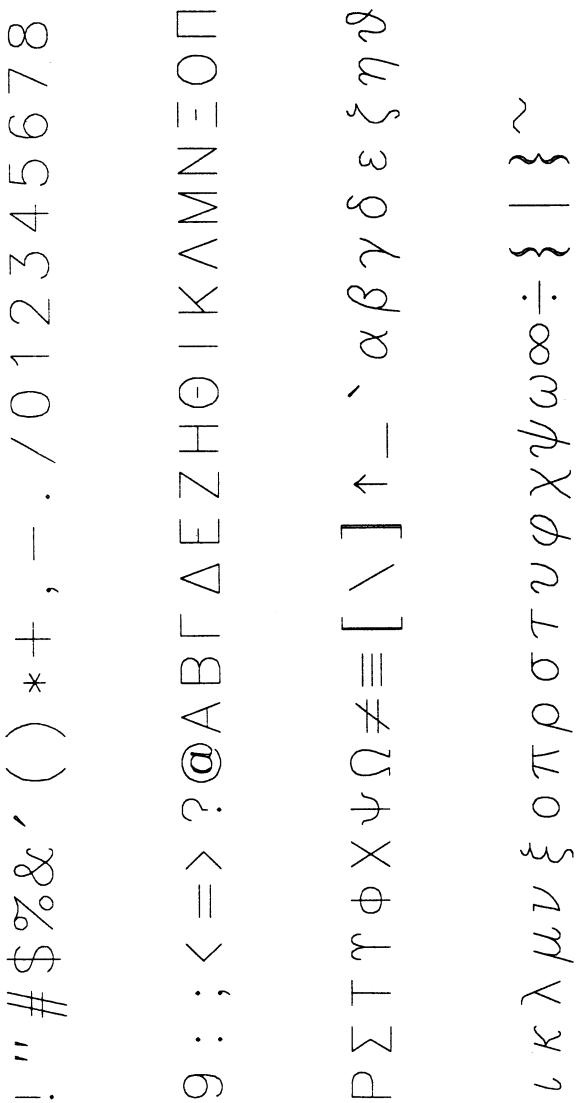


Figure B-11. Greek Font

33	34	35	36	37	38	39	40	41	42	43	44
!	"	#	$	%	&	'	(	)	*	+	,
45	46	47	48	49	50	51	52	53	54	55	56
-	.	/	0	1	2	3	4	5	6	7	8
57	58	59	60	61	62	63	64	65	66	67	68
9	:	;	<	=	>	?	@	A	B	Γ	Δ
69	70	71	72	73	74	75	76	77	78	79	80
E	Z	H	Θ	I	K	Λ	M	N	Ξ	O	Π
81	82	83	84	85	86	87	88	89	90	91	92
P	Σ	T	Υ	Φ	X	Ψ	Ω	≠	≡	[	\
93	94	95	96	97	98	99	100	101	102	103	104
]	↑	_	`	α	β	γ	δ	ε	ζ	η	ϑ
105	106	107	108	109	110	111	112	113	114	115	116
ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ
117	118	119	120	121	122	123	124	125	126		
φ	χ	ψ	ω	∞	÷	{	|	}	~		

