Using ValidGED on Your Sun Workstation™

Insert this manual in your ValidGED binder.

January 15, 1989

P/N: 900-00581

Using ValidGED on Your Sun Workstation™

January 15, 1989

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, California 95134
(408) 432-9400 Telex 371 9004
FAX 408 432-9430

MANUAL REVISION HISTORY

Rev	Date	Software Release	Reason for Change
A	1/15/89	GED 9.0	Initial release

Preface

This manual is intended as a supplement to the *ValidGED User's Manual* and *ValidGED Command Reference Manual.* The features it describes are applicable only on the Sun Workstation™.

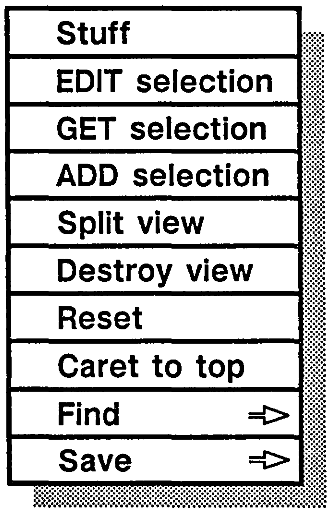
Table of Contents

1 Message Window Popup Menu

This section explains the Message Window Popup Menu.

Menu Operations

On the Sun workstation, a popup menu is available within the message window that provides a few "shortcuts" to some GED operations. The main message window popup menu is shown below.



There are two methods of selection for all of the popup menu operations:

- Single selection
- Multiple selection

Use the right mouse button to perform popup menu operations after either type of text selection.

Single selection uses the left and right mouse buttons:

- Press the left mouse button to start a selection.
- Press the middle mouse button to end a selection.

Multiple selection uses only the left mouse button. The mouse button presses should be in quick succession.

- Press the button once to select one character.
- Press the button twice to select the word under the cursor.
- Press the button three times to select the whole line under the cursor.
- Press the button four times to select the contents of the entire message window.

The SunView function keys **Put** (L6), **Get** (L8), and **Find** (L9) operate in the standard SunView manner within the GED message window.

Stuff

"Stuff" makes a copy of the selected item and inserts the text as GED input. This is useful for repeating commands. Another method of copying and repeating text is to hold down the Shift key and the right mouse button at the same time. This causes the selected text to be "stuffed" immediately, without bringing up the menu.

NOTE: **Stuff** *does not include a* Return. *You must enter the* Return *yourself.*

You can also copy an item from other Sun windows and "stuff" the information into the GED window. You cannot, however, copy items from the GED window and "stuff" the information into other Sun windows.

EDIT Selection

Use the selected item as an argument to the **edit** command. For example, get a **directory** listing of your current directory, highlight a drawing name with the mouse, then pick **edit selection** from the popup menu. GED executes the **edit** command on the selected drawing.

Note: If you do a simple **directory** listing (showing the drawing names without extensions or version numbers) and then choose **edit selection** from the popup menu, GED automatically appends a LOGIC.1.1 extension to the drawing name. If you are trying to edit a BODY drawing, you get the message, "`New drawing started`," and GED creates a LOGIC drawing with the same name as the BODY drawing.

To avoid creating the wrong type of drawing, use the command **directory *.*** to include drawing extensions and version numbers in the listing. Then when you choose **edit selection**, GED opens the correct drawing.

GET Selection

Use the selected item as an argument to the **get** command. **get** replaces the current copy of a drawing with the version stored on the disk. The new copy replaces any previously read (and perhaps modified) version in GED.

ADD Selection

Use the selected item as an argument to the **add** command. For example, list the lsttl library, highlight a part name with the mouse, then pick **add selection** from the popup menu. GED adds the selected part to the current drawing.

Split View

Split the message window into any number of subwindows. When you enter information or when GED prints messages, the active subwindow(s) updates to show the current position. If you scroll backwards in a subwindow to a previous view, the previous view remains visible in that subwindow.

Destroy View

If there is only one message window, **destroy view** appears on the menu in shaded gray text and is not available as a selection. **destroy view** only appears as a menu option if you have used the **split view** command to create a subwindow. To close a subwindow, place the cursor in the window and select the **destroy view** menu option.

Reset

Clear the subwindow of its contents. If there is no new text in the window, the window resets immediately. If there is new text in the window, the system asks you to confirm the window reset.

Caret to Top

Reposition the message window so the line containing the caret cursor is near the top of the window. This is useful when you have been scrolling the window and the caret is not currently visible.

Find

Search through the message window for the selected item. **Find** has an additional popup menu:

Find, forward
Find, backward
Find Shelf, forward
Find Shelf, backward

Refer to the Sun manual Windows and Window Based Tools *for more information on the* shelf *facility.*

Find, forward is the default. **Find, backward** searches in reverse for the selected item. **Find Shelf, forward** and **Find Shelf, backward** search forward and reverse for the text that matches whatever text is currently in the cut/paste buffer.

Save

Save the contents of the message window to a file. **Save** has an additional popup menu:

Save file
Save & Quit
Save & Reset
Close & Save
Store to named file
Store & Quit
Close & Store

The first four options default to shaded gray text and are not immediately available for selection. **Store to named file** allows you to store the contents of the message window to a particular file. If there is any selected text in the window, that text is used as the name of the new file. If no text is selected, the system requests a filename entry.

After you select the **Store to named file** option, all the menu options are available for selection.

- **Save file** writes to the current file until you use **Store to named file** to change to another file.
- **Save & Quit** and **Store & Quit** save the window contents to the current file or a named file and exit GED. *This is an abnormal GED exit;* these options *do not* allow GED to clean up temporary files or check to see if there are changes to save.
- **Close & Save** and **Close & Store** save the window contents to the current file or a named file and close GED to an icon. **Save & Reset** saves the window contents to the named file and clears the message window.

2 Customizing Menus

This section explains:

- Creating menus
- Creating menu icons
- Loading menu definition files

On the Sun workstation, you can design custom menus for your own use. By changing the number of menu boxes and size of the menu window, you can have any number of commands available on a menu. Icons allow you to create graphic menus as well as command–name menu boxes.

The Sun workstation also supports *popup* menus, which allow you to group commands and arguments and attach them to one menu item to simplify selection. There can be more than one level of popup menu selection so you can group items as required. Selecting an item from the popup menu appends the popup item to the menu item, separated by a space. Any other selection (outside the popup menu), abandons the popup menu.

Note: On the Sun workstation, the UNIX command *setkeys lefty* makes no impression on GED. The command reverses the left and right function key positions, but the **assign** and **show keys** commands still interpret key names as though the function keys were in their original order.

Creating Menus

The global menu definition file, *ged.menu*, defines the default on-screen menu that you see on the right-hand side of the screen when you start GED. Only the System Manager can modify this global menu.

If you want to change the default menu on your own GED display, there are other pre-defined GED menus available. All the pre-defined GED menus can be found in the directory:

/usr/valid/tools/editor/menus

Figure 2-1 shows three pre-defined GED menus, *ged.menu, pr.menu,* and *gr.menu.*

You can also create your own menu definition file and change your *startup.ged* file to read in the file you have created. Creating a menu definition file is described following the figure.

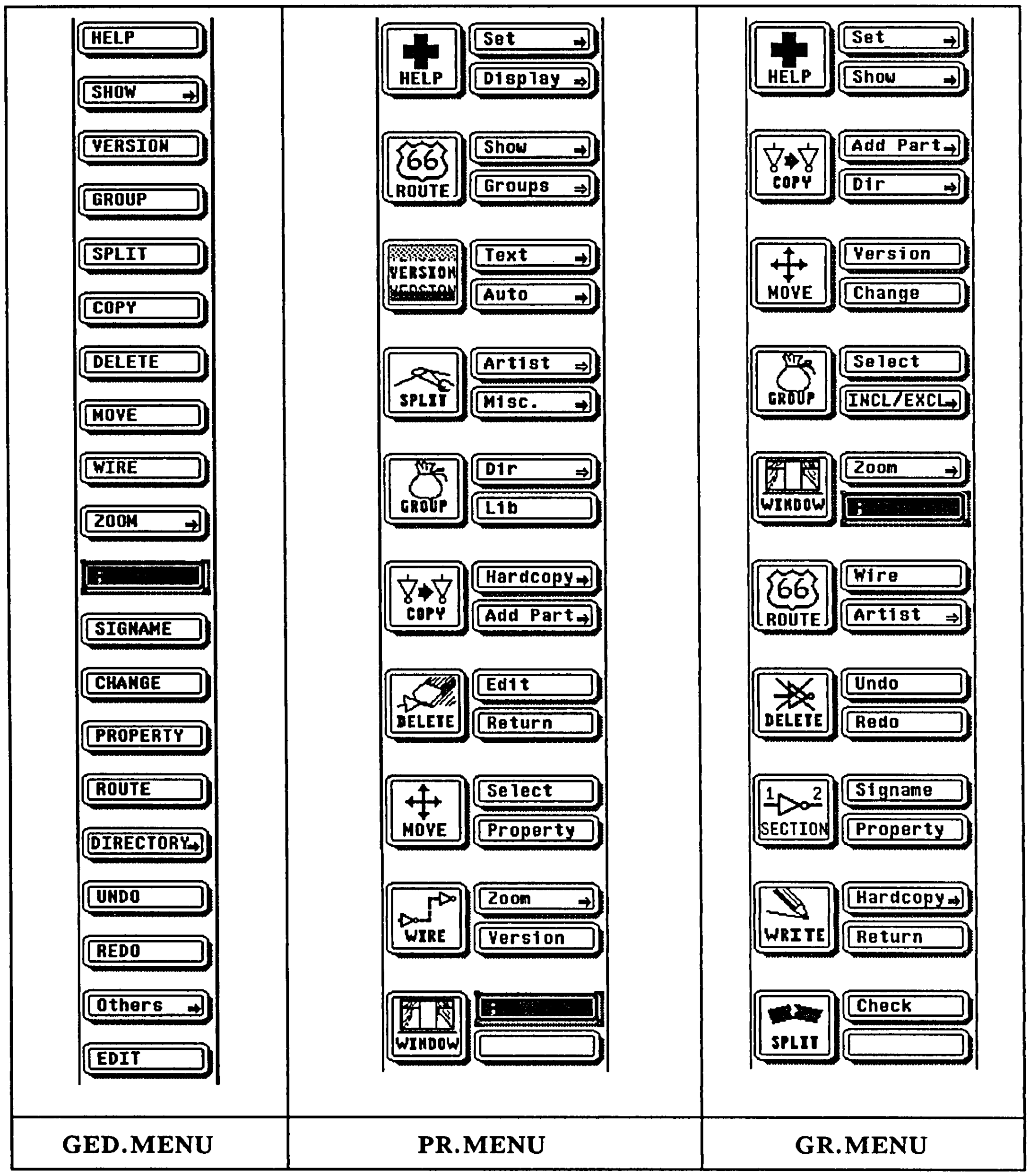


Figure 2-1. Sample GED Menus

The Menu Definition File

The menu definition file determines:

- Menu dimensions
- Popup menus associated with any of the menu boxes
- The size, location, appearance, and functions of the menu boxes

Figure 2–2 shows a portion of the menu definition file. The backslash (\) at the end of a line indicates that the command line is continued on the following file line, and a period (.) indicates the end of a command line. Curly braces ({ }) enclose comments, which are ignored by the program.

```
FILE_TYPE = MENU_DEFN.
dimensions (20, 3).
assign_popup <zoom>\
          "FIT" "FIT",\
          "PREVIOUS" "PREVIOUS",\
          "; (REFRESH)" ";",\
          "LEFT" "LEFT",\
          "RIGHT" "RIGHT",\
          "UP" "UP",\
          "DOWN" "DOWN",\
          "IN 1.625" "1.625",\
          "OUT 1.625" "-1.625",\
          "Enter points" ""\ {let user define zoom}
          .

assign_box (1, 1) 13_BOX "HELP" "HELP".
assign_box (2, 1) 13_BOX "SHOW" "SHOW".
assign_box (3, 1) 13_BOX "VERSION" "VERSION".
assign_box (4, 1) 13_BOX "GROUP" "GROUP".
assign_box (5, 1) 13_BOX "SPLIT" "SPLIT".
assign_box (6, 1) 13_BOX "COPY" "COPY".
assign_box (7, 1) 13_BOX "DELETE" "DELETE".
assign_box (8, 1) 13_BOX "MOVE" "MOVE".
assign_box (9, 1) 13_BOX "WIRE" "WIRE".
assign_box (10, 1) 13_BOX "ZOOM" <zoom> "ZOOM".
assign_box (11, 1) 13_BOX ";" ";".
assign_box (12, 1) 13_BOX "SIGNAME" "SIGNAME".
assign_box (13, 1) 13_BOX "CHANGE" "CHANGE".
assign_box (14, 1) 13_BOX "PROPERTY" "PROPERTY".
assign_box (15, 1) 13_BOX "ROUTE" "ROUTE".
assign_box (16, 1) 13_BOX "DIRECTORY" "DIRECTORY".
assign_box (17, 1) 13_BOX "UNDO" "UNDO".
assign_box (18, 1) 13_BOX "REDO" "REDO".
assign_box (19, 1) 13_BOX "Others" "".
free_box (20, 1) 13_BOX "EDIT" "EDIT".
end.
```

Figure 2–2. Partial Menu Definition File

Defining the File Type

The first and last lines of the definition file are required. The first line tells GED what type of file this is; the last line defines the end of the file.

```
FILE_TYPE = MENU_DEFN.

end.
```

Defining the Window Size

The **dimensions** statement specifies the overall size of the menu window.

SYNTAX

dimensions *length_of_column, window_width*

length_of_column The number of menu boxes in a column. The maximum number of boxes per column varies depending on the size of the menu boxes.

#_of_columns Width of the menu window.

EXAMPLES

```
dimensions (20, 3).
```

The default window dimensions for ged.menu.

```
dimensions (20, 5).
```

The window dimensions for gr.menu *and* pr.menu.

Defining the Popup Menus

Any menu boxes that require associated popup menus must be defined.

SYNTAX

assign_popup <*popupname*> "*send_string*" "*show_string*",

< *popupname* >

The name of the popup menu. The *popupname* can be any name up to 24 characters long. It must be enclosed in angle brackets.

"*show_string*"

The text to display on the screen. There is a maximum of 80 characters per *show_string*.

"*send_string*"

The command string to send to GED. There is a maximum of 256 characters per *send_string*.

There must be one *send_string-show_string* pair for each item that appears in the popup menu. Separate the pairs with a space and the items with commas. End the entire popup definition with a period.

A menu box that has an associated popup menu automatically displays an arrow on the right-hand side of the box.

EXAMPLE

```
assign_popup <zoom>\
          "FIT" "FIT",\
          "; (REFRESH)" ";",\
          "LEFT" "LEFT",\
          "RIGHT" "RIGHT",\
          "IN 1.625" "1.625",\
          "OUT 1.625" "-1.625",\
          .
```

Defining the Menu Boxes

There are two types of menu boxes:

- Static function
- Variable function

Static function boxes are assigned specific functions. They can send command strings to GED or display popup menus. Static functions can be changed only with the **assign_box** statement. Most of the boxes in the menu window are static function boxes.

Free boxes are assigned functions as you issue GED commands during an editing session. If you use a command that is not already assigned to a static function menu box, the command is assigned to a free box. If all the free boxes are assigned commands and you use a command not assigned to a menu box, that command replaces the function in the least recently used free box.

Lines beginning with the **assign_box** statement define static function menu boxes; lines beginning with the **free_box** statement define variable function free boxes. The command arguments are the same for both types of boxes.

SYNTAX

assign_box (*row,col* **)** *size menu_entry*
free_box (*row,col* **)** *size menu_entry*

(*row, col*)

The location of the box in the menu window, counting from (1, 1) in the upper left corner of the window. At least one menu box must be defined with a row number of 1. The maximum number of boxes per column varies depending on the size of the menu boxes.

When you assign boxes to rows and columns, you can place boxes next to each other or you can overlap the rows and column of boxes. Figure 2-3 shows a menu configured with overlapping boxes and one defined with adjacent boxes. Since overlapping boxes can be difficult to read, you should plan your menu layout to avoid confusion.

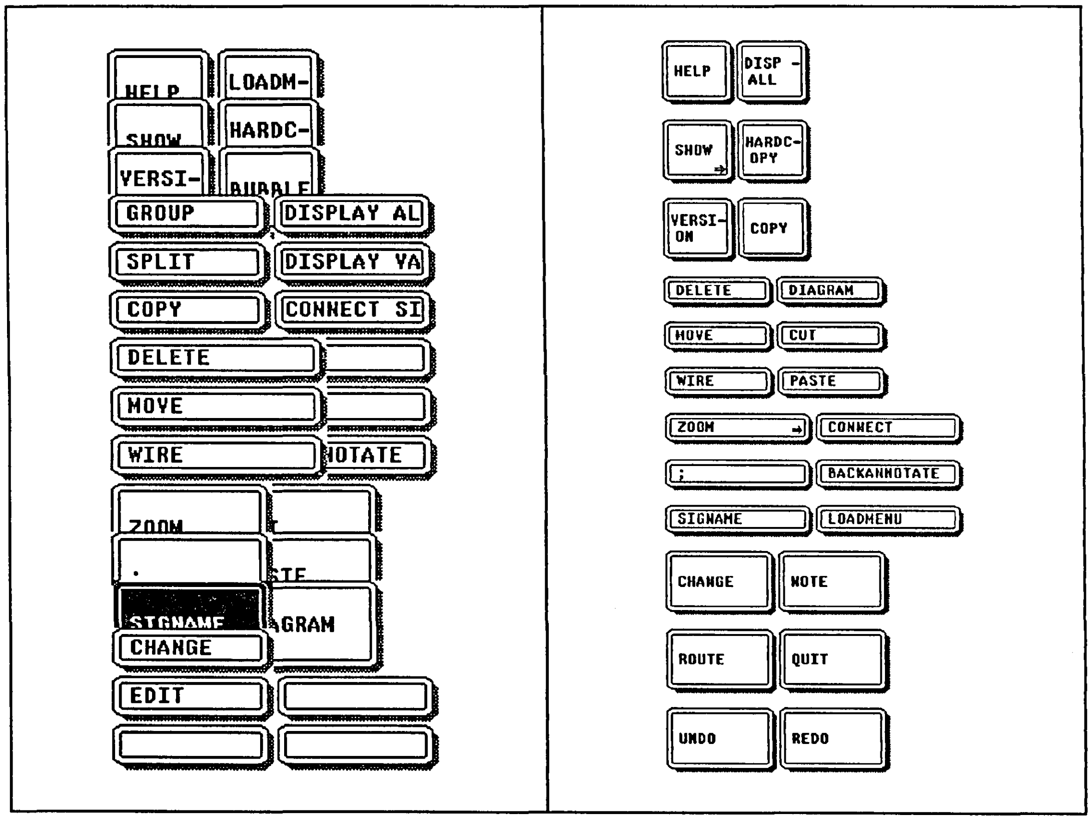


Figure 2-3. Overlapping and Adjacent Menu Boxes

size — The size of the menu box. There are four predefined menu box sizes, as shown in Figure 2–4. If you enter an illegal box size, the menu box defaults to a 22_BOX.