Figure B-12. Greek Font ASCII Codes

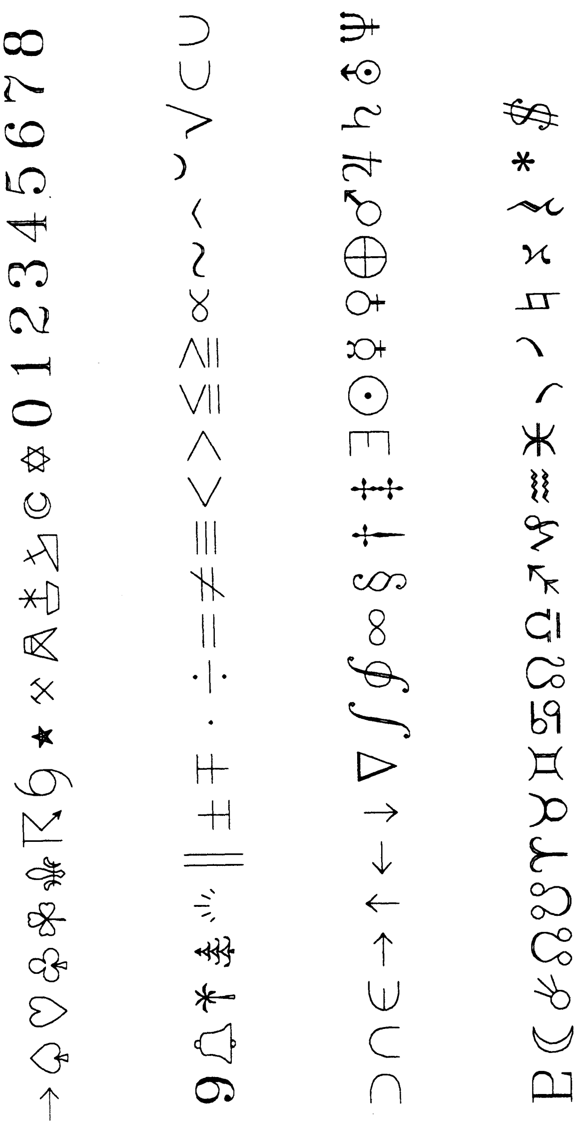


Figure B-13. Symbol Font

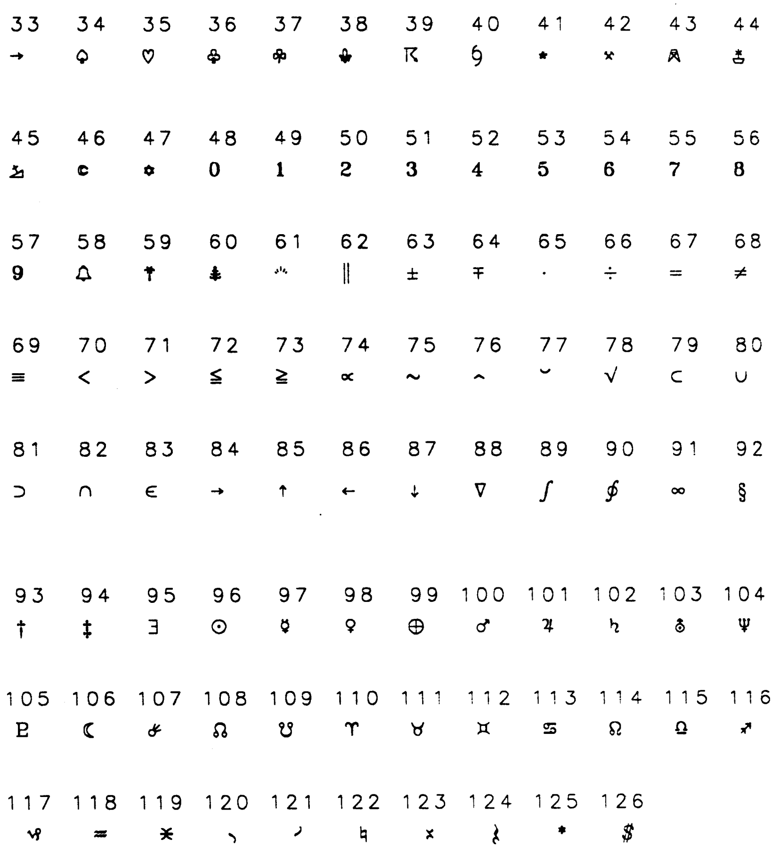


Figure B-14. Symbol Font ASCII Codes

INDEX

SYSTEM UTILITIES
REFERENCE MANUAL

10 March 1986

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371 9004

TABLE OF CONTENTS

Unix to Unix System Copy

SECTION 1
NETWORK COMMUNICATIONS

1.1 INTRODUCTION

This manual describes the various utilities that support host-to-host communications. Throughout this manual, references to UNIX commands and utilities are in **bold-face** type with the chapter of the corresponding UNIX man page from the *UNIX System V User Reference Manual* and the *UNIX System V Administrator Reference Manual* enclosed in parentheses.

1.2 NETWORK TYPES

There are a number of utilities that support communications between a local PC-AT host and a remote host. The utility best suited depends on the networking capabilities of the remote host. In terms of inter-host communications, the following three classifications or "types" of host communications are supported by the local PC-AT client host.

- **Internet Homogeneous Hosts**
  Internet homogeneous hosts are UNIX 4.2 BSD based and use the TCP/IP (Transmission Control Protocol/Internet Protocol) layers of the ARPANET protocol. This type of networked system uses the 4.2 UNIX utilities for remote operations and remote file transfer.

- **Internet Heterogeneous Hosts**
  Internet heterogeneous hosts use the TCP/IP layers of the ARPANET protocol and the "ftp" (file transfer protocol) and "telnet" utilities to provide host-to-host file transfer and remote login operations with hosts that are not necessarily UNIX-based.

- **Asynchronous UNIX-to-UNIX System Copy**
  UNIX system to UNIX system copy uses the **uucp** network to exchange information between UNIX systems over permanent or dial-up connections.

SECTION 2
INTERNET HOMOGENEOUS HOSTS

Internet homogeneous hosts are UNIX 4.2 BSD based and use the TCP/IP layers of the ARPANET protocol to effect remote file transfers and remote shell operations over the network.

2.1 REMOTE FILE TRANSFERS

The remote copy utility **rcp**(1) is used to transfer files between hosts running the UNIX 4.2 BSD or similar operating system. The syntax for remote copy is:

rcp *hostname*:*/pathname/filename /pathname/filename*

or

rcp */pathname/filename hostname*:*/pathname/filename*

For example, to copy the file /u0/tmp/newdata from remote host sys3 to /u0/kristin/mydata on the local host, the following command line is entered:

rcp sys3:/u0/tmp/newdata /u0/kristin/mydata

Similarly, to copy the same file from the local host to remote host sys3, the following command line is entered:

rcp /u0/kristin/mydata sys3:/u0/tmp/newdata

Since **rcp** is a remote shell operation, the user must have a password account on the remote host.

2.2 REMOTE OPERATIONS

In addition to the remote copy (**rcp**) command, the following remote operations are supported.

REMOTE SHELL (rsh)

The **rsh**(1) command connects to the specified host and executes the specified command. The syntax for the **rsh** command is

rsh *hostname command*

where *hostname* is the name of the remote host and *command* is the command to be executed. The command specified is executed on the remote host with all command dialog displayed on the local host. The **rsh** command is terminated following execution of the remote command.

The **rsh** command passes the user's name to the remote host (since the remote host uses the user id and group id in its own /etc/password file, the user names must be identical; passwords are not verified and may differ).