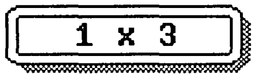
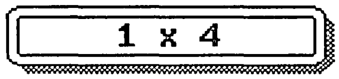
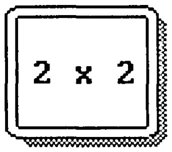
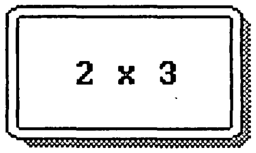
Box Type		Size in Pixels (h x w)	Size in Menu Units (h x w)	Maximum Boxes per Column
13_BOX		27 x 94	1 x 3	28
14_BOX		27 x 126	1 x 4	28
22_BOX		56 x 62	2 x 2	14
23_BOX		56 x 94	2 x 3	14

Figure 2–4. Menu Icon Box Sizes

menu_entry	The *menu_entry* consists of the text or icon to display on the menu box, an optional popup menu name, and the command string to send to GED. The *menu_entry* format is: *"text_label" <popupname> "send_string"*, or *<icon_file> <popupname> "send_string"*,
"text_label"	A text string of up to 80 characters to display in the menu box. The *text_label* must be enclosed in quotation marks.
<icon_file>	The name and full directory path of a file defining an icon to display in the menu box. An *icon_file* must be enclosed in angle brackets; do not enclose an *icon_file* in quotation marks or it will be interpreted as a text string. You can use an *icon_file* definition from any directory as long as the directory pathname is included in the *icon_file* definition. You cannot create icons in the default icon directory; see *Creating Menu Icons*, page 2–14, for information on designing your own menu icons.
<popupname>	The name of a previously-defined popup menu to access when you select the menu box. The *popupname* must be enclosed in angle brackets.
"send_string"	The command string to send to GED when the menu box is selected. The complete string, including strings appended through popup selections, can have a maximum of 256 characters.

EXAMPLES

```
assign_box (1, 1) 23_BOX "HELP"
"HELP".

assign_box (15, 1) 22_BOX \
</icons/section.icon> "SECTION".

assign_box (18, 2) 22_BOX \
</usr/catie/test.icon> "".

assign_box (2, 3) 13_BOX \
"show" <show> "show".

free_box (20, 1) 14_BOX \
"edit" "edit".

free_box (24, 2) 14_BOX \ "" "".
```

Creating Menu Icons

The directory */usr/valid/tools/editor/menus/icons* contains the files for any icons predefined for your system. This is a read-only directory; you cannot write icon files to this location. You must create new icons in another directory.

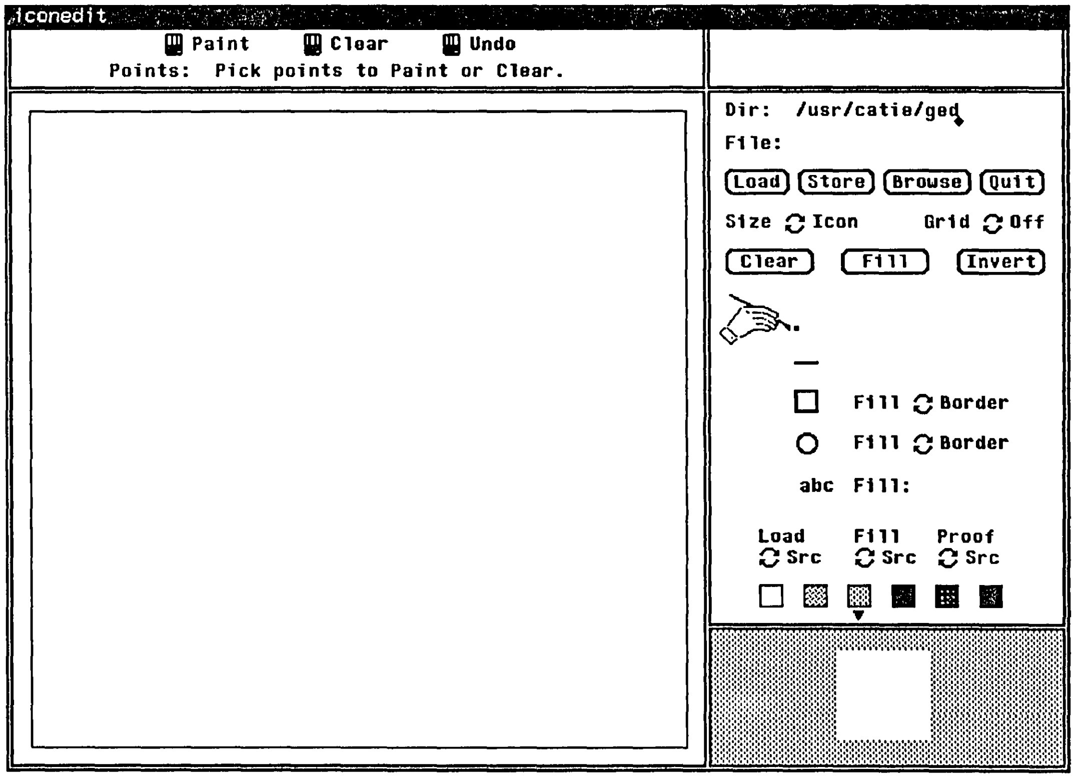
A menu definition file can include icons from any number of directories as long as the directory path is included in the **assign_box** statement for the icon.

To have the menu box border around any icons you create, the menu box templates must reside in the directory where you are creating icons.

To create a new icon:

1 Copy the menu box templates from the */usr/valid/tools/editor/menus/icons* directory into the directory where you are creating icons.

2 Select the IconEditor from the Suntools menu. The IconEditor form is displayed.



Points: Pick points to Paint or Clear.

Dir: /usr/catie/ged

File:

Size Icon Grid Off

Fill Border

Fill Border

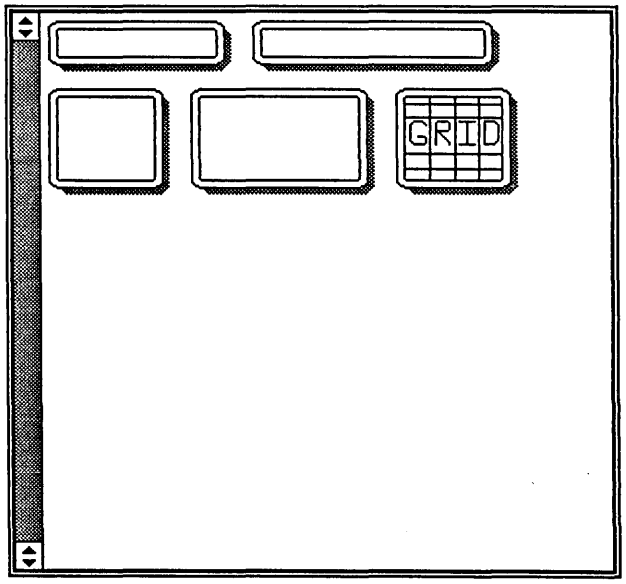
Fill:

Load Src Fill Src Proof Src

Figure 2–5. The IconEditor Form

3 Change to the directory where you want your icons to reside. Use the Delete key to remove the directory name at the top of the form, and type in the new directory name. As you enter characters, the beginning of the entry disappears to the left.

4 Select the Browse box with the mouse to see the existing icons in the current directory. System messages state the number of files found in that directory and how many icon images are being loaded. A new window displays the image icons.



5 Use the mouse to select the template you want to use. The icon image window disappears, and the template icon appears for editing.

6 Select the File line (below the directory name) with the mouse. Delete the existing file name and type in a **temporary** name to assign to your new icon (see below). Include the *.icon* file extension.

7 Create your icon, then select the Store box to save the icon.

8 Select the Quit box to exit the editor.

The menu system does not add an arrow symbol to menu boxes read from an icon file even if the box has a popup associated with it; menu icons are displayed as defined. You can include an arrow symbol, when appropriate, in the corner of icons that you create.

Next you need to resize the new icon for use in the GED menu. The **fixicon** command is used to correct the icon dimensions. The **fixicon** command is located in the */usr/valid/tools/editor/menus/icons* directory.

SYNTAX

fixicon *temporary.icon menu.icon*

temporary.icon	The temporary name of the icon you created with the Icon Editor.
menu.icon	The actual name of the icon to include in the GED menu.

EXAMPLE

```
fixicon xxx.icon grid.icon
```

Loading Menu Definition Files

A menu definition file is loaded with the **loadmenu** command. This command may be typed in the GED message window or it may be included in your *startup.ged* file so that your menu is loaded when you enter the editor. In either case, the menu definition file must be in the correct format and must have a name ending with a *.menu* extension.

You can load menus on either side of the drawing area. The right-side menu is the default. The command **loadmenu left** loads the menu on the left side of the screen. If no menu position is specified, the new menu is loaded on the same side as the current menu position.

When you specify the name of the file containing the desired menu, include the *.menu* filename extension. If it is not present, you get an error message and no menu is loaded.

Index

ValidGED™

User's Guide

January 15, 1989

P/N: 900–00576

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, California 95134
(408) 432–9400 Telex 371 9004
FAX 408 432–9430

MANUAL REVISION HISTORY

Rev	Date	Software Release	Reason for Change
A	1-15-89	GED 9.0	Initial Release

Table of Contents

Introduction

The topics covered in this manual include:

- An overview of GED
- The GED editing environment
- Creating and printing a design using GED
- Drawing Maintenance
- GED Files

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

GED includes features and commands specifically developed for drawing schematics:

- Extensive component libraries for the most commonly used logic families are available for access through the editor. GED also provides facilities for designing and creating your own bodies or symbols that can be added to a drawing or component library.
- The interconnecting (wiring) of component bodies is done with conventional orthogonal (bent) wires. Direct (diagonal) wires are also available.
- A special feature called dynamic dragging allows bodies to be moved in real time; wire connections are maintained when bodies are moved.
- Body versioning and rotating are supported.
- Properties can be assigned to bodies or wires to specify circuit attributes.
- Notes can be added to document 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 commonly-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 GED. These sections are included:

GED Overview	General information about using GED and your workstation.
The Editing Environment	The file and directory structures used by GED; working efficiently in the GED editing environment.
Creating a Design	Using GED commands to create a schematic.
Design Techniques	Using GED to create hierarchical and structured designs.
Producing a Hardcopy	Making a plot of your schematic.
Adding Physical Information	Adding information about physical part assignments to your logical design.
Mixing Text and Graphics	Using GED to produce a mixed text and graphics document.
Drawing Maintenance	Updating drawings and recovering from system failures.
GED Files	The format of the files created and used by GED.
Hardcopy Fonts	Illustrations and ASCII codes for supported fonts.
Batch and Nongraphical GED	Nongraphical GED and file redirection.

Manual Set

In addition to this manual, information on GED can be found in the following manuals:

- *ValidGED Command Reference Manual*
- *Using ValidGED on Your Sun Workstation*
- *ValidGED Tutorial I: Logic Design*

Documentation Conventions

Figure 1 lists the documentation conventions used in this manual. The UNIX operating system distinguishes between uppercase and lowercase letters, and UNIX commands are normally entered in lowercase letters. UNIX commands must be entered *exactly* as they appear in the text.

Convention	Meaning	Example
bold font	Literal keyboard input	**set path =**
[optional]	Optional user input; brackets are not entered	[-options]
keyname	Name of key or button the user should press	Return *
italic font	Variables; must be replaced by specific values supplied by the user	*user_name*

Figure 1. Documentation Conventions

* On the PC AT platform, the Return key is labelled Enter.

1 GED Overview

This section introduces general information about using GED and the workstation:

- Command conventions
- Access and exit procedures
- Elements of the screen
- The keyboard and default function keys
- The cursor controller
- On-line help

GED creates logic drawings (schematics) and body drawings (shapes of parts) using a high resolution CRT display, alphanumeric keyboard, and cursor controller. In addition to creating and modifying drawings, GED interacts with the operating system to retrieve and store drawings.

GED Commands

Commands are issued to GED using both the on-screen menu and the keyboard. These commands allow you to place bodies on the drawing, connect pins with wires, add text information (such as signal names and notes), and manipulate the information contained in the drawing.

File names and text added to drawings are case-sensitive.

GED is case-insensitive and recognizes commands typed in either uppercase or lowercase letters.

GED commands are structured so that the system recognizes both the complete command name and the smallest unique portion of the command name. For example, the **edit** command can be issued by typing any of the following:

```
EDIT

edit

EDI

ed
```

Refer to the *ValidGED Command Reference Manual* for an alphabetical reference of all GED commands.

Accessing GED

On a SCALDsystem, you need one large window. Refer to the System Utilities Reference Manual *for information about creating windows on a SCALDsystem.*

Follow these procedures to start the Graphics Editor and edit a drawing:

1. Turn on your workstation and log in.

2. Make sure you are in your own login directory or in the appropriate project directory.

3. At the prompt, type:

   ```
   ged Return
   ```

   Refer to Section 3, The Editing Environment, *for more information about directory and drawing names.*

   You can enter the name of the drawing as an option to the **ged** command. If the drawing name contains spaces or other special characters, such as angle brackets (< >), place quotation marks around the name.

   EXAMPLES

   ```
   ged "<scald_dir>cpu board.logic.1.2"
   ged counter
   ```

4. To begin working on a drawing, type:

   edit *drawing_name* Return

   drawing_name is the name of the drawing you wish to edit. If the drawing exists, GED accesses the appropriate drawing file and displays the drawing on the screen. If you are creating a new drawing, a blank page is displayed.

Exiting GED

The following procedures describe the two ways to exit from GED and return to the system prompt:

Exit and Save

To exit from GED and *save* your current drawing:

1 Type:

`write` Return

2 Type:

`exit` Return

or

`quit` Return

to return to the system prompt.

Exit without Save

To exit from GED *without saving* changes in your current drawing:

1 Type:

`exit` Return

or

`quit` Return

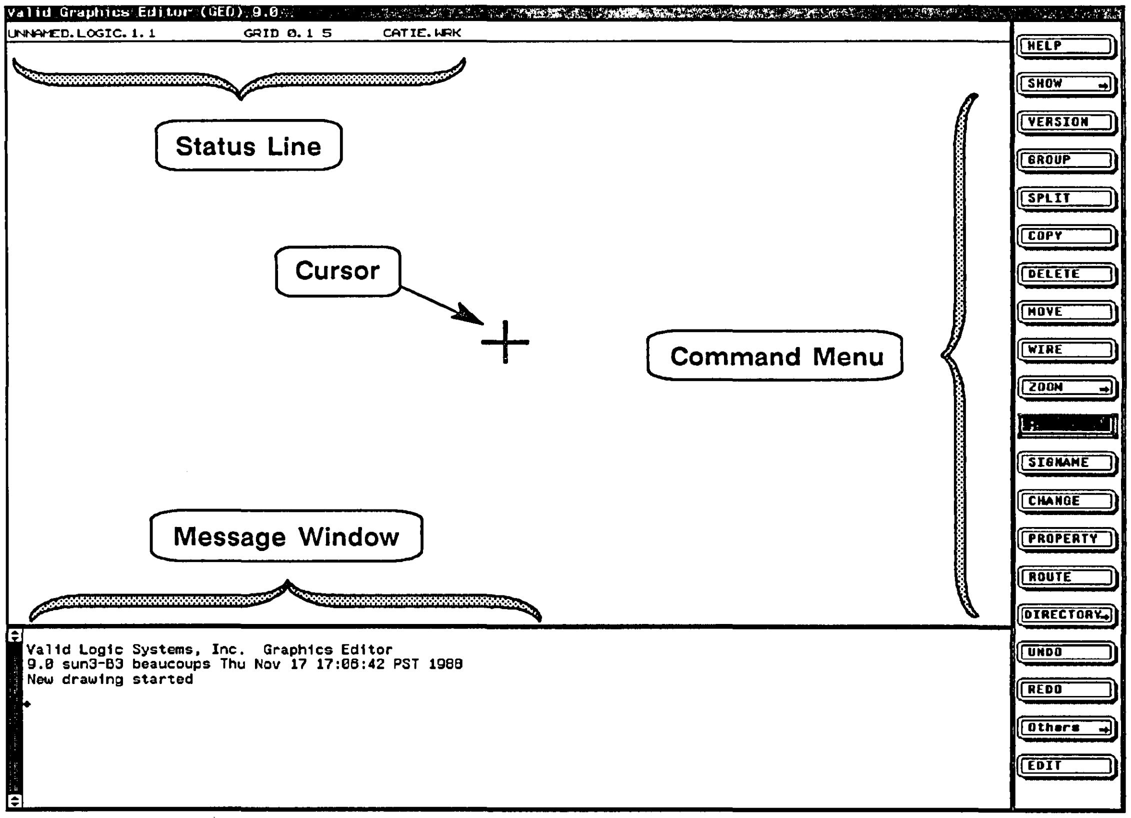
GED prompts you if there are unwritten changes to any drawings you were editing.

2 Retype the command and press Return to override the warning, discard the changes, and exit from GED.

The GED Screen

After you issue the **ged** command, the GED window, cursor, on-screen menu, status line, and message window are displayed on the screen. Figure 1-1 shows a typical GED screen display. There may be minor differences in the screen display on different hardware platforms.

If you specified the drawing name on the command line, that drawing is displayed on the screen.



```
UNNAMED.LOGIC.1.1            GRID 0.1 5        CATIE.WRK
```

```
Valid Logic Systems, Inc.  Graphics Editor
9.0 sun3-B3 beaucoups Thu Nov 17 17:06:42 PST 1988
New drawing started
```

Figure 1-1. Typical GED Screen

The GED window includes these four items:

- Cursor
- Command menu
- Message window
- Status line

GED Cursor

The *cursor* appears as a cross on the screen. You move the cursor by moving the puck or mouse (cursor controller). You 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 cursor controller. You also use the cursor to wire components, draw lines, position library parts, and move items on drawings.

On-Screen Menu

A *menu* of frequently-used commands appears along the right side of the screen. Most of the command names are self-explanatory. The semicolon (;) on the menu is used to end commands. Some of the boxes on the menu may be *free boxes*, where the last commands issued from the keyboard are displayed. The currently-active command is highlighted. The **menu** and **loadmenu** commands allow you to determine which commands are displayed on the menu.

Status Line

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

Message Window

If you make a mistake during command entry (before pressing Return*), you can correct it by using the backspace key and retyping the command correctly.*

At the bottom of the screen is the *message window*. This is where you type in commands and receive status messages from GED.

GED output that requires more than one message window page pauses after each page and inquires:

```
    More? [ync]
```

The possible responses are:

y	Yes. Present more information.
n	No. Do not print any more output.
c	Continue. Print the entire message output without pausing for page prompts.

You can also respond *q* (for *quit*), which acts like a *no* response, or you can enter a Return, which acts like a *yes* response.

You can resize the drawing and message windows within the GED frame. Grab the border between the message and drawing windows, press the middle mouse button, and move the border up or down to resize the windows. (On the Sun system, press the Control key and the middle mouse button simultaneously to resize the windows.)

Program Messages

GED informs you when some operations such as **write**, **group**, and **check** are complete and lists information requested by commands such as **set**, **library**, **directory**, and **show**. GED also displays informational messages to let you know when it cannot interpret or perform a command you entered. *Messages* are displayed in the message window.

Grid Display

GED uses a *grid* to define where objects can be placed on a drawing. When you first access GED, the grid display is turned off.

The **grid** command is a toggle command. Typing:

```
grid Return
```

turns the grid *off* if it is already displayed, or *on* if it is currently not displayed. The command:

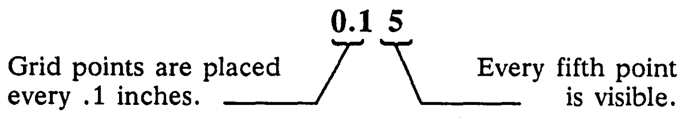
```
grid on Return
```

always turns the grid display on. The command:

```
grid off Return
```

always turns the grid display off.

The default grid setting is displayed on the status line:


0.1 5
Grid points are placed every .1 inches.
Every fifth point is visible.

To change the default grid size, use the command **set default_grid**.

Valid component libraries depend on the default grid setting to function properly. The grid for body drawings is changed to a setting that is twice as fine as the default grid. *You should use caution when you change the default grid setting; bodies could be placed off the grid and wires might not be connected.*

The Keyboard

The keyboard is a standard alphanumeric keyboard with programmable function keys. The function keys on your keyboard are predefined with the most commonly used GED commands. The primary purpose of the keyboard is to allow you to type commands, signal names, properties, notes, and other text information required to create a drawing. The programmable function keys allow you to enter commands with a single keystroke.

You can change the default function key assignments or program additional function keys with the **assign** command. Refer to the **assign** command description in the *ValidGED Command Reference Manual* for more information.

To see the function key assignments programmed for your system, type:

The number keys on the numeric keypad of the VAX workstations are named K0 through K9.

```
show keys [Return]
```

Table 1–1 lists the default function key assignments.

Table 1-1. Default Function Keys

Sun	SCALD	PC AT	VAX	Description
F1	-	-	Help	**help** - Display the on-line help screen
F2	LF2	F2	F7	**window fit** - Redisplay the drawing to fit the screen
F3	LF3	F3	F8	**display both** - Display the name and value of the selected properties
F4	LF4	F4	F9	**show attach** - Display attachments between properties and objects
F5	LF5	F5	F10	**window ;** - Refresh the screen
F6	LF6	F6	F11	**show prop** - Display the name and value of all properties
F7	LF7	F7	F12	**directory** - List the drawings in the current directory
F8	LF8	F8	F13	**display 1.25** - Enlarge selected text 25%
F9	LF9	-	F14	**display 0.8** - Reduce selected text 80%
R1	RF1	F1	PF1	**hardcopy** - Plot the current drawing
R2	RF2	F9	PF2	**undo** - Undo previous operation(s)
R3	RF3	F10	PF3	**redo** - Redo previous **undo** operation(s)
R4	RF4	-	K7	**auto path** - add path properties to a drawing
R5	RF5	-	K8	**check** - Examine drawing for errors
R6	RF6	-	K9	**error** - Display errors located by **check**
R7	RF7	-	K4	**return** - Display a previously-edited drawing
R8	RF8	-	K5	**edit** - Enter the **edit** command
R9	RF9	-	K6	**bubble** - Bubble the selected pin

On the numeric keypad of the Sun workstation and the SCALDsystem, several keys perform different functions when pressed with the Shift key. These shifted function keys are all **zoom** command functions. The keys are shown in Table 1-2. Use the **show keys** command to be sure the function keys have not been reassigned.

Table 1-2. Shifted Function Keys

Sun	SCALD	Command	Description
R4	RF4	**zoom fit**	Redisplay the drawing to fit the screen
R5	RF5	**zoom up**	Reposition the center of the screen up above the drawing (move the drawing down on the screen)
R6	RF6	**zoom previous**	Switch from the current zoom scale/position to the previous zoom scale/position
R7	RF7	**zoom left**	Reposition the center of the screen to the left of the drawing (move the drawing right on the screen)
R8	RF8	**zoom ;**	Refresh the screen
R9	RF9	**zoom right**	Reposition the center of the screen to the right of the drawing (move the drawing left on the screen)
R10	RF10	**zoom out**	Reduce the size of the drawing on the screen
R11	RF11	**zoom down**	Reposition the center of the screen down below the drawing (move the drawing up on the screen)
R12	RF12	**zoom in**	Enlarge the size of the drawing on the screen

Several keyboard characters perform different functions when pressed with the Control key. Table 1-3 describes these terminal characters and their operations. The Delete key also performs a special function on some keyboards.

Table 1-3. Control and Delete Key Operations

PC AT and SCALDsystem	
Control -C	Interrupt the current GED operation
Control -R	Redisplay the current input line
Control -U	Back up to the beginning of the current line
Control -W	Back up to the beginning of the previous word
VAX Workstations	
Control -C	Interrupt the current GED operation
Control -R	Redisplay the current input line
Control -U	Erase the current line
Delete	Erase the previous character
Sun Workstations	
Control -C	Interrupt the current GED operation
Control -R	Redisplay the current input line
Control -U	Back up to the beginning of the current line
Control -W	Back up to the beginning of the previous word
Delete	Erase the previous character

On the SCALDsystem and Sun workstations, all of these characters can be reassigned. Use the command **stty all** to determine the character assignments. On the PC AT, you cannot change the word-erase and reprint-line functions. On VAX workstations, none of these functions can be reassigned.

Cursor Controller

The actual type of cursor controller you use to move the cursor and issue commands to GED depends on the hardware options purchased with your Valid system. Whether you have a puck or a mouse, the buttons are used to select commands or specify points on the drawing. The actual operation performed depends on the command being issued, the position of the cursor, and the particular button that is pressed.

The puck has four buttons: yellow, white, green, and blue. A mouse has three buttons: the left button performs the same operation as the yellow button, the center button performs the same operation as the white button, and the right button performs the same operation as the blue button. In this manual, the buttons are referred to by color.

To select a command from the menu, simply position the cursor in the box containing the command and press any button. To indicate "points" in your drawing to GED commands, you must press specific buttons. GED's interpretation of a point depends on which command is executed and which button is pressed. Buttons are pressed and immediately released. GED does not require buttons to be held down during operations.

The yellow and white buttons use the nearest grid intersection as the point for the operation being performed. The blue and green buttons refer to the vertex, or attachment point, of the nearest object.

For example, if you start a wire with the yellow 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 blue button. One press of the blue button snaps to the nearest vertex of an object or wire endpoint.

When you use the **wire** command, pressing the white or green buttons changes the direction of a wire.

For other commands, the white and green buttons operate on the nearest defined group; the blue and yellow buttons operate on the nearest individual object.

Figure 1–2 shows the mouse and puck cursor controller buttons and their operations.

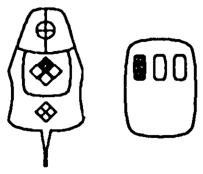
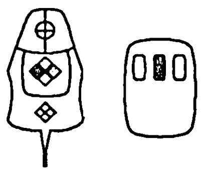
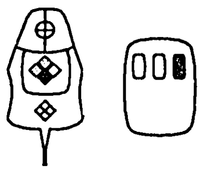
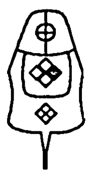
Yellow Button		Selects commands Selects items to be edited Attaches items to nearest grid point Picks up items Starts wires at the nearest grid point
White Button		Operates on groups Toggles through **wire** and **zoom** command options Rotates bodies in the **add** command
Blue Button		Attaches items at the nearest vertex Picks up items at the nearest vertex Starts wires at the nearest vertex Selects nearest object for **group** and **select** commands Pans during **zoom** command
Green Button		Operates on groups Toggles through **wire** and **zoom** command options

Figure 1–2. Cursor Controller Operations

On–Line Help

For more information on the **window** *command, see Section 3,* Creating a Design, *or refer to the* ValidGED Command Reference Manual.

On–line help is provided for each GED command to allow you to use GED efficiently. The top command on the GED menu is **help**.

To get help on a command:

1. Select **help** from the menu.
2. Select a topic (command) from the menu with the cursor or type the command name on the keyboard and press Return.

When help is displayed, you can use the **window** command to enlarge the text in small windows. For instance, to enlarge the text by 25%, use the command:

```
window 1.25
```

There are three ways to display a list of the commands for which **help** is available:

- Move the cursor to the **help** menu and press the yellow button twice.
- Type:

  ```
  help Return
  ```
- Type:

  ```
  dir <help>* Return
  ```

To exit from the **help** command, type or select any GED command except **window**.

2 The Editing Environment

This section explains:

- SCALDsystem default files
- SCALD directory system
- File and drawing names
- Basic drawing operations
- Creating a new project directory

GED drawings are stored in the operating system in a series of related files that make up a design database. The SCALDsystem software allows you to locate, store, and manipulate entire drawings using GED commands. The directory system and tools you use to create, store, and manage drawings and files is called the *editing environment*. Since component libraries are also stored in the system in the form of GED drawings, library access is also performed in the editing environment.

SCALDsystem Default Files

Command files, data files, and SCALD directories are the three file types that help you access GED and the other Valid design tools. These files are automatically created in your login directory when your user account is created.

Command Files

The following command files are automatically created when your user account is created.

startup.ged

This file specifies the libraries and SCALD directories that you want to access automatically each time you run GED. You can also enter other GED commands into this file to tailor the default GED environment to your needs. The *startup.ged* file is discussed in greater detail on page 2–5.

compiler.cmd

This command file contains the directives for the Compiler. The Compiler prepares your design for further analysis and processing. See the *ValidCompiler Reference Manual* for additional information on the Compiler.

packager.cmd

This command file contains the directives for the Packager. The Packager tests your design for loading violations and wiring errors, prepares your design for use by a physical design system, and creates the back annotation file read by GED. See the *ValidPackager Reference Manual* for additional information on the Packager.

Depending on the Valid tools included in your system configuration, the following command files may be included in your user account.

simulate.cmd

This command file contains the directives for the Logic Simulator. The Simulator performs detailed simulation of your design at the component level.

verifier.cmd

This command file contains the directives for the Timing Verifier. The Timing Verifier analyzes partial or complete designs for timing errors.

td.cmd

This command file contains the directives for the PLOTTIME program. The PLOTTIME program plots waveform diagrams created by the Simulator and Timing Verifier.

Data Files

The following data files are automatically created when your user account is created.

master.local

This file contains an optional list of abbreviations and pathnames for the SCALD directories you want to access during a design session. The *master.local* file is discussed in greater detail on page 2–5.

Depending on the Valid tools included in your system configuration, the following data files may be included in your user account.

case.dat

This file contains data for the Timing Verifier when you want to test the timing for specific cases.

delay.dat

This file is used to feedback delay information to the Timing Verifier from a physical design system.

SCALD Directories

SCALD directory files end with a **.wrk** *extension in your directory listing.*

Each drawing you create is stored as a group of related drawing files in an operating system directory that is referenced in a special file called a *SCALD directory*. GED automatically manages drawing storage and retrieval operations through this special file.

The SCALD directory is actually a file that maps GED drawing names to the operating system file names where the drawing is stored. The SCALD directory serves two purposes:

- It allows you to refer to drawings by their drawing names rather than by their system names (which may be more cryptic).
- It does not require you to learn system-specific file naming conventions (which are handled automatically).

The SCALD directories make the file system transparent to you. SCALD directories are discussed in greater detail later in this section.

The Startup.ged File

The default *startup.ged* file contains only one command: **use** *username*.**wrk**. This command creates a SCALD directory, *username*.wrk, the first time you write a drawing. When you edit this file:

- Add only one command per line in the file. When defining libraries, use a separate **library** command for each library to be accessed.
- Unlike the other command files, do not add the word **end** to the end of the file.

The **assign** *and* **set** *commands are described in detail in the* ValidGED Command Reference Manual.

Other commands you can add to this file include **assign** (to assign a function key operation) and **set** (to set a default option such as turning the grid on or using filled dots). Although you can enter commands directly from GED, placing commands in the *startup.ged* file automatically executes the commands each time you enter GED. You use a text editor to edit this file.

The Master.local File

The *master.local* file in your directory allows you to associate short names for SCALD directories with their corresponding full file specifications or pathnames. The **masterlibrary** command, which you enter in your *startup.ged* file, specifies the name and location of the *master.local* file.

After editing the *master.local* file and adding your defined short names, you can specify these short names whenever you use commands that require the specification of a SCALD directory, such as **use**, **ignore**, **library**, and **write**.

The format of the default *master.local* file is shown in the example in Figure 2–1.

```
FILE TYPE = MASTER_LIBRARY;
"susan.wrk"      'susan.wrk';
{"proj1.wrk"     '[.proj1]proj1.wrk';}
END.
```

Figure 2–1. Default Master.local File

The `FILE_TYPE` line **must** include spaces around the equal sign. Your user name appears instead of *susan*. Double quotes surround the full name of the file; single quotes surround the abbreviation. The third line contains an example (shown as a comment line enclosed in curly braces) for a hypothetical SCALD directory, called *proj1.wrk,* located in the directory *proj1* just below the login directory. You can specify both full and relative pathnames for the SCALD directories you want to access. You can also use network facilities to access SCALD directories on other workstations.

SCALD Directories

The SCALD directory is a file that GED uses to locate your drawings in the operating system. GED automatically updates this file each time a new drawing is written. For each drawing, GED also creates a separate drawing directory that contains the files that describe the drawing.

A SCALD directory name is specified as a name and an extension. You can use any combination of alphanumeric characters for the name and extension strings. By convention, the name is the same as the operating system directory and the extension is either *.wrk* (for a user directory) or *.lib* (for a library of components). Although you are not required to match the directory name and use the *.wrk* and *.lib* extensions, this convention makes SCALD directories readily visible in the directory listing.

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

```
use username.wrk
```

or by issuing a **use** command from within GED.

Figure 2–2 shows a sample SCALD directory.

```
FILE_TYPE = LOGIC_DIR;
"SHIFT REGISTER"                'shiftregister';
"MEMORY SELECT LOGIC"           'mmryslectlogic';
"32-BIT ALU"                    '32bitalu';
END.
```

Figure 2–2. Sample SCALD Directory

The first line of the file identifies the directory type (`FILE_TYPE = LOGIC_DIR;`). When GED creates a drawing directory, it is a LOGIC directory. A LOGIC directory contains the drawings you create. This is the default directory type. Drawings of any type (LOGIC, BODY, etc.) can be placed in a LOGIC directory.

There are other directory types used for simulation and timing verification; see the ValidCOMPILER Reference Manual *for details.*

In any SCALD directory listing, the GED drawing name appears on the left in uppercase letters and enclosed in double quotes. (Remember that GED drawing names are not case-sensitive.) The system directory where the drawing is stored appears on the right enclosed in single quotes.

In the example, the drawing named `MEMORY SELECT LOGIC` is stored in the UNIX directory `mmryslectlogic`.

In VMS, the directory name is `MMRYSLECTLOGIC.DIR`.

Each time you save a new drawing with the **write** command, GED creates an entry in the appropriate SCALD directory with your name for the drawing and creates the drawing directory where the drawing is stored. The name of the drawing directory is the

GED drawing name, automatically shortened to 14 characters, with special characters removed. Each drawing directory contains the following set of files:

UNIX: **logic.1.1**
VMS: **LOGIC$1$1.DAT**

The graphic information in ASCII format; the file is read by GED.

UNIX: **logic_bn.1.1**
VMS: **LOGIC_BN$1$1.DAT**

The graphic information in binary format; the file is read by GED.

UNIX: **logic_cn.1.1**
VMS: **LOGIC_CN$1$1.DAT**

The connectivity file that contains information about the parts and interconnections in the drawing; the file is read by the Compiler.

UNIX: **logic_dp.1.1**
VMS: **LOGIC_DP$1$1.DAT**

The dependency file that lists each part used in the drawing and its library directory; the file is used by the GED update facility to ensure that the parts in the drawing are current.

See Section 8, Drawing Maintenance, *for information on the GED update facility.*

Listing Directory Information

Within GED, the **directory** command lists information about active SCALD directories. An *active* SCALD directory is one you have accessed through a **use** or **library** command in your *startup.ged* file or during the current GED session.