The command **rsh** *hostname* with no other arguments executes an rlogin command. Shell metacharacters

< > >> | { } [ ] ;

are interpreted by the local host. In order to interpret metacharacters at the remote host, they must be enclosed in quotes. Accordingly, the command:

rsh *hostname* **cat** *file1* >> *file2*

appends *file1* on the remote host to file *file2* on the local host. To append *file1* on the remote host to *file2* on the remote host, use the command:

rsh *hostname* **cat** *file1* ">>" *file2*

Interactive commands (e.g., mail, vi, ex, ed), interactive shell scripts, or user programs cannot be run with **rsh**; use **rlogin.**

Stop signals (e.g., control-Z) stop the local rsh process only.

REMOTE LOGIN (rlogin)

The **rlogin**(1) command connects the local host to the remote host as if the local host were a terminal on the remote host. The command syntax is

rlogin *hostname*

where *hostname* is the name of the remote host.

The remote terminal type is the same as the local terminal type (as given in the environment TERM variable). All echoing occurs at the remote host so that (except for delay) the rlogin is transparent. Flow control via CTRL-S and CTRL-Q and the flushing of input and output on interrupt are handled properly. To terminate the connection with the remote host, either enter

~. (tilde - period)

or log off of the remote host. Similarly, to suspend an rlogin session, enter:

~^Z (tilde - control-Z)

The **~^Z** command, like a control-Z in the UNIX C shell, stops the current job (i.e., rlogin) and returns to the local shell prompt. Once stopped, rlogin can be placed in the background by entering **bg** and can be placed in the foreground (i.e., resumed) by entering **fg**.

NOTE

Since background/foreground control only is supported by the C shell, the **~^Z** command cannot be used when the local host is operating in the Bourne shell.

REMOTE WHO (rwho)

The **rwho**(1) command produces output similar to **who**(1), but for all hosts on the local network

REMOTE UPTIME (ruptime)

The **ruptime**(1) command displays status of each host on the local network. The command displays the status (up or down), the length of time each host has been up, the number of users logged on to the system, and the average number of jobs in the run queue over the last 1, 5, and 15 minutes.

2.3 REMOTE HOST ACCESS

In order to use **rcp**, **rsh**, **rlogin**, **rwho**, and **ruptime**, the user must have a password account on the remote host (see ''File Security'' later in this section). In addition, the name of the user's local host must also be present in the /etc/hosts.equiv file on the remote host (see the *Valid Guide to Operations* manual).

If the user has an account on the remote host with an identical user name, a password is not requested; if the user does not have an account with the same user name, a login and password will be requested as with **login**(1).

2.4 NETWORK STATUS

A host cannot send requests to a host that is not enabled or to a host that is not connected to the network. To determine the reachable hosts or ''nodes'' on the network, use the **ruptime** command.

NOTE

> In order to use the **ruptime** command, the command daemon (**ruptimed**(1M) must be running on all of the remote hosts on the network as well as the local host. By default, the **ruptimed** daemon is **not** running and must be started by the system administrator at each site (see the *valid Guide to Operations*).

2.5 FILE SECURITY

The remote facilities assume that the password files (*/etc/passwd* and */etc/group*) on both the client and server hosts contain identical information. Specifically, users must have the same user and group id numbers present in each host on which they are granted file access. When making a remote call, the local user's user and group id numbers are transmitted; the remote server accepts the user id without question. To ensure a minimal level of security, each host on the network should have an identical password file. To give a user access to a remote host, the system administrator on the remote host must run the **mkusr**(1M) command to install that user on the remote host. For a given user, **mkusr** must be run on every host on which that user has access. Note that by default, a system administrator does not have superuser privilege on remote hosts.

SECTION 3
INTERNET HETEROGENEOUS HOSTS

Internet heterogeneous hosts use the TCP/IP layers of the ARPANET protocol and the **ftp**(1) and **telnet**(1) utilities to provide host-to-host file transfer and remote login operations, respectively, on non-UNIX based hosts. Note that a password account and password are always required for user authentication.

3.1 FILE TRANSFER PROTOCOL

The file transfer protocol or ''ftp'' is a file transfer utility that can be used between a SCALDsystem host and any remote host that supports the ARPANET standard TCP/IP protocol.

The primary purpose of the **ftp** utility is to provide a common file transfer mechanism between heterogeneous hosts on a network (e.g., a SCALDsystem user can transfer a file to a VAX/VMS host running TWG WIN/VX).

The sections that follow describe the operation and individual commands available with the ftp utility on a SCALDsystem; it is not required for the user to have any knowledge of the Internet protocols or the version of the remote ftp server on the target system.

3.2 ENTERING THE FTP SHELL

To start an ftp session, enter

ftp

in response to the UNIX shell prompt (%).

When the RETURN key is pressed, the following prompt is displayed:

ftp>

This prompt is the ftp command shell prompt. A complete list of the available ftp commands can be displayed by entering either:

ftp> **help**
or
ftp> **?**

A one line summary of any command is displayed by entering either

ftp> **help** *commandname*
or
ftp> **?** *commandname*

where *commandname* is the name of the command.

3.3 CONNECTING TO A REMOTE HOST

To connect to a remote host, enter

ftp> **open** *hostname*

where *hostname* is the name or the Internet address of the remote host (the Internet address is specified in the /etc/hosts file).

NOTE

When initiating an ftp session (i.e., entering **ftp** in response to the UNIX shell prompt), if *hostname* is included as a command-line argument (e.g., % **ftp** *hostname*), the specified host is automatically opened.