You can list SCALD directories and related drawings, and you can list the contents of a specific SCALD directory. The following examples illustrate some of the ways you can use the **directory** command:

directory	List all drawings in the current SCALD directory.
directory *	Same as **directory**.
directory <*>	List all active SCALD directories (but no drawing names).
directory <lsttl>*	List all drawing names (parts) in the **lsttl** library.
directory sh*	List all drawing names beginning with **sh** in the current SCALD directory.
directory *.body.*	List all **.body** drawings in the current SCALD directory.
directory <*>*	List all drawings in all active SCALD directories.
directory *.*	List the name, type, and version of each drawing in the current SCALD directory.

In the examples, the asterisk (*) is a *wildcard* character that matches all character strings.

Specifying SCALD Directories

The **use** command entered in the *startup.ged* file can also be issued from within GED to specify a SCALD directory. When the SCALD directory to be used is not in your current directory, you must specify the full pathname or file specification of the "target" SCALD directory.

Alternately, you can use the **masterlibrary** command in your *startup.ged* file to specify an abbreviation file for the SCALD directories you use during design sessions. This facility allows you to specify short names for SCALD directories when you enter the **use** command. A default abbreviation file, *master.local*, is placed in your directory when your account is created. You can add abbreviations to this file.

If you specify a non-existent SCALD directory, a message is displayed, and the directory name specified appears on the status line (it becomes the current SCALD directory). GED creates the actual SCALD directory when you write the drawing.

Creating A Search Stack

When you include multiple **use** commands in your *startup.ged* file, you create a *search stack* of SCALD directories for GED. The order of the **use** commands in the file determines the order in which the SCALD directories are searched. The last SCALD directory listed in the file is the first SCALD directory searched for the specified drawing; if the drawing is not found, the next directory is searched.

You also put **library** commands in your *startup.ged* file to specify which component libraries you require for your design. Libraries are searched according to the first library listed in the *startup.ged* file. Normally, this is of little concern since the part names usually differ among libraries. An exception is the standard and ANSI versions of the same 54 or 74 series library. (The same parts appear in both versions; the body drawings are different.)

Figure 2-3 shows a *startup.ged* file and the search stack that those commands create. The file is assumed to be in the directory proj1.

These GED commands...	**Create this Search Stack**
`use proj2/proj2.wrk`	`proj1.wrk`
`use proj1.wrk`	`proj2/proj2.wrk`
`lib tutorial`	`tutorial`
`lib lsttl`	`lsttl`

Figure 2-3. Search Stack Example

This search stack allows you to access drawings stored in two SCALD directories, *proj1.wrk* and *proj2/proj2.wrk*, and provides access to the TUTORIAL and LSTTL libraries.

When you enter GED from your local project directory (*proj1*), the current, or *active*, SCALD directory is *proj1.wrk* (the SCALD directory on the top of the stack). If an existing drawing to be edited is located in the *proj2* directory, GED searches, in order, *proj1.wrk*, and then *proj2.wrk*.

If the specified drawing does not exist in any of the directories (including libraries) in the search stack, it is *opened* as a new drawing. When the new drawing is written, it is stored in *proj1.wrk*.

The **use** command can be used to add a new SCALD directory to the search stack. Each time you issue the **use** command, the newly-named SCALD directory is put on the top of the stack. The **ignore** command removes a SCALD directory from the search stack.

If you specify a library name with the **use** command, the associated SCALD directory is placed at the top of the active SCALD directory stack.

Borrowing a Drawing From Another User

By default, GED stores an existing drawing in the directory from where it was taken. The **ignore** command allows a SCALD directory to be temporarily deleted in order to "borrow" a drawing from another SCALD directory. To copy a drawing from another SCALD directory, follow these steps:

1 Go to the directory where the copy of the drawing is to be stored.

2 Enter GED. Your current SCALD directory (for example, proj1.wrk) is specified by your *startup.ged* file.

3 Enter the **use** command with the required file specification or abbreviation to change to the new SCALD directory, for example:

```
use /u0/project2/project2.wrk
```

or, if the abbreviation is set up in *master.local*:

```
use project2
```

4 Type:

`edit` *drawing name*

drawing name is the name of the drawing to copy.

5 Type:

```
ignore
```

to delete *project2.wrk* from your active search path and to return to the initial SCALD directory *(proj1.wrk)*. You are prompted to confirm the operation.

6 Type:

```
y          (yes)
```

to confirm the **ignore** operation in response to the GED prompt.

7 Type:

```
write
```

to store the drawing in your SCALD directory.

Before you write the drawing, you can enter the **use** command to specify a different SCALD directory. Alternately, you can specify a different SCALD directory with the **write** command. Unless you have abbreviations set up in *master.local*, specify the pathname or file specification of the SCALD directory.

EXAMPLE

```
UNIX:   write </julie/proj2/proj2.wrk>
VMS:    write <SCALD$ROOT:[TOM.PROJ2]PROJ2.WRK>
```

Note: Do not use the operating system command to copy a GED drawing. No entry would be made in the SCALD directory, and GED would be unable to find the drawing.

Drawing Names

The *drawing name* is used to identify your designs. 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 you to enter a drawing (or component) name.

Drawing names are made up of the following four fields:

name.type.version.page

EXAMPLE

```
subtractor.logic.1.1
```

By default, if only a drawing name is entered (`subtractor`), GED assumes version 1 and page 1 of a LOGIC drawing.

To edit page 2 of the current drawing, type:

```
edit ...2
```

Table 2-1 illustrates GED naming conventions.

You enter:	GED assumes:
`write 32 bit alu`	`32 bit alu.logic.1.1`
`edit nand.body`	`nand.body.1.1`
`edit mux box...4`	`mux box.logic.1.4`
`add nand`	`nand.body.1.1`
`add ls00..2`	`ls00.body.1.2`

Table 2-1. Drawing Name Conventions

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

Name Field

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

EXAMPLE

```
Ansi Disk Controller
32-bit alu
LS112
10109
HIGH-SPEED RAM
```

The drawing name is not restricted to short alphabetic identifiers or to uppercase letters. The name can be up to 255 characters long and can contain any printing ASCII character except the period (.), quotation mark ("), or tilde (~). Internal spaces are permitted.

Type Field

You can change the default type with the **push_type** *option of the* **set** *command.*

The second field of a drawing name identifies the particular type of drawing. If this field 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**. The **add** and **replace** commands assume BODY drawings. The **get** command assumes the type of the currently-edited drawing.

See the ValidCOMPILER Reference Manual *and the* Library Reference Manual *for more information on body types.*

The six standard drawing types are: LOGIC, BODY, TIME, SIM, PART, and PRIM. Generally, only the LOGIC and BODY types are used; the other drawing types are used for library development.

LOGIC

A LOGIC drawing is the standard type of drawing created with GED and is used to define a circuit made up of components (such as TTL or CMOS). Components are defined in the Valid libraries and additional components can be defined in user-created libraries.

A LOGIC drawing can contain library parts (bodies) and hierarchical bodies that represent other logic drawings. 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. These parts are only used in library development.

BODY

A BODY drawing is the symbolic representation of a library part that you add to your design. This drawing defines the shape, pins, and general properties of the library part.

See Section 4, Design Techniques, *for more information on hierarchical drawings.*

When making a hierarchical design, you make a BODY drawing to represent an entire LOGIC drawing made up of library parts. This is called a *hierarchical body*. Using a hierarchical body allows you to refer to a collection of logic without having to include that logic in the drawing.

Version Field

The version field identifies different symbolic representations of body drawings. If the version is not specified, GED assumes version 1.

The drawing version allows you to select different versions of a body. For example, an LS00 NAND gate has two representations. One of the body drawings is called LS00.BODY.1.1, and the other is called LS00.BODY.2.1, as shown in Figure 2-4.

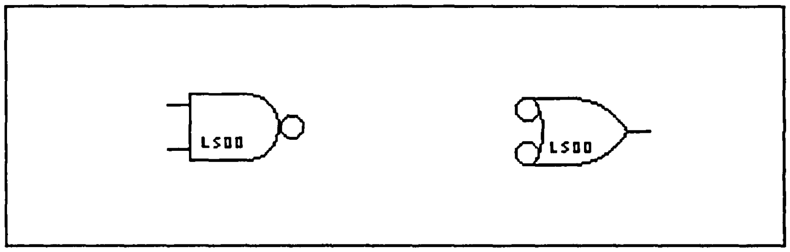


Figure 2-4. LS00 Body - Versions 1 and 2

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

In addition to logical versions, a large number of library components have "sizeable" versions to support the SCALD structured design methodology.

Refer to the SCALD Language Reference Manual *for more information about using select expressions.*

Several versions of a drawing (for example, a timing model) can be created, with each version containing a different value or parameter. You can then use select expressions to specify a particular version for an application or process.

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 Field

The last field is used to create a drawing that extends 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 current 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 unlimited.

Working With Drawings

This section describes basic drawing operations:

- Bringing a drawing onto the screen
- Saving a drawing
- Renaming a drawing
- Deleting a drawing

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 you want 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 your design. The name of the new drawing appears on the status line and is automatically added to the current SCALD directory when you write the drawing.

You can edit a second drawing without writing the current drawing. The **edit** command saves the first drawing in a temporary file and then displays the new drawing. You can edit the first drawing again or you can use the **return** command to display the previously-edited drawing. The **show history** command lists all of the drawings you edited during the current session.

If you make 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 command replaces the drawing you are editing with the original drawing stored on the disk. Type **get** and the name of the drawing.

Saving a Drawing

To save your drawing on disk, use the **write** command. You can specify the SCALD directory where the drawing is to be stored and the drawing name. If you are saving a newly-created design, GED stores the drawing in the current SCALD directory. GED stores a drawing you borrowed from another SCALD directory in the directory from which it was retrieved, unless you delete the SCALD directory from your search stack with the **ignore** command or specify another SCALD directory with the **write** command.

Renaming a Drawing

The **diagram** command allows a copy of a drawing to be saved under a different name. You 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 associated drawing files. To proceed with the delete operation, type a semicolon (;) and press Return. GED deletes the displayed files and the drawing directory and removes the drawing name entry from the SCALD directory.

You can delete a specific page or version of a drawing or a specific drawing type by using the complete drawing name.

EXAMPLES

```
remove cktl.logic.1.2
```

This command removes only page 2 of the drawing cktl.

```
remove cktl.body
```

This command removes the BODY drawing for cktl, *but does not remove the LOGIC drawing.*

Creating a New Project Directory

Normally, you should create a subdirectory for each design project rather than keeping all designs in your login directory. You copy the command and data files over to the new directory and set up a new SCALD directory file for the project drawings.

To create a directory for a new project:

1 Make sure you are in your login directory.

2 Enter the appropriate operating system command to create the new directory.

3 Copy the default files from your login directory into the new directory. These files are:

Use a wildcard character to copy all the .cmd files to the new directory.

- All the command files (*.cmd*)
- *startup.ged*
- *case.dat*
- *delay.dat*

4 Move to the new directory.

5 Edit the *startup.ged* file:

- Change the line **use** ***username*.wrk** to read **use** ***newdirectory*.wrk** (for example, `use proj1.wrk`). *Proj1.wrk* is now the name of the new SCALD directory for this project.
- Add a **library** command for each required library (each library must be defined with a **library** command on a separate line).

- Make sure that the **masterlibrary** command indicates the correct path or file specification for the abbreviation file *(master.local)*.
- Save the new *startup.ged* file.

6 Edit the *compiler.cmd* file (and the *td.cmd* file) and change the name of the SCALD directory that appears in the DIRECTORY directive to the new name. For example, `directory proj1.wrk`.

The first time you save a drawing (using the **write** *command), the SCALD directory* proj1.wrk *is automatically created.*

Now you are ready to start a new project in the directory *proj1*. To start working:

1 Log in and move to your new directory.

2 Type:

```
ged
```

3 Creating a Design

This section explains:

- Adding design elements to a drawing
- Defining groups
- Specifying drawing colors
- Duplicating objects
- Editing drawings and text
- Changing drawing views
- Checking drawings for errors
- Using GED scripts

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

Although there are few restrictions, the following rules ensure compatibility between your schematics and other SCALD design tools.

- You cannot change the *type* of a drawing when you **write** the drawing. For example, if you are editing the drawing *shifter.logic*, you cannot save it as *shifter.sim*. You must use the **diagram** command to change the name and type of a drawing and then write the 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, GED 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 are illegal in TIME drawings. 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 can add parts from any library, including SIM and TIME, to DOC drawings.

Adding Design Elements

All SCALD drawings are constructed from seven objects, or *primitives*:

- Bodies
- Wires
- Signal Names
- Dots
- Arcs
- Notes
- Properties

Each of these items is added to a drawing by one or more vertices.

The *vertex* is normally used to select an object. Design elements have the following vertices:

- Wires have a vertex at each end and at each bend.
- Bodies have a vertex to refer to the body itself and a vertex for each pin.
- Text strings (signal names, notes, or properties) have one vertex, located at the lower left corner of left–justified text strings (the default), the lower right corner of right–justified text strings, and the center of center–justified text strings.
- Circles and arcs have a vertex at the center of the circle.

The **show vertices** command displays asterisks at all the vertices in the drawing. Figure 3–1 shows the vertices for each design element.

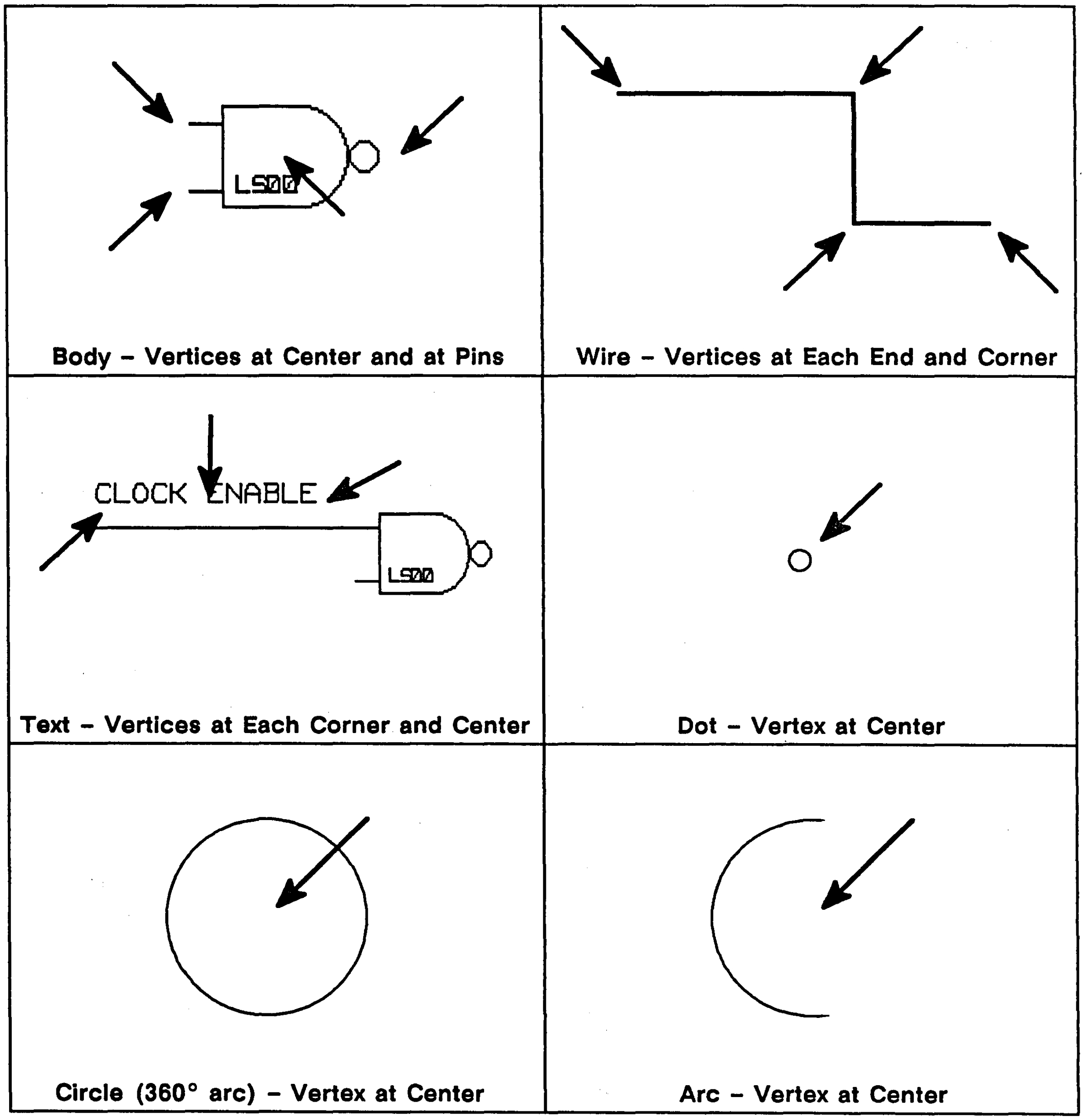

LS00
Body – Vertices at Center and at Pins
Wire – Vertices at Each End and Corner
CLOCK ENABLE
LS00
Text – Vertices at Each Corner and Center
Dot – Vertex at Center
Circle (360° arc) – Vertex at Center
Arc – Vertex at Center

Figure 3–1. Design Element Vertices

Adding Bodies

A *body* is the symbolic representation of a drawing. The body drawing allows you to refer to a collection of logic without having to include that logic in your design. Generally, bodies are the components that are defined in the libraries you purchase with your Valid system software. You can also create bodies to represent repeated sections of a design or circuit.

Refer to Section 4, Design Techniques, *for information about using BODY drawings in hierarchical designs.*

Before you can add a body to your drawing, you have to specify the required SCALD directory with the **library** or **use** command. Also, a Standard library is automatically included with GED. This library contains special bodies that you can use in your designs.

Refer to the Library Reference Manual *for more information on the Standard library.*

To add a body to a drawing:

1 Make sure that the proper library is accessed. You can issue the **library** command in GED or you can include the required **library** command in your *startup.ged* file.

2 Use the **edit** command to begin working on a schematic.

3 Type:

add *bodyname* [Return]

Substitute the name of the required component for *bodyname*. This command attaches the specified body to the cursor.

OPTIONAL: *Before you set the body down, you can rotate it by pressing the white button.*

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

Commands are active *when they are highlighted in a menu box.*

As long as the **add** command is active, you can add bodies to the drawing by typing the name of each body and pressing the yellow button to place it on the drawing.