When *hostname* is entered, the ftp utility attempts to connect to the remote host. If the connection is successful (i.e., if *hostname* appears in the local /etc/hosts file and if the remote host can be accessed), the following typical messages are displayed (the actual messages are remote-host dependent):

> **Connected to** *hostname*
> **220** *hostname* **FTP server (Version** *x.x date*) **ready**
> **Name (***hostname:username***):**

These messages originate from the ftp server process on the remote host; the last message is a prompt for a user login name entry. After the user's name is entered, a password is requested (if required) to verify permission for access to the remote host. When access is granted, a confirmation message such as

> **230 User logged in, default directory** *directorypathname*

is displayed. Conversely, if an invalid user name or password is entered, a message similar to

> **530 Login failed (bad Username or Password), give new USER and PASS**
> **Login failed**

is displayed. If an error is made during the username or password entry, the ftp **user** command can be used to repeat the login sequence.

Note that the numbers preceding the messages describe the state of the ftp deamon and can be disregarded by the user (the numbers are defined in the document *File Transfer Protocol*†).

† Postel, J., "File Transfer Protocol", *Internet Protocol Transition Workbook*, SRI Network Information Center, RFC 765, March 1982.

3.4 TRANSFERRING FILES

Once a user has successfully logged in on the remote host, any file (with read permission) can be read from the remote host to the local host or any file on the local host can be written to the remote host. Note that with ftp, the file naming convention of the remote file system is specific to the remote host (see "Filename Syntax" in the next section). Two ftp commands (**ls** and **dir**) can be used to obtain a listing of the files in the current directory on the remote host; **ls** causes a simple listing to be displayed while **dir** typically causes a more detailed listing to be displayed (equivalent to the UNIX **ls**(1) command with a -lg option). Note that the actual format of the data returned is remote host specific.

When ftp is invoked, all file transfers are assumed to be ASCII text files; transfer of binary (executable) files must be expressly stated (with the binary command).

To transfer files between the local and remote hosts, the ftp **get** and **put** commands are used; **get** transfers (reads) a file from the remote host, and **put** transfers (writes) a file from the local host to the remote host. The syntax for the **get** and **put** commands is:

> **get** *remote-filename* [*local-filename*]
>
> **put** *local-filename* [*remote-filename*]

In the **get** command, *local-filename* is optional, and in the **put** command, *remote-filename* is optional (if either is omitted, the name of the destination file is the same as the source file). When a message like

> **226 Closing data connection; requested file action successful**

is displayed, the file transfer is complete. Note again that the actual message displayed is remote host dependent.

3.5 FILENAME SYNTAX

The filename syntax is system dependent. An example of a directory/filename structure for a VAX running under the VMS operating system would be:

drc3:[garyl.manual]ch12.txt

In this example, "drc3:" identifies the disk, "manual" is a directory in the user account "garyl," and "ch12.txt" is the name of the file. To avoid transferring a file with an incompatible syntax when full pathnames are used (brackets in a filename are awkward in UNIX), the optional destination name must be included. To transfer the file in the example (ch12.txt), the following ftp command would be used:

ftp> **get drc3:[garyl.manual]ch12.txt ch12.txt**

3.6 MULTIPLE FILE TRANSFERS

The **get** and **put** commands transfer single files; to transfer multiple files, the **mget** and **mput** commands are used. The syntax for these commands is:

mget *remote-filename1 remote-filename2 ...*

and

mput *local-filename1 local-filename2 ...*

Note that with the **mget** and **mput** commands, the optional destination filenames cannot be specified and that all files are transferred to the current local (**mget**) or remote (**mput**) directory. When the list of files to be transferred does not fit on a line, the line can be continued by entering a backslash (\) and a RETURN.

Metacharacters are processed by default by the **mput** command. The metacharacters recognized by **mput** are:

*	Match all (or zero) characters
?	Match any single (or zero) character
[]	Match any of the character patterns between the brackets
~	Substitute home directory

Note that metacharacter processing is supported by some remote hosts (metacharacters may be used with the **mget** and **mdelete** commands). To disable metacharacter processing, use the ftp **glob** command.

3.7 CHANGING DIRECTORIES

Since the syntax of the **mget** and **mput** commands does not accept a target filename, the ftp **cd** command can be used to change the remote directory. The syntax of remote directory pathnames is server dependent (some servers accept UNIX notation). Refer to the users manual for the server. To change the local directory, use the ftp **lcd** command.

3.8 DELETING REMOTE FILES

Files can be deleted from the remote server using either the **delete** (single file) or **mdelete** (multiple files) command. The syntax of these commands is:

delete *remote-filename*
or
mdelete *remote-filename1 remote-filename2 ...*

Note that metacharacter processing is supported by the **mdelete** command.

3.9 DISCONNECTING FROM THE REMOTE HOST

To close a connection to a remote host without exiting ftp (e.g., to connect to another remote host), enter:

ftp> **close**

Note that entering a control-C (^C) during a transfer will abort the transfer and return to the ftp command prompt level (ftp>).

Once the current connection is closed, a connection with another host can be initiated (**open** *remote-host*). To close a connection and terminate the ftp program, enter either:

ftp> **bye**
or
ftp> **quit**

3.10 COMMUNICATION WITH THE SHELL

Shell commands can be executed locally if the command is preceded by an exclamation point ('!'). As an example, to create a new directory on the local host from within an ftp session, enter:

ftp> **!mkdir** *dirname*

Following command execution, the ftp command shell is automatically reentered.

PREVIEWING A FILE

To preview a remote file prior to transfer, a dash ('-') is used in place of *local-file* in the **get** or **recv** command to cause the file to be displayed on standard output. For example

ftp> **get remfile -**

displays the contents of remote file ''remfile'' on the screen of the local host (equivalent to a remote ''cat'').

Note that a dash also can be used with the **put** command to send keyboard entry from the local host to the specified file on the remote host. For example, entering the command

ftp> **put - remfile**

followed by keyboard-entered text terminated with a control-D writes the text input to the file "remfile" on the remote host.

SHELL COMMAND EXECUTION

If the first character of the file name is '|', the rest of the file name is considered to be a shell command. For example, the command

ftp> **put |ls mydir**

writes a listing of the current local directory to the file "mydir" on the remote host. Another example of shell command execution would be:

ftp> **ls . |more**

This command "mores" the directory listing when the number of files exceeds the number of screen lines. If spaces are required to delimit the command from its arguments, quotes must be put around the shell command, such as

ftp> **dir . "|sort +4nr"**

3.11 FTP COMMANDS SUPPORTED BY THE REMOTE HOST

The command **remotehelp** returns a list of the commands supported by the remote host.

Note that all of the commands supported by the remote host may not be supported by the local SCALDsystem. Specifically, the following ftp commands are **NOT** supported:

REIN	PASV	MLFL	MAIL	MSND	MSOM
MSAM	MRSQ	MRCP	REST	ABOR	SITE

The "SITE" command may be necessary for file transfers on some systems (file structure and type are some of the examples on which the "SITE" command may have an effect). Check the remote host's server manual for a more complete description of command operation. To send a "SITE" command to the remote host, type:

> ftp> **quote site** *arg1 arg2 ...*

The **quote** command sends its arguments verbatim to the server. There must be only one return reply from the server to keep communication between the client and server in a known state.

3.12 COMMAND SUMMARY

The following table summarizes the ftp command set recognized by a SCALDsystem UNIX-based host. The portion of the command in **bold-face** type indicates the smallest non-ambiguous abbreviation that can be entered). These commands are also described in **ftp**(1) of the *UNIX System V User Reference Manual.* Note that some of the commands are UNIX specific and are not part of the ARPANET protocol (e.g., **pwd**, **mkdir**, and **rmdir**).

Table 3-1. FTP Command Summary

Command	Description
! *UNIX-command*	Executes a UNIX command on local host (following command execution, the ftp shell is automatically reentered).
ap*pend local-file* [*remote-file*]	Appends *local-file* to a file on the remote host; if *remote-file* is not specified, *local-file* is used as the name of the file on the remote host.
ascii	Sets file transfer type to ASCII (default).
bell	Toggles bell (^G) on and off. When on, bell is sounded after each file transferred (default OFF).
binary	Sets file transfer type to binary.
bye	Closes connection and terminates ftp session; see also **quit**.
cd *remote-dirname*	Changes directory on remote host to *remote-dirname*. Note that **cd** (with no argument) prompts for a directory name and does not select the user's home directory on the remote host.
close	Terminates current connection to remote host and returns to ftp command shell to allow a new connection to be established.
delete *remote-file*	Deletes named file from remote host.

Table 3-1. FTP Command Summary (Con't)

Command	Description
debug	Enables/disables debugging mode (default is disabled).
dir [*dirname*]	Lists contents of current remote directory in long format (if available); if *dirname* is specified, lists contents of named directory.
form	Sets file transfer format (only non-print format supported).
get *remote-file* [*local-file*]	Transfers (reads) named file from remote host to local host; if *local-file* is specified, file transferred is named *local-file* (see also **recv**).
glob	Toggles metacharacters processing in local filename for **mput**, **mget**, and **mdelete** commands (default is metacharacter processing ON).
hash	Toggles display of a "#" character for every 512 bytes transferred (default is # display OFF).
help [*command-name*]	Lists commands for which help is available; if *command-name* is entered, displays help text for command (see also **?**).
lcd [*dirname*]	Displays name of current directory on local host; if *dirname* is specified, changes working directory to *dirname*. Note that the **!cd** command is ineffective in the ftp shell, and that **lcd** must be used to change the working directory on the local host.

Table 3-1. FTP Command Summary (Con't)

Command	Description
ls [*dirname*]	Lists contents of current directory on remote host; if *dirname* is specified, lists contents of named directory on remote host (see also **dir**).
mdelete *remote-file1 remote-file2 ...*	Deletes specified files from remote host. If prompting is enabled (the default), user must confirm each file to be removed.
mdir *remote-directory* [*local-file*]	Copies contents of remote directory to file *local-file* on local host. If *local-file* is not specified, the name of the remote directory is used for the name of the local file.
mget *remote-file1 remote-file2 ...*	Transfers (reads) named files from remote host into current local directory using remote filenames.
mkdir *dirname*	Creates directory *dirname* on remote host. Note that this command may not be supported by all remote hosts.
mls *remote-directory* [*local-file*]	Copies file listing of remote directory to file *local-file* on local host. If *local-file* is not specified, the name of the local directory is used for the name of the local file (see also **mdir**).

Table 3-1. FTP Command Summary (Con't)

Command	Description
mode	Sets file transfer mode (only stream mode is supported).
mput *local-file1 local-file2 ...*	Transfers (writes) named files from local host into current remote directory using local filenames.
open *remote-host*	Connects to *remote-host* for communications.
prompt	Toggles interactive confirmation for ftp commands involving multiple file transfers (i.e., **mdelete**, **mget**, and **mput**). Default is confirmation ON.
put *local-file* [*remote-file*]	Transfers (writes) named local file to remote host; if *remote-file* is specified, file transferred is named *remote-file* (see also **send**).
pwd	Prints name of current (working) directory on remote host. Note that this command may not be supported by all remote hosts.
quit	Terminates ftp session and exits ftp (see also **bye**).
quote *command-name*	Sends ftp command to remote host for execution on remote host. Note that this command should not be used by users who are unfamiliar with ftp implementation or with the remote host's operating system.