Also, **add** remembers the last body that was added to the drawing and attaches another copy of it to the cursor if you press any button before you enter a part name. The cursor must be in the drawing area.

Body Versions

Many bodies in the Valid component libraries are represented by more than one version. Body versions support different graphical, but functionally equivalent, 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 with **add** *bodyname*..2, or you can use the **version** command to cycle through all available versions of the body. To use the **version** command, select **version,** then point to the body and press the yellow button.

Changing Pin States

The **bubble** command allows you to change 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, then point to the pin and press the yellow button.

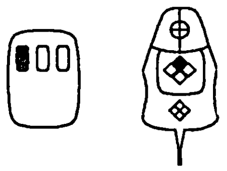
Positioning Bodies

GED also supports several commands that allow you to position and rotate bodies to meet your drawing requirements.

mirror	Creates a mirror image of the selected body about the Y axis.
rotate	Rotates a body 90° each time you press the button, with mirror images of the body at 180° and 270°.
spin	Rotates the body 90° each time you press the cursor button.

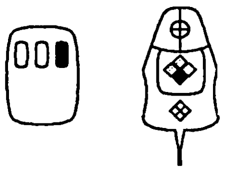
Drawing Wires

The **wire** command is used to draw lines to connect the components of your schematic. This command is used with the buttons to begin, position, bend, and attach wires where required on the schematic.



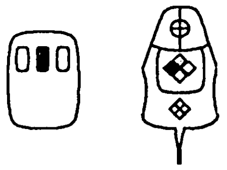
Yellow Cursor Controller Button

Starts and attaches the wire at the nearest grid point. This button is also used to place additional bends in a wire.



Blue Cursor Controller Button

Starts and attaches the wire at the nearest appropriate vertex. The wire is not attached to vertices of text strings.



White (Green) Cursor Controller Button

Toggles between orthogonal and diagonal (direct) wire modes.

Press the yellow button twice to end a wire not attached to a pin or wire. Wires are automatically terminated at pins and wire junctions.

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

Figure 3–2 illustrates the most common wire shapes and the buttons used to draw them.

The following conventions are used in the chart:

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

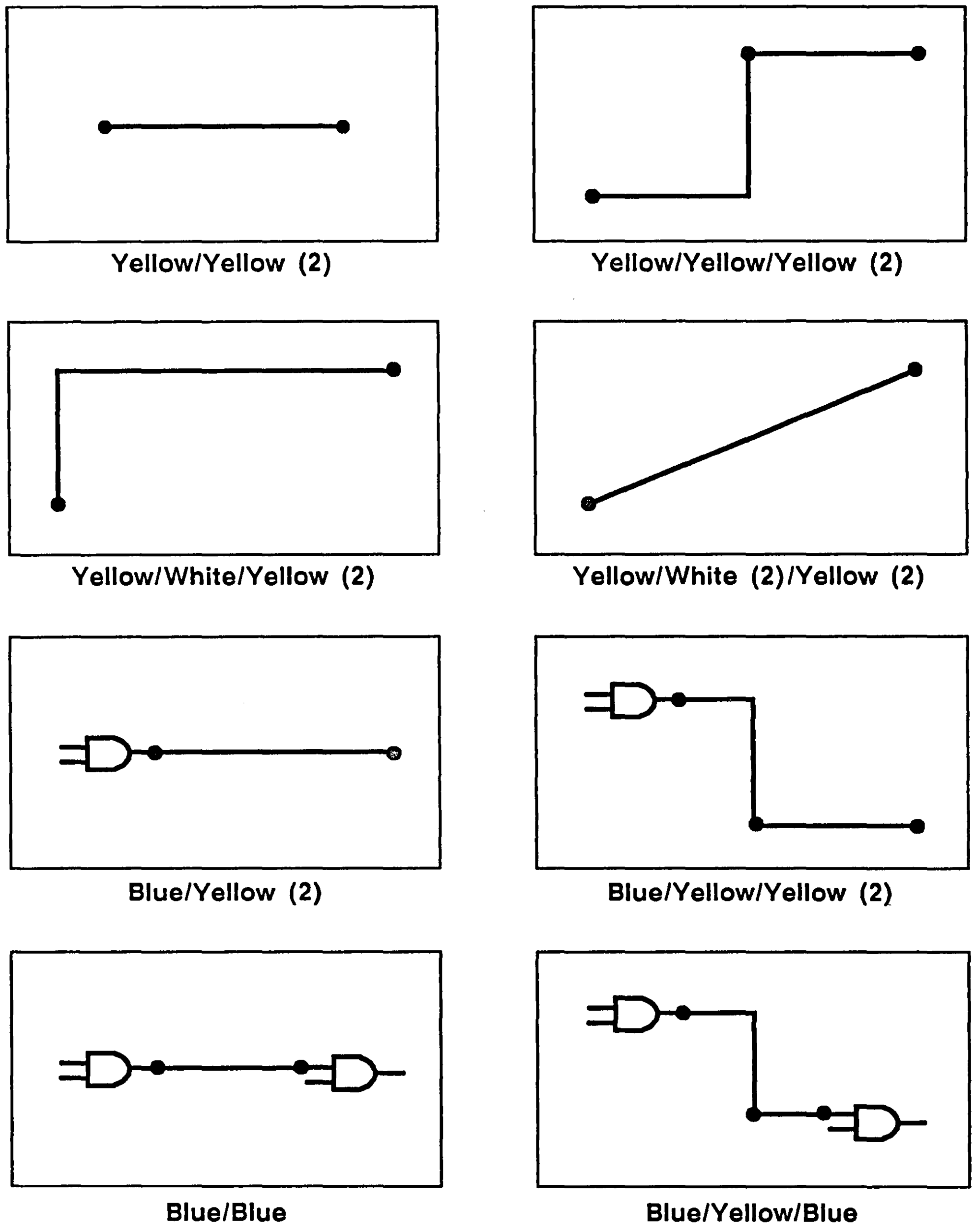


Figure 3–2. Wiring Reference Chart

The Route Command

You can use the **route** command to wire components together automatically. The **route** command connects two points by drawing a wire between them. If it cannot draw horizontal or vertical lines, it draws a diagonal line. **route** will not run a wire through any existing objects.

The **route** command is simple to use.

1. Enter the **route** command or select the **route** command from the menu.
2. Select the first point. A flexible line is attached to the cursor.
3. Attach the flexible line to the second point. **route** connects the two points with a wire.

   To select the nearest pin or wire vertex for a **route** point, use the blue button to select the point. Use any other button to select the nearest grid point.

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 are usually 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 the next flip-flop. Wiring with bus-through pins is shown in Figure 3-3.

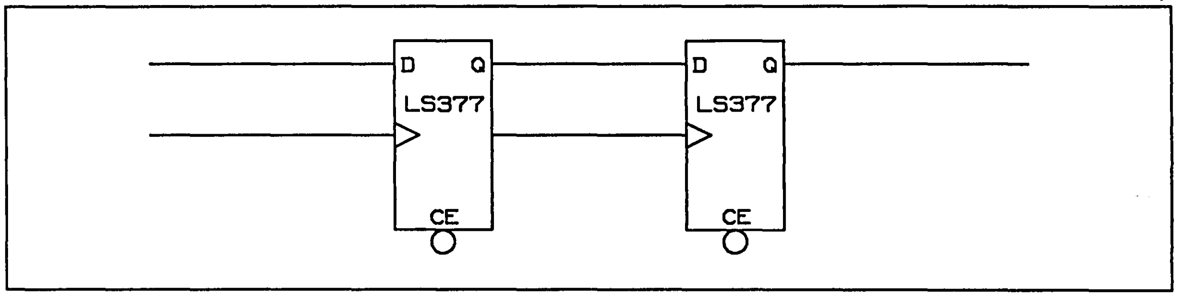


Figure 3-3. Wiring with Bus-through Pins

To connect a wire to a bus-through pin, use the **wire** command, position the cursor across from the input pin and press the blue 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.

You may use the **show vectors** command to display the pin names for the body. Bus-through pins have the same name as the corresponding real pin. The **show pins** command displays an asterisk on the side of the body where the extra pin is located.

Defining Signals and Connectivity

You can identify each input 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 multiple page 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 SCALD system, signals can represent a single bit (scalar signals) or multiple bits (vectored signals). The bit portion of the signal name specifies the number of bits (and which ones) the vectored 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.

Refer to the SCALD Language Reference Manual *for more information on signal syntax.*

The names you attach to the signals in the drawing are written into the *connectivity file* that GED creates when you save the drawing. A signal name is represented in the GED database as a property named SIG_NAME with a value equal to the name of the signal.

Naming Signals

To name the signals on a schematic, you use the **signame** or **busname** command. These commands allow you to add names to the signals and buses in the drawing and associate 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 wires.

Signame

The **signame** command allows you to type in several of the signal names for the drawing at one time. You then point to each wire in turn to attach the required signal name. As you place each signal name, the signal name is displayed attached to the nearest wire or pin, and 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. Next, type the signal names in order. The signal names are automatically placed at the indicated positions.

For example, the following procedure can be used to name the primary input and output signals of the schematic shown in Figure 3-4.

1 Select **signame** from the GED menu.

2 Type:

A Return

The 'A' is attached to the cursor.

3 Type the following signal names. Be sure to press Return after each name:

```
B
C IN
SUM
C OUT
```

4 Now move the cursor to the wire for signal A and click the yellow button once. The name appears near the wire.

5 Next move the cursor to signal B and click the yellow button.

6 Move to C IN and click.

7 Move to SUM and click.

8 Move to C OUT and click.

9 Select ; (semicolon) from the menu to end the **signame** command.

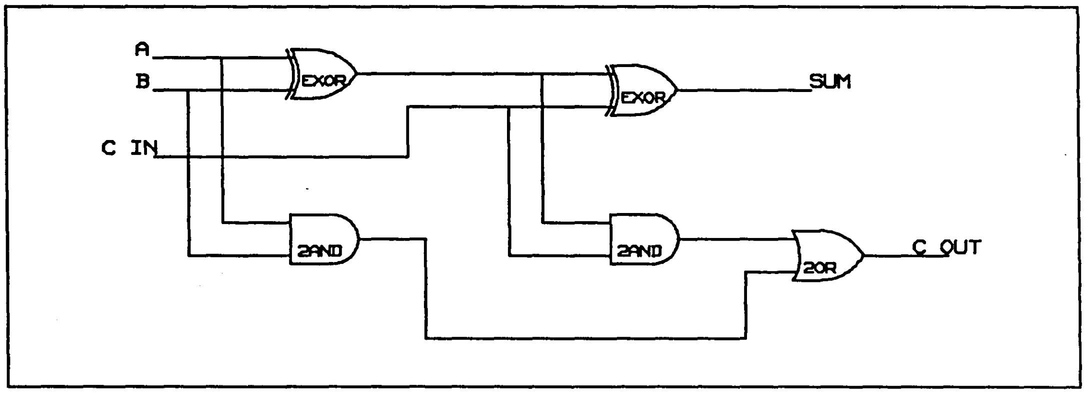


Figure 3-4. Sample Schematic with Signal Names

Busname

The **busname** command provides a convenient shorthand for naming signals in buses, or pins on bodies, whose names differ only in array subscripts. The name you specify is a simplified SCALD signal name such as:

ADDRESS<7..0>

GED reads the system-wide *config.dat* file to determine the array subscript string, such as two dots (..) or a colon (:). Bus bit ordering can be MSB ⧫ LSB or LSB ⧫ MSB. Table 3-1 illustrates bus name syntax and the resulting signal names.

Table 3-1. Bus Name Syntax

Bus Name	Signal Name
`A<3..0>`	`A<3>, A<2>, A<1>, A<0>`
`A<0..3>`	`A<0>, A<1>, A<2>, A<3>`
`A<0>`	`A<0>`
`A<7..0:2>`	`A<7>, A<5>, A<3>, A<1>`

If the array subscript character is a colon (:), the field separator becomes a double colon (::). For example:

A<0:7::2> becomes **A<0>, A<2>, A<4>, A<6>**

To use the **busname** command, enter the command with one of the following syntax options:

- **pt pt** *name*
- **pt** *name* **pt**
- *name* **pt pt**

The *name* is a simplified SCALD signal name, such as A<0..3>.

The two points specify the location of the first two names. The remaining names are placed automatically, with spacing between each name defined by the first two points.

GED draws a bright line between the name and the wire to which the name is attached to verify that the signal names are attached correctly.

For example, the following procedure can be used to name the primary input and output signals of the schematic shown in Figure 3–5.

1. Type:

   ```
   busname DATA<7..0>
   ```

   GED attaches the string `DATA<7>` to the cursor.

2. Place the name at the top wire and press the yellow button to enter the point.

   GED attaches the string `DATA<6>` to the cursor.

3. Position the second name and press the yellow button.

   This name and the remaining signal names are placed on the drawing.

Bright lines are displayed so you can verify the accurate attachment of signal names to the drawing.

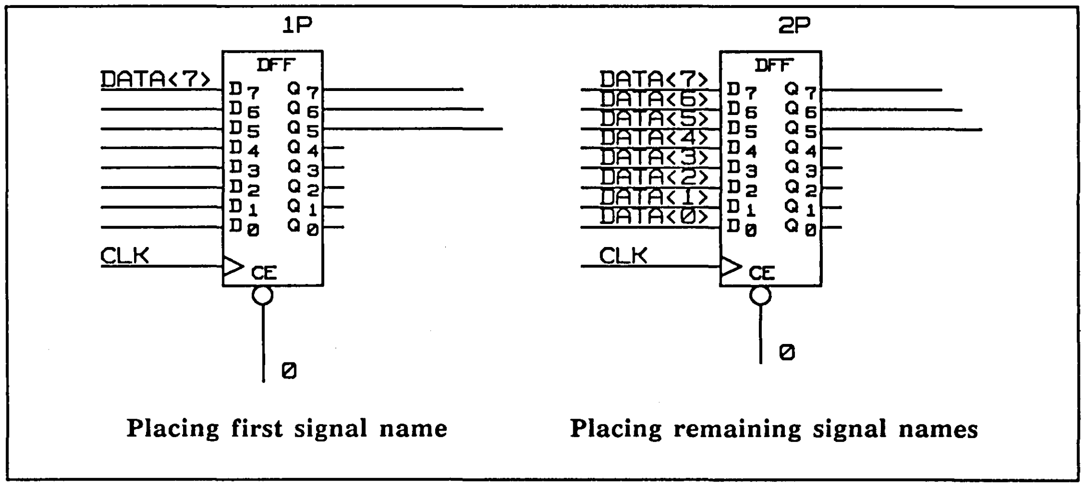


Figure 3-5. Using Busname Syntax

Adding Dots

You can use the **dot** command to place dots on the drawing to clearly mark the connection of two wires.

The system convention is that 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 connection points on BODY drawings.

To use the **dot** command:

1 Select **dot** from the menu.

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

Open dots are the default. If you want filled dots, first enter the command **set dots_filled.**

To automatically place dots on a complex circuit, you can use the following procedure with the **auto dots** command.

1 Type:

```
show connections [Return]
```

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 Type:

```
auto dots [Return]
```

All the junctions are automatically dotted.

You can automatically remove the dots from a drawing with the **auto undot** command. This command removes all the dots except those occurring at the intersections of four wires.

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 that does not have a matching NOT body. This allows you to catch errors when you intentionally connect a signal of one assertion to a pin of the opposite assertion. You 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 or pin.

To add a NOT body to a drawing:

1 Type:

```
add not [Return]
```

2 Place the NOT 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.

An example of using a NOT body is shown in Figure 3-6.

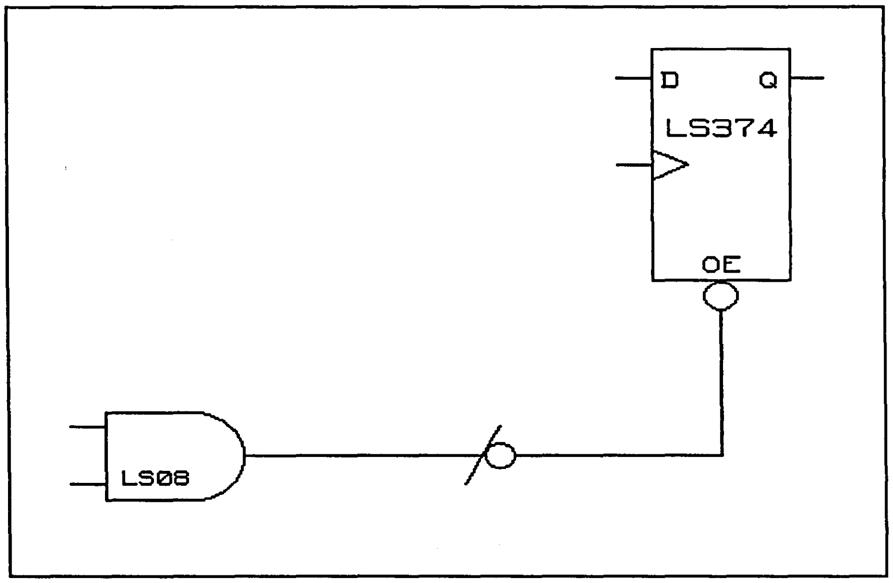


Figure 3-6. Using a NOT Body

See the VALIDCompiler Reference Manual *for more information on NOT bodies.*

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 a Compiler directive), the signals on either side of the NOT body are joined together and the NOT body is ignored.

Adding Properties

A *property* is used to convey information about a design. It consists of:

- A capitalized name
- A case-insensitive value

Properties can be attached to bodies, signals (wires), 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. GED does interpret the following properties:

- LAST_MODIFIED (on the DRAWING body)
- PIN_NAME (attached to connection points in BODY drawings.
- SIG_NAME
- Properties added by the **backannotate**, **section**, and **pinswap** commands.
- PATH (uniquely identifies parts)

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

GED uses PIN_NAME and SIG_NAME properties as part of its treatment of components. The information represented by the properties (except for back annotated properties) in a drawing is interpreted by the Compiler. This information is then passed to the other Valid design tools as well as to user-developed programs.

There are two ways of adding properties to a drawing:

- Use the **property** command
- Include a property in a signal name

The meaning of the property is the same regardless of the method used.