Table 3-1. FTP Command Summary (Con't)

Command	Description
recv *remote-file* [*local-file*]	Transfers (reads) named file from remote host to local host; if *local-file* is specified, file transferred is named *local-file* (equivalent to **get**).
remotehelp [*command-name*]	Displays list of commands supported by remote host on which help is available; if *command-name* is specified, displays help text for named command (refer to **ftpd**(1M) or Internet Protocol document for description).
rename *old-filename new-filename*	Changes *old-filename* to *new-filename* on remote host.
rmdir *dirname*	Removes named directory from remote host. Note that this command may not be supported on all remote hosts.
send *local-file* [*remote-file*]	Transfers (writes) named local-file to remote host; if *remote-file* is specified, file transferred is named *remote-file*.
sendport	Enables/disables port commands (default enabled).
status	Displays current ftp status and settings.
struct	Sets file transfer structure (only file structures supported).

Table 3-1. FTP Command Summary (Con't)

Command	Description
tenex	Sets tenex file transfer type.
type	Reports file transfer type (ASCII, binary, or tenex); see also **status**.
user	Specifies user on remote host. Note that some remote hosts only allow this command to be used once per connection.
verbose	Enables/disables verbose mode (i.e., number of bytes transferred and transfer time reported); default is enabled.
? [*command-name*]	Displays list of local ftp commands on which help is available; if *command-name* is specified, displays help text for named command (see also **help**).

SECTION 4
TELNET

The **telnet**(1) program provides heterogeneous hosts on a network a common interface to run a login session. With **telnet**, a user on a local system running UNIX can log on to a VAX running VMS (e.g., VAX/VMS with TWG WIN/VX). **Telnet** can be thought of as functionally equivalent to the UNIX utility **rlogin**(1). To use the telnet utility on the PC-AT, knowledge of the Internet protocols is not required although some level of expertise about the target system's operating system and screen display is necessary since the UNIX terminal characteristics (TERM environment variable and window parameters) are not "passed" to the remote host.

4.1 GETTING STARTED

To initiate a telnet session, enter the command

telnet

in response to the UNIX shell prompt (%). When the RETURN key is pressed, the following prompt is displayed:

telnet>

This prompt is the telnet command shell prompt. A complete list and brief description of the available telnet commands can be displayed by entering:

telnet> **?**

4.2 CONNECTING TO A REMOTE HOST

To connect to a remote host, enter

telnet> **open** *hostname*

where *hostname* is the name of the remote host.

NOTE

> When initiating a telnet session (i.e., entering **telnet** in response to the UNIX shell prompt), if *hostname* is included as a command-line argument (e.g., % **telnet** *hostname*), the specified host is automatically opened.

When *hostname* is entered, the telnet program attempts to connect to the remote host. If the connection is successful (i.e., if *hostname* appears in the local /etc/hosts file and the remote host can be accessed), the following typical messages are displayed:

Connected to *hostname*
Escape character is '^]'
Hostname **login:**

The last message is a prompt for a user login entry on the remote host as if the local host were a terminal on the remote host.

4.3 RETURNING TO THE TELNET COMMAND SHELL

Note that the default escape character is '^]' (control-]). Entering this character while in a telnet session returns the user to the telnet command shell (telnet>).

Note that if the remote host interprets '^]' to be special character, the escape character must be changed with the telnet **escape** command. For example, to change the escape character to '^x', enter:

telnet> **escape**

Telnet will prompt for a new escape character. Type in control-x (i.e., hold the CONTROL key down and press the 'x' key). Telnet will set the new escape character and return the user to the login session.

4.4 DISCONNECTING FROM REMOTE HOST

To disconnect from the remote host without exiting telnet, first enter the escape character (to return to the telnet command shell) and then enter:

telnet> **close**

Once the current connection is closed, a connection with another host can be initiated (**open** *hostname*). To close a connection and terminate the telnet session, either enter the escape character followed by:

telnet> **quit**

or log off of the remote host (logging off of the remote host automatically quits the telnet session).

4.5 SUSPENDING A SESSION

To suspend a telnet session (i.e., to maintain the connection while performing local operations), return to the telnet shell prompt (enter the escape character) and enter:

telnet> **z**

The **z** command, like a control-z in the UNIX C shell, places the current job (i.e., telnet) in the background and returns to the local shell prompt. Once the job (telnet) is placed in the background, it can be and returned to the foreground by entering **fg**.

NOTE

Since background/foreground control only is effective in the C shell, the **z** command should not be used when the local host is operating in the Bourne shell.

4.6 CHECKING STATUS

The status of the telnet session is verified with the **status** command. Entering

telnet> **status**

causes the following information to be displayed:

Connected to *hostname*
Escape character is *x*

4.7 TELNET COMMAND SUMMARY

The following table summarizes the telnet command set. The portion of the command in **bold-face** type indicates the smallest non-ambiguous abbreviation that can be entered. These commands are also described in **telnet**(1) of the *UNIX System V User Reference Manual.*

Table 4-1. Telnet Command Summary

Command	Description
close	Closes current telnet connection and returns to telnet shell command prompt.
crmod	Enables/disables the mapping of received carriage returns (default is no mapping).
debug	Enables/disables debugging mode (default is debug mode off).
escape	Sets escape character (default escape character is control-] (^]).
flow	Enables/disables flow control (usually XON/XOFF; i.e., control-Q/control-S). Default is flow characters disabled.
open *hostname*	Connects to remote host *hostname*.
options	Enables/disables viewing of options processing (default is viewing disabled).
quit	Closes current telnet connection and terminates telnet session.
status	Displays current host connection and escape character.
z	Suspends current telnet session and returns to local command shell.
?	Displays list of available telnet commands and brief description.