Tips on Adding Properties

- Attach properties to bodies (or groups of bodies) with the **property** command.
- Signal properties are usually included in the signal name (**signame** command), but can be added with the **property** command.
- Properties attached to pins 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 yellow button. An asterisk appears on the selected object.
3. Type the name of the property and the value of the property on the command line and press Return. Leave a space between the property name and the value. The value appears at the cursor.

   Alternately, you can separate the name and value with an equal sign (=) or by pressing Return between entries.
4. Move the cursor to position the property and press the yellow button.

By default, only the value is displayed when you add a property to a drawing. Use the **set prop_display** command to change the default display of properties. The **display** command allows you to change the way a property is displayed. There are several **display** command options that affect properties:

display name	Display only the name of the property.
display value	Display only the value of the property (default).
display both	Display both the name and the value (separated by an equal sign) of the selected property.
display invisible	Display neither the name nor the value of the selected property.

To change the display of a property:

1 Type:

```
display option
```

Substitute the actual value for *option*.

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

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

You can issue the **show** command to temporarily display information about the properties on the drawing. Two of the **show** command options are:

show properties

Display both the name and the value of all the properties (including signal names and invisible properties) on the drawing.

show attachments

Display the connections between visible 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 circles represent bubbled pins.

To draw a circle:

1 Select **circle** from the menu.

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

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

To draw an arc:

You can also use the **circle** *command to draw arcs. See the* **circle** *command description in the* ValidGED Command Reference Manual *for procedures.*

1 Enter the **arc** command and enter two points to indicate the ends of the arc.

   A circle is drawn with the two initial points indicating the end points of the arc. A flexible wire running between the initial points is attached to the cursor.

2 Position the arc as required and enter a third point to determine the curvature of the arc. The arc passes from the first point, through the third point, to the second point.

- Press the yellow button to position the curve of the arc at the nearest screen pixel.
- Press the white button to position the curve of the arc at the nearest grid intersection.

3 *OPTIONAL:* Select the semicolon (;) or the **arc** command instead of the third point to leave the circle on the drawing.

Adding Notes and Documentation

You add annotations and comments to your drawing with the **note** and **filenote** commands. The information you place on the drawing with these commands is not interpreted by the Compiler or by the other Valid analysis programs. The **note** command is useful when you want to add a single line of text to the drawing. The **filenote** command allows you to place the entire 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 ABBREV property allows you to specify an abbreviation for the drawing name. The SCALD system software 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 you specify should exactly match the GED drawing name.

To add a border around your drawing, you can add a PAGE body. Page bodies vary in size, and may include company logos. Two types of PAGE bodies available in the Valid Standard Library:

A SIZE PAGE	Places an A–size border (8 1/2 x 11 inches) around a drawing
B SIZE PAGE	Places a B–size border (11 x 17 inches) around a drawing.

Defining Groups

GED provides several facilities for defining groups of objects. In GED, a *group* is made up of all vertices that are affected when the group is defined.

For each defined group, 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 a name to a group by entering a single letter after you issue the **group** or **select** command. Do *not* enclose the single-letter name in quotation marks.

The Group Command

The **group** command allows you to draw a closed polygon around the required objects to identify a group. Use the yellow button to draw a line around the objects. Close the polygon by pressing the blue button when the cursor is near the starting point. In addition to drawing a polygon to group objects, you can simply press the blue button to include the nearest object in the same group. The option **all** places the entire drawing into a group.

The Select Command

The **select** command provides a stretchable rectangle to define the boundaries of a group. Use the cursor and press the yellow button to position the rectangle around the objects to be included in the group. You can also press the blue button to include the nearest object in the same group. The **all** option places the entire drawing into a group.

The Find Command

The **find** command allows you to group all the occurrences of a specified string or body. For example, **find** locates all the occurrences of a specified body name and places them into a group. You can then perform any group-manipulation commands (such as **paint, replace**, or **version**) on the group.

The Include Command

The **include** command allows you to add objects to a defined group. Specify the initial group by entering the group name on the same line as the **include** command; otherwise, the most recently created group is used. The starting group is highlighted. Press the yellow button to indicate objects to be included; press the white button to indicate groups to be included.

Several optional arguments allow you to **include** *types* of objects in a group. The objects are included based on their association with objects already in the group. You can include bodies, connections, other groups, nets, properties, and wires. You can also include the owners of properties or signal names already present in the group.

The Exclude Command

The **exclude** command allows you to remove selected objects from a defined group. Specify the required group by entering the group name on the same line as the **exclude** command; otherwise, the most recently created group is used. The starting group is highlighted. Press the yellow button to indicate objects to be removed; press the white button to indicate groups to be removed.

Several optional arguments allow you to exclude *types* of objects from a group. You can exclude bodies, connections, other groups, nets, properties, or wires. You can also exclude the owners of properties or signal names in the group.

After the group is defined you can use the **copy**, **cut**, **paste**, **delete**, **change**, **smash**, **display**, **move**, **paint**, **replace**, and **version** commands to manipulate the group. To perform an editing operation on a group, select the group by typing the group name or by pressing the white or green button when the cursor is near the group.

Using Color

The **paint** command allows you to specify the colors of the objects and groups in your drawing. Even if you are working on a monochrome monitor, you can add color to drawings that can be printed on a color plotter or transferred to a color workstation.

You can use up to 16 colors in your designs. They are preset colors and cannot be changed:

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

You can assign a color to a defined group by including the color name and group name on the **paint** command line or by entering the **paint** command, selecting the color from the menu, and entering a single-letter group name.

There is also a **default** option to the paint command. This paints objects in their preset default colors. Use the **set** command to establish default colors for the objects in your drawings.

If you have a color monitor, the objects are drawn in the actual colors you specify. If you use a monochrome monitor, you can use the **show color** command to display the names of the colors assigned to the objects in your drawing.

The **window** and **zoom** commands can be nested within the **paint** command.

When you issue the **paint** command, the status line 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 yellow button.

Making Duplicates

You can copy 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 **copy** from the menu.

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

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

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

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

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

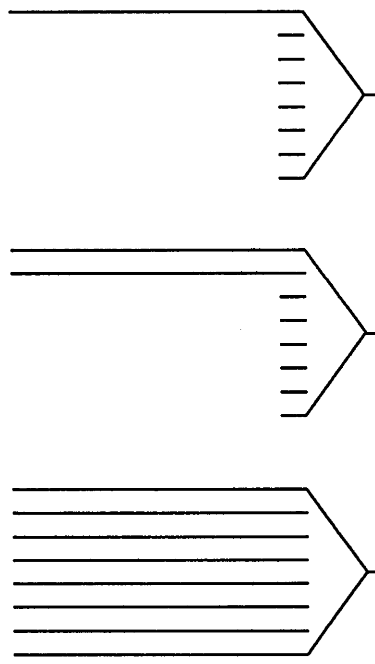
   - The blue button attaches the copy to the vertex nearest the cursor. This is useful for attaching copies of wires at new locations.

To make a copy of a group, use the white or green 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 yellow button to place the copy down on the drawing.

The **copy** command also accepts an argument to specify the number of copies to be made. After the first copy is placed, the remaining copies are automatically added to the drawing. The second copy is offset from the first copy by the same distance as the first copy from the original. You can use this feature to copy single items and groups. Use the following procedure:

1. Issue the **copy** command and enter a number to specify the number of copies to be made.
2. Move the cursor to the object or group to be copied and press the appropriate button (yellow for objects, white for groups).
3. Move the copy to its location and press the appropriate button (yellow to place the copy on the nearest grid point, blue to attach the copy to the nearest vertex).

Figure 3–7 shows how this command can be used to attach wires to an 8–MERGE body.



1. Wire the first connection.
2. Enter the command **copy 7**.
3. Move the copy to the first wire and press the yellow button.
4. Position the cursor at the next pin and press the yellow button.
The remaining six wires are placed on the drawing.

Figure 3–7. Making Multiple Copies

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 bodies. However, pin properties, such as pin numbers, are not copied. Wire properties are not automatically included when you copy a wire. Unnamed signals and PATH properties are not copied, since GED creates them automatically.

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 lets you attach the property to its new location:

1. Select **copy** from the menu.

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

3. Use the cursor to position the copy and press the yellow button to position the property on the drawing. A flexible 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 yellow button.

To copy objects or groups from one drawing to another drawing, 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 bodies. 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 new drawing.

2 Issue the **paste** command. A copy of the **cut** buffer is attached to the cursor.

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

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 uses orthogonal wires. The **set** command is issued during an editing session to change a command's default behavior, while the **display** command is used to change the drawing. Additionally, the **set** command can be placed in your *startup.ged* file. Both commands have many options to allow you to tailor GED to your particular preferences and requirements.

Refer to the discussion of the **display** *command in the* ValidGED Command Reference Manual *for more information.*

The **display** command is used to change 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. (The **show** command has only temporary effects on the drawing.)

The **set** command allows you to change the default behavior of many GED commands. If you issue the **set** command during an editing session, you change the value for that entire session. For example, to increase the default size of text, 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 the* ValidGED Command Reference Manual *for details.*

set commands can also be placed in the system-wide *startup.ged* file to establish system-wide default values.

Editing Text on a Drawing

You can use the **change** command to edit the text on your drawing. This command allows you to change property names and values, 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 (although "soft" properties can be changed). GED gives an error message if an illegal operation is attempted.

To use the line editor:

1. Select **change** from the menu.
2. Point to each text string to be changed and press the yellow cursor button. You can also define groups of text strings and then select the group with the white button or by entering the group name before you select more strings.

   The text string being edited is displayed on the status line.
3. Use the line editing functions to modify the text (see Table 3–2). When you make a change and then press Return, the text string is repositioned on the drawing and the next string is displayed on the edit line for modification.

Table 3–2. Line Editor Functions

Keys	Result
Control – **F**	Moves the cursor forward one character.
Control – **B**	Moves the cursor backward 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 – **H**	Deletes one character to the left of the cursor.
Control – **D**	Deletes one character to the right of the cursor.
Control – **K**	Deletes the remainder of the line (right of the cursor).
Control – **Q**	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. When all text has been edited, exits the line editor.
Control – **S** *character* Return	Searches to the right of the cursor for the specified character.
Control – **R** *character* Return	Searches to the left of the cursor for the specified character.
Control – **U** *number* Return *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.

To use the system text editor:

1. Enter the **change** command.
2. Select the text strings with the cursor.
3. Press Control-V. This accesses the system editor.

   UNIX systems use the vi *editor; VMS systems use the* EDT *editor.*

   A file containing the text strings you selected is displayed. Edit the file to make the required changes. Be sure not to add or delete any lines in the file. Refer to the appropriate manual for information about using the editor on your system. When you end the editing session, the drawing is redisplayed with the modified text.

   You can change the default system editor with the **set user_editor** *command.*

Searching For Patterns

The **find** and **next** commands allow you to locate all 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 objects matching the pattern into a group. GED labels the group and lists the number of drawing elements it contains. The **next** command displays an asterisk at each item so it can be changed or deleted. If necessary, items are also centered on the screen. Because all the items are placed in a group, you can perform an operation such as **replace**, **version**, **change**, **display**, or **delete** on that group to make a global change to the drawing.

To specify a search pattern:

1 Type:

```
find pattern
```

The **find** command is not case-sensitive; it does not distinguish between uppercase and lowercase letters. You can use wildcard characters in the pattern. An asterisk (*) matches anything, and a question mark (?) matches any single character. An equal sign (=) implies a search for properties; separate patterns for the name and value are accepted.

2 Type:

```
next Return
```

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**, **reattach**, or **pinswap**, depending on the particular application.

The Move Command

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 dragging*.

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 **move** from the menu.
2. Position the cursor at the object to be moved and press the appropriate button.
   - The yellow button picks up the object at the grid point nearest the cursor.
   - The blue button picks up the object at the nearest vertex. (Useful for copying bodies.)
   - The white or green buttons pick 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 yellow button places the object on the grid point nearest the cursor.
   - The blue button attaches the object to the nearest vertex.

The Split Command

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.

To use the **split** command:

1. Select **split** from the menu.
2. Point to the objects and press the blue 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 blue button again. You can continue to select objects at that vertex until the correct object 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. Select **split** and identify a point along the wire using the yellow button. This adds a vertex at the specified point.
2. Move the vertex to its new location and press the yellow button.
3. Use the **wire** command to add a new section of wire.

The Reattach Command

The **reattach** command reattaches properties (including signal names) from one object to another.

To use the **reattach** command:

1 Type:

```
reattach [Return]
```

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.

The Swap Command

Use the **swap** command to interchange the positions of two properties or two notes. Only two properties or two notes can be exchanged, not a note and a property.

To use the **swap** command:

1 Enter the **swap** command.

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

The **pinswap** command works just like the **swap** command. It is used to exchange the pin numbers assigned to a body by the **section** command.

Deleting Objects

The **delete** command removes unwanted objects and text from the drawing. You can also use the **delete** command to delete specified groups of objects.

To use the **delete** command:

1. Select **delete** from the menu.
2. Point to the object to be deleted and press the yellow button. The object nearest the cursor is deleted.

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

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

Viewing the Drawing

GED manages drawings that can be as large as 64 inches on each side if plotted in a single drawing page. When this 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 the 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 screen display.

GED provides **window** and **zoom** commands to let you zoom in on part of a drawing, zoom out, pan to different areas, and center the screen around a specified point. You can also reduce and enlarge 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, you issue the **zoom** or **window** command and then either specify a point to be used as the new center of the viewing area, or you type in one of the **zoom** or **window** command options that specifies a panning direction: **left**, **right**, **up**, or **down**. The drawing remains at the current size, but you see a different view of it.

To pan a drawing using a specified point:

1. Enter the **zoom** command.
2. Move the cursor to the point on the screen to be centered and press the blue button.

You can also specify a panning point with the **window** command:

1. Select **window** from the menu.
2. Move the cursor to the place on the drawing to be centered on the screen.
3. Press the yellow button. An asterisk appears on the screen.
4. Select the semicolon (;) from the menu or type:

   ; Return

   The part of the drawing where you placed 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 let you see more detail. It is especially useful for checking the wiring and connections on large drawings. You can use either the **zoom** or the **window** command to zoom in on part of the drawing.

The **zoom** command allows you to specify a rectangle defining an area of the drawing to be enlarged. To zoom in on a portion of the drawing:

1. Issue the **zoom** command.
2. Press the yellow button to indicate one corner of the rectangle.
3. Move the cursor to the opposite corner and press the yellow button.

   The area of the drawing defined by the rectangle is enlarged to fill the screen. You can draw additional rectangles to further zoom in on the design.

To use the **window** command to enlarge a portion of the drawing, follow 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 yellow button.
4. Select the semicolon (;) from the menu, or type:

   ```
   ; Return
   ```

   The selected area enlarges to fill the entire screen.

You can issue additional **window** commands to further zoom in on the design. Another version of the **window** command allows you to zoom in and center the display simultaneously.

1 Select **window** from the menu.

Refer to Figure 3–8. *This first point is labeled C (for center).*

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.

The second point is labeled "2" in the figure.

3 Move the cursor to the right an inch (or so) and press any button. A second asterisk appears.

This third point is labeled "3" in the figure.

4 Move the cursor to the right for an equal distance (about an inch) and press any button.

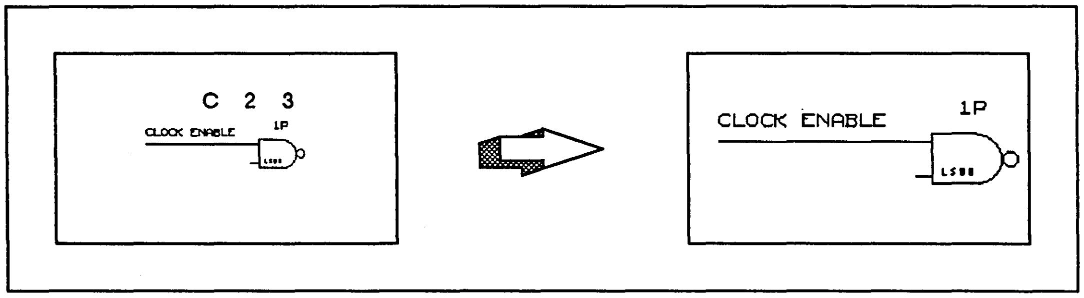


Figure 3–8. Zooming In and Enlarging

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 point C and point 3 is twice as far as the distance between point C 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 screen size of the selected portion of the drawing.

1 Select the **window** command from the menu.

Refer to Figure 3–9. *The first point is labeled C (for center).*

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.

The second point is labeled "2" in the figure.

3 Move the cursor to the right an inch (or so) and press any button. A second asterisk appears.

This third point is labeled "3" in the figure.

4 Move the cursor back to the left to a point about halfway between points 1 and 2 and press any button.

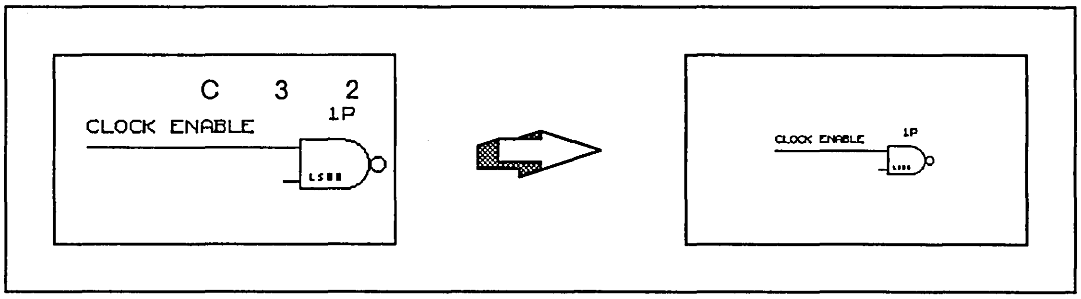


Figure 3–9. Zooming Out and Reducing

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 point C and point 3 is half as far as the distance between point C and point 2, the drawing size is reduced by half. You can vary the reduction factor by changing this ratio.

When you zoom out of the drawing, text that becomes too small to read is replaced by rectangles. The rectangles remind you that there is text at the indicated positions. If you zoom out far enough (depending on your workstation), the rectangles themselves may disappear entirely when the text is too small to be seen. When you enlarge the drawing, the actual text is redisplayed.