SECTION 5
UNIX to UNIX SYSTEM COPY

The **uucp**(1) network is a network of UNIX systems that allows file transfer and remote execution to take place on a network of UNIX systems. The extent of the network is a function of both the interconnection hardware and the controlloing network software. Access to the network is tightly controlled by the software to preserve the integrity of all users on the network (a user cannot use the **uucp** facility to send files to a system that is not part of the network).

5.1 NETWORK HARDWARE

The three most common methods of connecting systems are:

1. Directly connecting two UNIX systems by cross-coupling (via a null modem) two of the RS-232 ports. This method is limited to short distances (typically several hundred feet) and is usually run at high speed (9600 baud).

2. Using a modem (a private line or minimum-distance modem) to directly connect two systems over a private line through data sets.

3. Connecting a system to another system through a modem, an automatic calling unit (ACU), and the Direct Distance Dialing (DDD) network.

5.2 NETWORK SOFTWARE

The **uucp** network is a batch network (i.e., when a request is made, the request is spooled for subsequent transmission by a daemon). Jobs submitted to the network are assigned a sequence number for transmission. Jobs are represented by a file (or files) in the common spool directory */usr/spool/uucp*. When the file transfer daemon (**uucico**) is started to transmit a job, it selects a system to contact and then transmits all jobs to that system. Before breaking off the conversation, any jobs to be received from the remote system are accepted. **Uucp** may be sending to or receiving from many systems simultaneously; the number of incoming requests is limited only by the number of connections on the system, and the number of outgoing transfers is limited by the number of direct connections or ACUs.

When a job is submitted to the network, an attempt is made immediately to contact the system; only one conversation can exist between the same two systems at any time. Systems that are polled cannot force the immediate transmission of data; jobs are only transmitted when the system is polled by the remote system.

The **uucp** network is persistent in its attempt to contact remote systems in order to complete a transmission. To prevent **uucp** from continually calling systems that are not available, a "hysteresis mechanism" is built into the algorithm that is used to contact other systems. This mechanism forces a minimum fixed delay to occur before another transmission can occur to that system.

In order to allow the transfer of files to a system on which a user does not have a login account, the *public* directory (usually kept in *usr/spool/uucppublic*) is available with general access privilege. When receiving files in the public area, the user should remove them quickly as the administrative portion of **uucp** regularly purges this area.

5.3 PERMISSIONS

Uucp uses the UNIX system password mechanism in conjunction with the system file */usr/lib/uucp/L.sys* and the file system permission file */usr/lib/uucp/USERFILE* to control access between systems. **Uucp** also uses the file */usr/lib/uucp/L.cmds* to restrict **uux** remote command execution. The password file entries for **uucp** (usually, **luucp**, **nuucp**, **uucp**, etc.) allow only the remote systems that know the passwords for these IDs to access the local system. Note that care should be taken in revealing the password for these **uucp** logins since knowing the password allows a system to join the network. The system file */usr/lib/uucp/L.sys* defines the remote systems accessible to the local host. This file contains all of the information required for a local host to access a remote system (i.e., system name, password, login sequence, etc.) and is protected from viewing by ordinary users (permission is usually "600").

In the transfer of files between systems, users should make sure that the destination area is writable by **uucp**. The **uucp** daemons preserve execute permission between systems and assign permission "666" (read/write) to transfer files. The system administrator determines the global access permissions on a machine-by-machine basis. Accordingly, access between systems may be confined by the administrator to only selected areas of the file system.

5.4 USING UUCP

Uucp uses the following syntax to reference files on remote systems

system_name! pathname

where *system_name* is a the name of a system on the network. *Pathname* may include any of the following forms:

1. A fully qualified *pathname* such as:

   lore!/u0/kristin/bigfile

   The *pathname* may also be a directory name as in:

   lore!/u0/kristin/bigdir

2. The login directory on a remote system may be specified by the '~ (tilde) character. The combination ~*user* references the login directory of a user on the remote system. For example

   lore!~kristin/ch1

   is expanded to

lore!/usr/sys/kristin/ch1

if the login directory for user kristin is */usr/sys/kristin.*

3. The *public* area can be referenced by a similar use of the ~/*user* pathname prefix. For example

   lore!~/kristin/newfile

   is expanded to

   lore!/usr/spool/uucp/kristin/newfile

   if */usr/spool/uucp* is used as the spool directory.

4. Pathnames not using any of the combinations or prefixes described above are, by default, prefixed with the current dirctory (or the login directory on the remote system). For example

   lore!yourfile

   is expanded to:

   lore!/usr/you/yourfile

5.5 FORWARDING

Uucp permits files to be passed between systems via intermediate nodes. This type of file transfer uses a variation of the "bang" (!) syntax to describe the path to be taken to reach the desired file. For example, if a user on system 'a' wishes to send a file to system 'e' through nodes 'b,' 'c,' and 'd,' the following command might be used:

uucp yourfile b!c!d!e!~/you/yourfile

Note that the pathname is the path that the file takes to reach node 'e.' Note also that the destination must be specified as the *public* area. Fetching a file from another system via intermediate nodes is done is a similar manner. For example, the command

uucp b!c!d!e!~/you/remotefile localfile

fetches the file *remotefile* from system 'e' and renames the file *localfile* on the local system. Note that the forwarding prefix is the path **from** the local system and is **not** the path from the remote system to the local system. The forwarding feature also can be used with remote execution. For example, the following command (entered on one line)

uux lore!uucp alien!darkstar!/usr/spool /uucppublic/oldfile newfile

sends a request to lore to execute a **uucp** command to transfer the file *oldfile* from darkstar through alien to the file *newfile* on lore.