To use the **zoom** command to zoom out and reduce the drawing:

1. Issue the **zoom** command.
2. Press the yellow button to indicate one corner of the rectangle.
3. Move the cursor to the opposite corner and press the white button.

   The screen shows arrows pointing from the corners of the screen image to the corners of a rectangle into which the current screen is compressed.

   The graphics in the drawing area are shrunk down to fit into the sizeable box.

The size of this rectangle relative to the size of the GED window represents the amount of reduction used to display the design.

4 Press the yellow button.

The area of the drawing displayed on the screen is reduced to fit into the rectangle.

Scaling the Drawing

You can use the **window** or **zoom** 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:

zoom -2	Makes the drawing appear half as large.
window 2	Makes the drawing appear twice as large.
zoom 1.5	Enlarges the drawing one and a half times.
window 0.5	Has the same effect as **zoom -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 **fit** option to the **window** and **zoom** commands. This option scales the drawing to fit into the window area, providing you with a global picture of your design.

To fit the entire drawing into the window, type:

```
window fit [Return]
```

or

```
zoom fit [Return]
```

The drawing is enlarged or reduced to fit entirely on the screen.

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:

- 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
- Wires that come close to but do not contact pins
- Duplicate PATH properties
- Unmarked wire connections
- Wires overlapping a body
- Missing TITLE and/or ABBREV properties
- Bodies that are placeholders
- Pins located at the origin (0,0) in BODY drawings
- Multiple dots at the same location
- Hard properties with the ? value (placeholders)
- Objects partially outside the GED drawing boundaries
- Wires connecting the pins of a two-pin body.

The **check** command lists each detected error. After you run the **check** command, you use the **error** command to locate each error on the drawing.

The **error** command steps through the errors located by **check**, draws an asterisk at the location of the error, and displays a message describing the error. To use these commands:

1. Type:

   ```
   check [Return]
   ```

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

2. Type:

   ```
   error [Return]
   ```

   error draws an asterisk at the first error and displays 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.

The **set check_on_write** feature automatically initiates the **check** command every time you write a drawing. To toggle this feature off, enter:

```
set check_on_write off
```

This command can be placed in your *startup.ged* file or issued during the editing session.

Using GED Scripts

GED's **script** command allows you to create a file containing a list of GED commands (a *script*) and execute that list of commands in GED. This allows you to operate in batch mode using the same syntax as if you typed in the command.

Scripts can call other scripts; in fact, this is done in the standard initialization script *startup.ged*. Scripts can also be interactive. You can use the cursor to enter points, you can specify the X–Y coordinates in the script file, or you can include *user input tokens* to allow a script to request user input during operation.

User input tokens must be placed at the beginning of a new line. There are two user input tokens.

$<

When GED encounters this token in a script, it prints from the token to the end of the text line as a prompt, then waits for one item of input. The input can be a typed line, a function key press, a cursor controller point, or a Control–C; you cannot use a Return as a response to a user input request.

$;

This token also prints from the token to the end of the text line as a prompt and awaits input, but this token accepts and interprets input until you enter a semicolon. If this token is included, GED follows the prompt with the message:

```
Type ; when done with user input.
```

When GED sees a user input token in a script, it highlights a menu button with the name of the GED command being executed.

Some simple script examples are shown below.

EXAMPLES

```
add ls04
$<Place the LS04
```

Add an LS04 to a drawing and use the mouse to position the part.

```
property
$<Choose the part to add a size to
size =
$<Type in the size you want
$<Place the property on the drawing
```

Add a size property to a part with a size specified at the time of entry.

```
rotate
$;Rotate the object until properly oriented
```

Rotate an object until the user enters a semicolon.

A more complicated script might contain a large number of **signame** commands and prompt the user for a point to place each SIG_NAME property.

4 Design Techniques

This section introduces:

- Flat design techniques
- Structured design techniques
- Hierarchical design techniques

Depending on your particular requirements, one of these three techniques can best meet your needs:

Flat Designs

A flat design is an efficient method for creating a design that is small and does not re-use portions of the circuitry. Flat designs are required for complete backannotation of the design and are more convenient for troubleshooting. Flat designs can include multiple drawing pages.

Structured Designs

Structured designs 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

Hierarchical designs use symbolic representations of circuitry for functions that are repeated throughout a design. Large designs that can be broken into functional modules or designs that re-use 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. SCALD tools are specially designed to operate efficiently with structured and hierarchical techniques.

Additionally, the optional schematic flattener, ValidFLAT/Transcribe™, allows you to use structured or hierarchical design techniques and produce a conventional, flat drawing. This tool transforms a structured or hierarchical design into a flat drawing that shows every pin and part explicitly.

Flat Designs

The flat drawing method is the most straightforward technique for creating a design on the Valid system. In a flat design, all parts on the drawing come from Valid or user-defined libraries and are one-to-one logical representations of 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 re-use 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-page and multiple-page flat drawings can be created with GED and processed by the Valid design analysis programs.

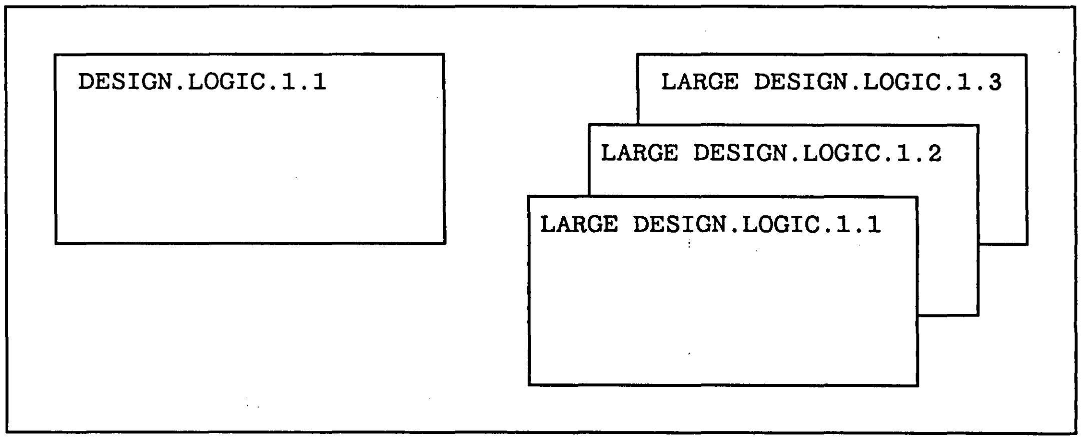


Figure 4–1. Single and Multiple Drawing Pages

Some designs are small enough to fit on one page of a drawing.

To create a single-page 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 SCALD analysis programs to compile, simulate, verify, and package the design.

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

1. Specify the drawing name with the **edit** command and create page 1 of the design.

2. Use the **write** command to save page 1.

3. Type:

   ```
   edit ...2
   ```

   to begin page 2 of the drawing.

   This command uses the current drawing name, the default drawing type, and the current version. The default drawing type (initially LOGIC) is changed with the

set push_type command. If necessary, enter the drawing type and version to specify the correct drawing name.

4 Use the **write** command to save page 2.

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

All pages of a multi-page 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 subsequent pages. 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 with both instances 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 are 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

documented. 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 layout the design.
- Troubleshooting Compiler errors in a large, multi-page flat design is time consuming and difficult.

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 on the schematic.

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 is 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. Version 1 is a one-bit slice of the part. It accepts a one-bit D input and produces a one-bit Q output. Version 2 is the full chip representation of the LS374 with all eight input and output bits explicitly shown. The two LS374 versions are shown in Figure 4-2.

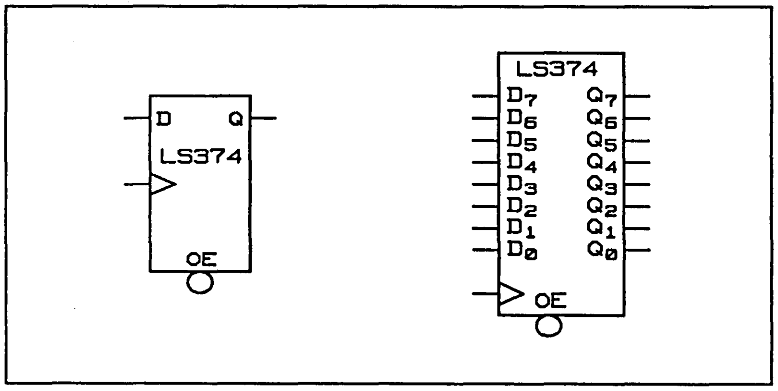


Figure 4-2. LS374 Body — Versions 1 and 2

Version 1 is *sizeable*; that is, you can specify the number of bits the part can represent. Library parts are generally developed with version 1 sizeable. The **show vectors** command displays the pin names of a selected part, allowing you to verify that a part is sizeable.

See the SCALD Language Reference Manual *for more information about signal syntax.*

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.

Figure 4–3 illustrates how you can use version 1 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.

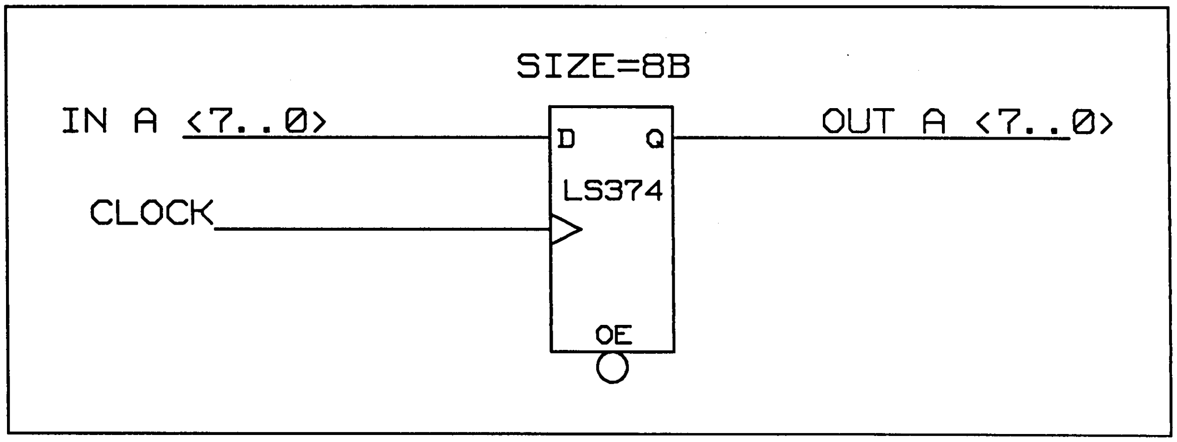


Figure 4–3. Using the SIZE Property to Structure LS374

Version 2 of LS374 (see Figure 4-4) is the flat representation of the part. Each pin on the drawing represents a pin in the physical package.

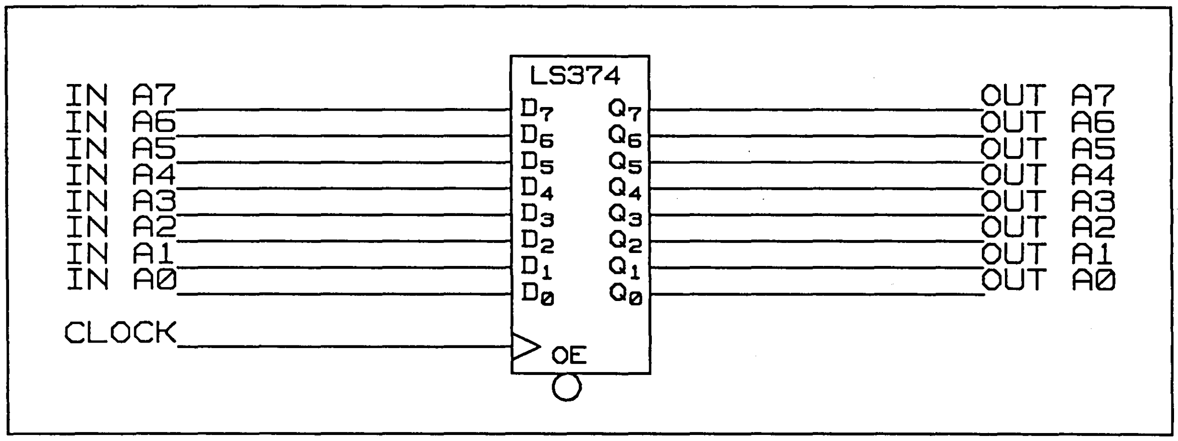


Figure 4-4. Using Version 2 of LS374

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.

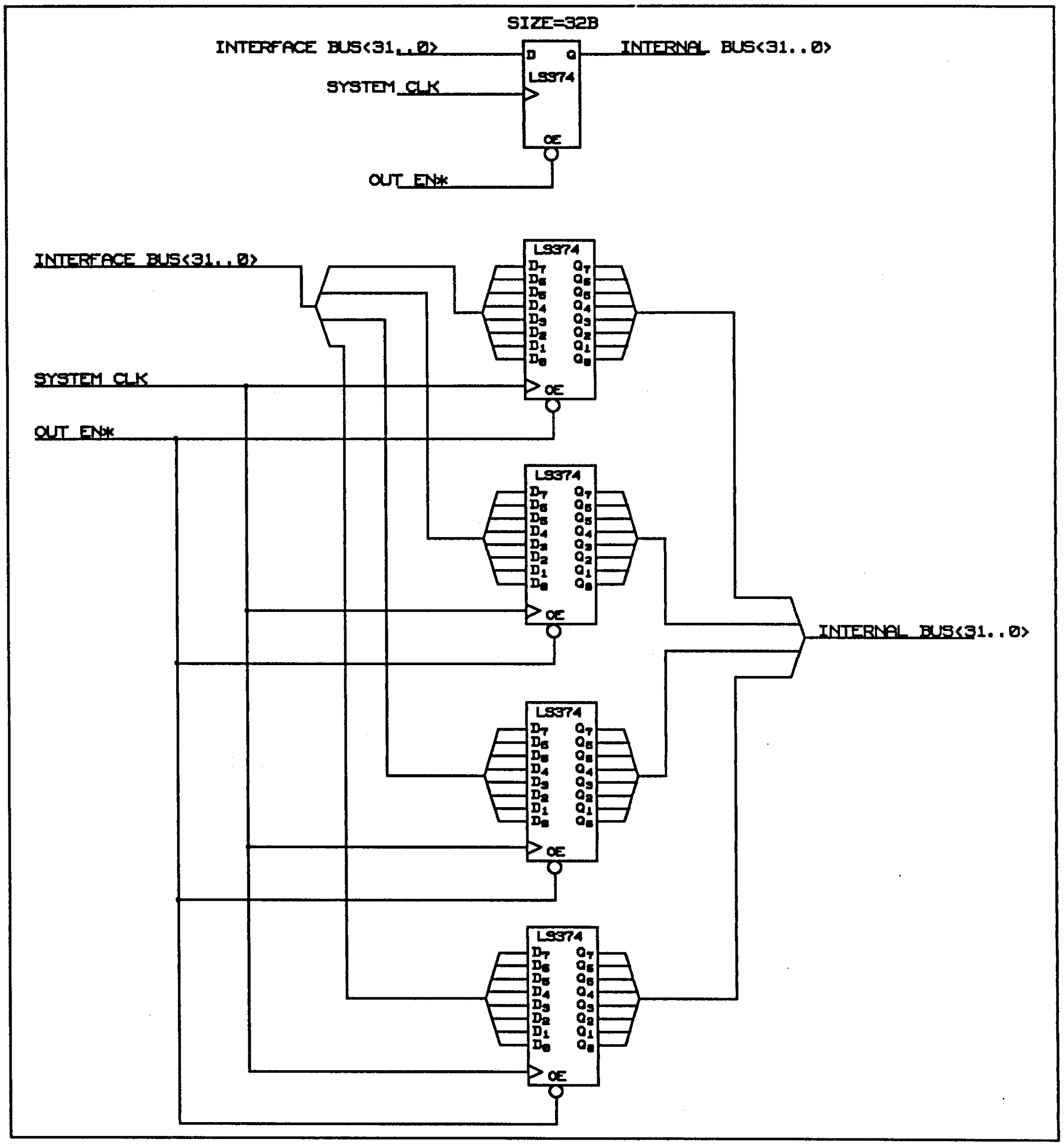


Figure 4-5. Structured and Flat Design Techniques

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.

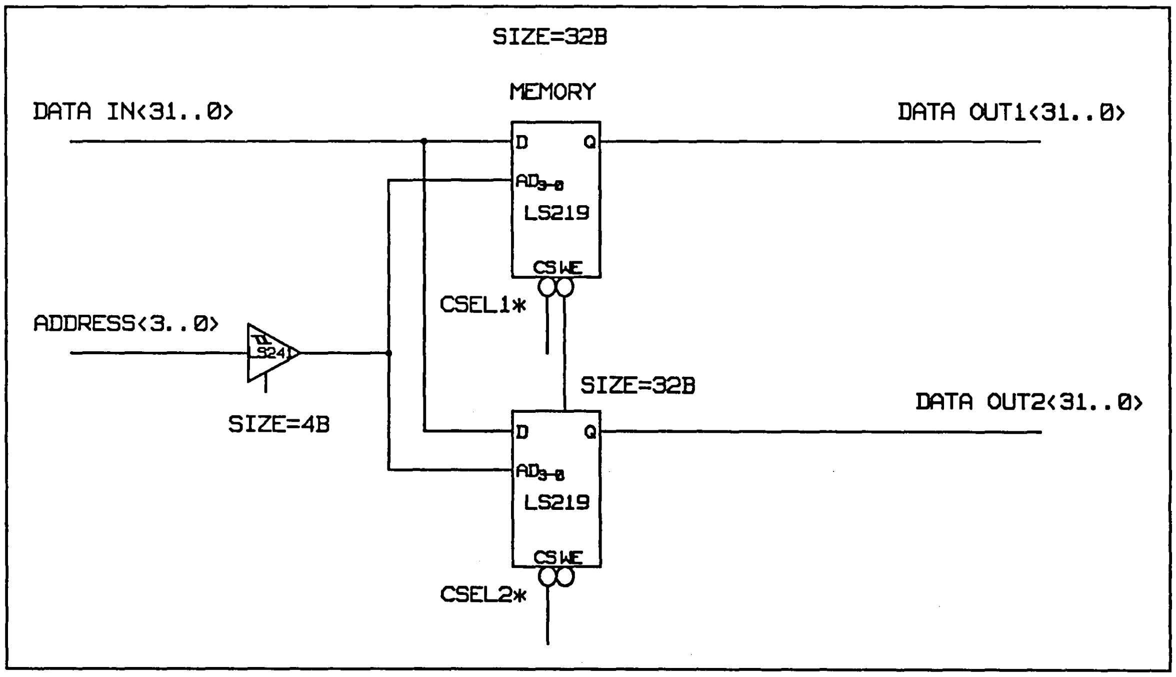


Figure 4-6. Structured Design with the SIZE Property

In this design, the 4-bit three-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.