5.6 TYPES OF TRANSFERS

Uucp offers a flexible command syntax for file transmission. The following sections give examples of various combinations of transfers.

METACHARACTERS

The **uucp** command syntax supports the ***, ?, and [..]** metacharacters for the transmission of multiple files. For example, the command

uucp *.[xy] lore!dir

transfers all files with extensions that end with 'x' or 'y' to the directory *dir* in the user's login directory on lore. Similarly, the command

uucp lore!*.tro dir

fetches all file with the extension *.tro* from the user's login directory to the subdirectory *dir* on the local system.

SWITCHING

The transmission of files can be arranged so that the local system effectively functions as a switch. For example, the command:

uucp alien!fatfile darkstar!bigfile

fetches the file *fatfile* from the user's login directory on alien, renames the file *bigfile*, and writes the file in the login directory on darkstar.

REMOTE EXECUTION

The remote execution facility allows commands to be executed remotely. For example, the following command (entered on one line)

uux "!diff lore!/etc/passwd
darkstar!/ect/passwd !pass.diff"

executes a **diff**(1) command on the password files on lore and darkstar and places the result in the file *pass.diff*.

LOCAL CONTROL

The **uuto** command uses the **uucp** facility to send files while allowing the local system to control file access. For example, the following command sends the file *myfile* from local system lore to user *srguru on remote system darkstar:*

uuto myfile lore!srguru

SPOOLING

To continue to modify a file while a copy is being transmitted across the network, the – **c** option is used. This option forces a copy of the file to be queued (the default for **uucp** is not to queue copies of files). For example, the following command forces the file *modfile* to be copied to the spool directory before it is transmitted:

uucp – c modfile lore! /user/modfile

5.7 NOTIFICATION

The success (or failure) of a transmission is asynchronously reported to users via the **mail**(1) command. The forms of notification are:

1. Notification returned to the requester's system (via the **– m** option). This form is useful when the requesting user is distributing files to other machines. Instead of logging on to the remote machine to read mial, mail is send to the requester when the copy is finished.

2. Notification returned in file specified by the requester. This form is a variation of the **– m** option and forces notification in a file (through the **– m***file* option where *file* is the name of the notification file). For example, the command

   uucp -mans /etc/passwd lore! /dev/null

   sends the file */etc/passwd* to the system lore and places the file in the */dev/null.* The status of the transfer is reported in the file *ans* as:

   uucp job *nnnn* (*queue/time*) (*execution/time*)
   /etc/passwd copy succeeded

3. **Uux**(1) always reports the exit status of a remote execution unless notification is explicitly suppressed (with the **– n** option). Notification can be sent to a different user on the remote system with the **– nuser** option.

A single job ID number can be associated with each command execution so that a job can be terminated or its status verified. The default for the **uucp** and **uux** commands is not to print the job ID number for each job. If the environment variable

JOBNO=ON

is made part of the user's environment and exported, **uucp** and **uun** print the job ID number (the environment variable JOBNO=OFF turns job ID numbers off). To force the printing of job ID numbers without using the environment mechanism, the **– j** option is used. For example, the command

uucp – j /etc/passwd lore! /dev/null

forces the job ID number to be displayed following the command line (e.g., uucp job 237). If the **– j** option is not used, the ID numbers of the jobs (belonging to the user) can be found by using the **uustat**(1) command.

JOB STATUS

The **uustat** command allows the user to check the status of specific jobs that have been queued. The ID number displayed when a job is queued (JOBNO=ON or **– j** option) can be entered as an argument to the **uustat** command to display the status of only that job.

There are several status messages that may be displayed with the **uustat** command; the most common messages are JOB IS QUEUED and JOB COMPLETED. For additional status messages definitions, see the manual page for the **uustat** command.

NETWORK STATUS

The status of the last transfer to each system on the network also can be found with the **uustat** command. For example, the command

uustat – mall

reports the status of the last transfer to all systems known to the local system. A typical output would be

```
lore  12/24-16:45    CONVERSATION SUCCEEDED
alien 12/23-08:30    DIAL FAILED
```

darkstar	12/24-11:17	JOB COMPLETED
xyzzy	12/21-13:32	LOGIN FAILED

where the status indicates the time and state of the last transfer to each system. When sending files to a system that has not been contacted recently, the **uustat** command should be used to determine when the last access occurred (the remote system may be down or out of service).

JOB CONTROL

When job ID numbers are generated for each **uucp** or **uux** command, the following controls can be exercised:

1. **Job Termination.** A job that transfers multiple files from several different systems can be terminated with the **– k** option of the **uustat** command. Note that this option only affects queued files on the local system and that files already transferred cannot be recalled.

2. **Requeuing a Job.** The **uucp** facility periodically purges jobs (typically every 72 hours) to prevent the accumulation of jobs that cannot be transmitted. The **– r** option of the **uucp** command can be used to force the date of a job to be updated with the current date in order to lengthen the interval that **uucp** attempts to transmit the job.

NETWORK NAMES

The **uuname** command is used to identify the names of the systems on the network. Note that this command only displays the system names.

INDEX