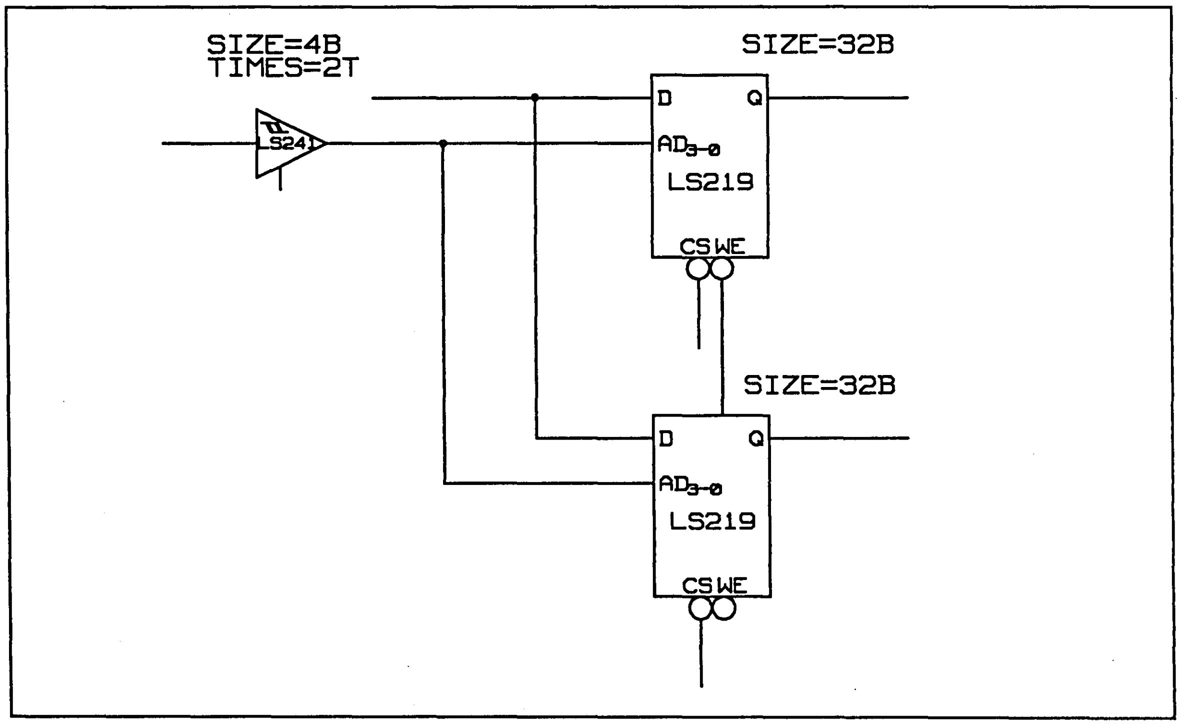


Figure 4-7. Using the TIMES Property

In this example, the TIMES property tells the system that two instances of a 4-bit three-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.

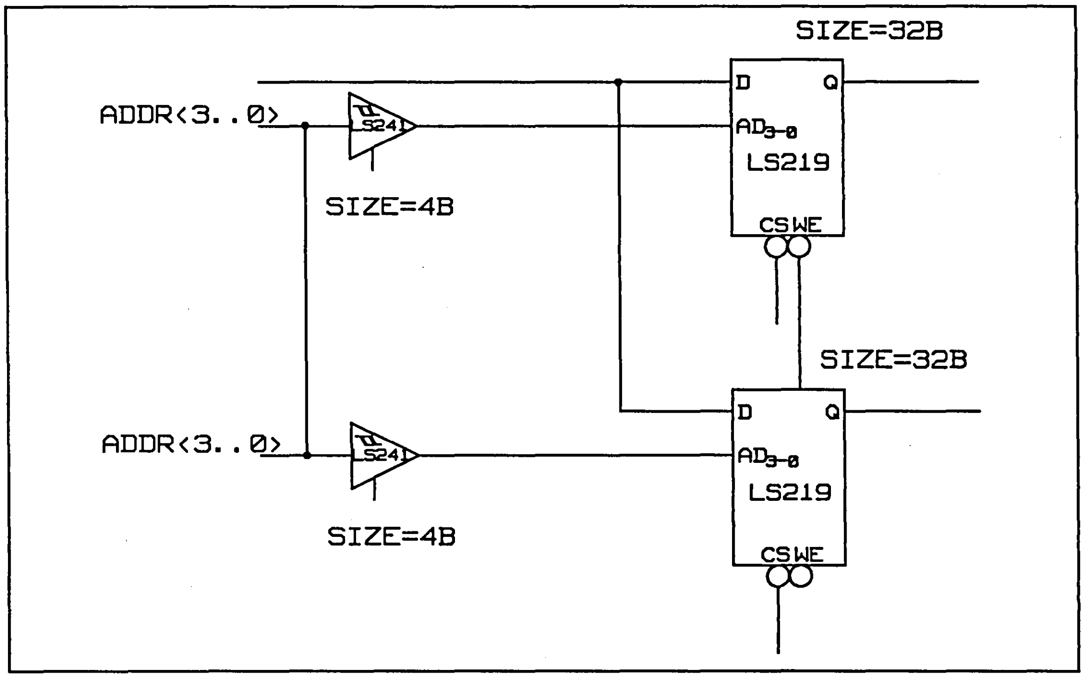


Figure 4-8. Manually Balancing Loads

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 your search list of SCALD 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.

See the Library Reference Manual *for illustrations and descriptions of the bodies in the Standard Library.*

The library contains merge bodies for merging signals, and tap bodies for tapping bits from buses, as well as several other special parts.

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 entered into GED 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, as shown in Figure 4–9.

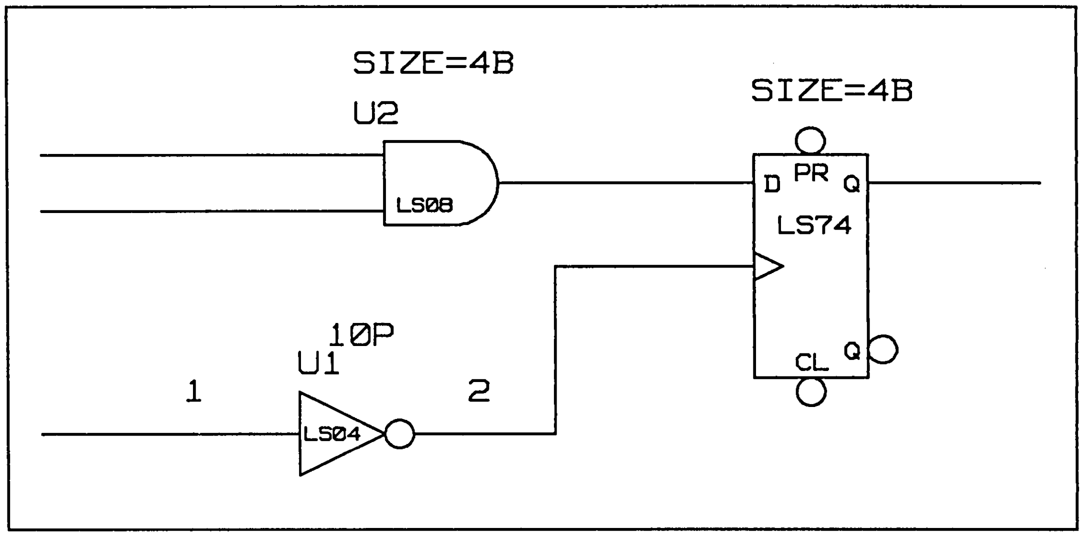


Figure 4–9. Backannotation of a Structured Drawing

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 an 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 be educated in how to read structured print sets and how to reference the physical information.

For those designers who create a structured or hierarchical drawing but want to produce a conventional flat drawing, the optional ValidFLAT/Transcribe Schematic Flattener is available. This tool transforms a structured and/or hierarchical design into a flat drawing that shows every pin and part explicitly.

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 SCALD 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 you can include in a hierarchical design. Figure 4–10 shows the hierarchical levels of the computer.

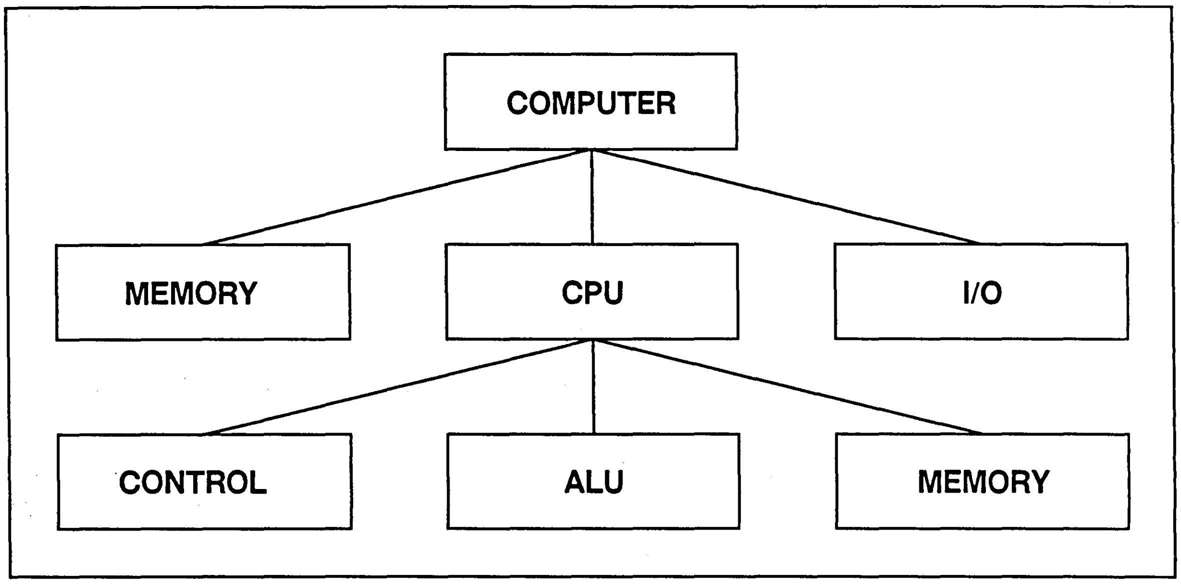


Figure 4–10. Levels of Hierarchy

After the modules of the design are planned, implementing the design uses 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. 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.

3 Create a BODY drawing to represent the design module.

4 Create a new LOGIC drawing and add the required number of BODY representations to it, building a circuit using the modules. You have added a symbol that represents the functional module you created in Step 1. The BODY drawing acts as a pointer to the LOGIC definition of the circuit.

5 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.

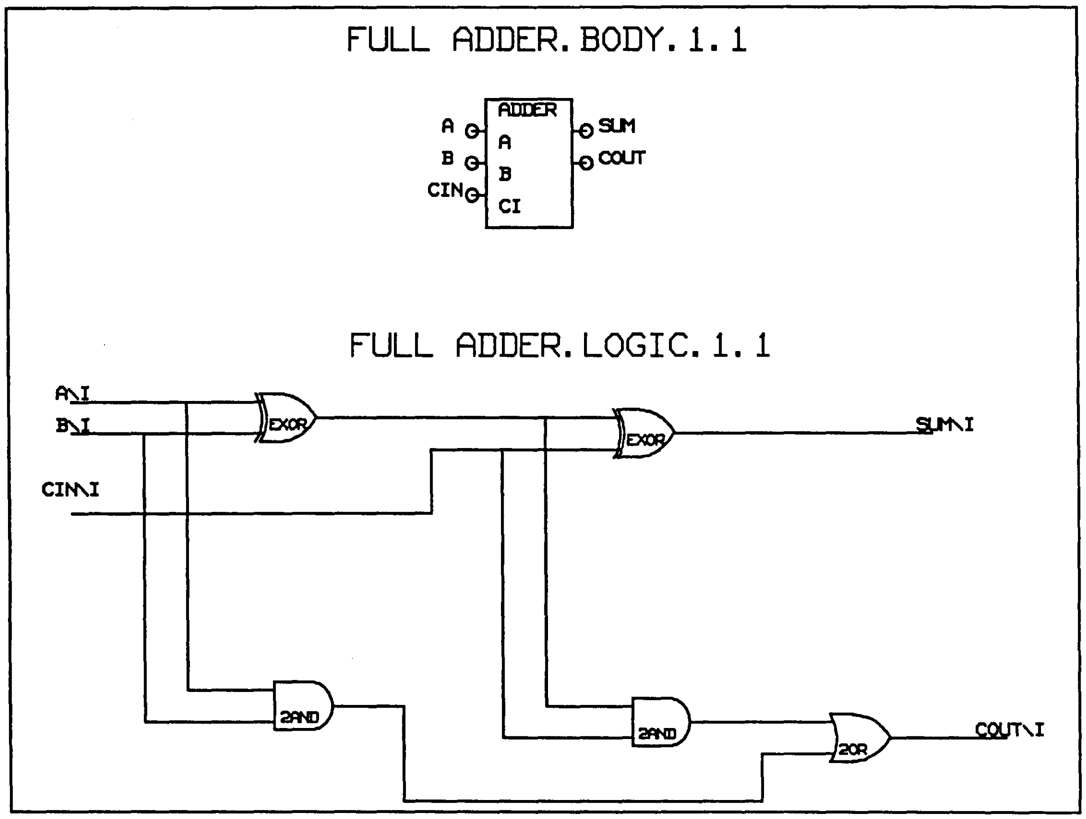


Figure 4-11. Full Adder Logic and Body Drawings

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

When you create a hierarchical design, you draw simple shapes (bodies) to represent the specific logic for each element of the design. GED provides you with the tools for drawing bodies and establishing the relationships between the body drawings and the logic drawings they represent.

For more information on creating bodies, see the Library Reference Manual.

The pins on the bodies that correspond to signals in the logic drawing must have the same name. Additionally, these signals in the LOGIC drawing are given the interface signal property (\I). This signal property is used to indicate an interface signal from a higher level drawing.

For more information on signal syntax, see the SCALD Language Reference Manual.

Follow these steps to use GED to create a body drawing:

1 Edit a drawing with the BODY extension.

A grid is displayed, with an X to mark the origin of the body. The grid for BODY drawings is twice as fine as the value set in by **set default_grid**. The initial system default grid for BODY drawings is 0.05 inches.

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. The origin should not be at a connection point (pin end) for the body.

3 Use the **wire** command to build the outline of the body around the origin body. The grid is

used as a guideline for the appropriate size and shape of the body.

See page 4–23 for information about defining low-asserted pins.

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 a dot at the end of each pin. Dots should be placed on displayed grid intersection points. Press the blue button to ensure that the dot is properly placed at the end of the wire.

6 Use the **signame** command to add pin names (corresponding to the signal names in the related logic drawing) to each pin. The name must match the corresponding name in the logic drawing exactly except for the omission 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.

Use the **wire** *command and press the white button to display diagonal lines.*

8 Mark the clock signal with a wedge.

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. You can use either the **circle** or **arc** command to add circles. The circle should be 0.1 inch in diameter. A dot is placed on the appropriate grid intersection point on the circumference of the circle to mark the connection point. The signal name should also be low-asserted (*). Figure 4-12 shows an example of using a low-asserted pin.

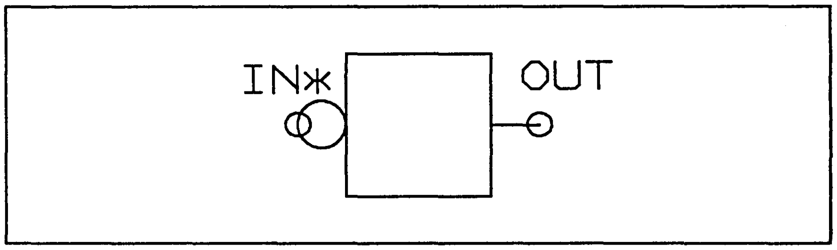


Figure 4-12. Using a Low-Asserted Pin

To define pins 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 so that both representations are available. Be sure to place a dot at the connection point. You also attach the BUBBLE property to the origin of the body to define which pins are bubbled when the part is added to a drawing.

See the Library Reference Manual *for more information about the BUBBLE_GROUP property.*

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.

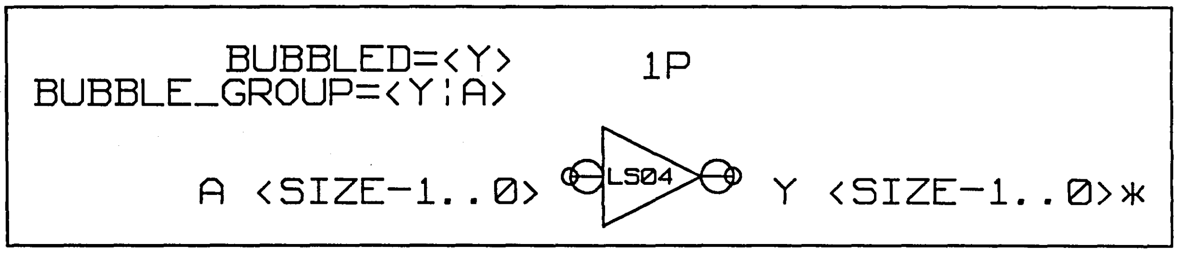


Figure 4-13. Using Bubbleable Pins

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

Creating Body Versions

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

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.2.1
```

Multiple versions can be useful when different sizes of the body are required on the logic drawing or when graphically different, but functionally equivalent, 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.

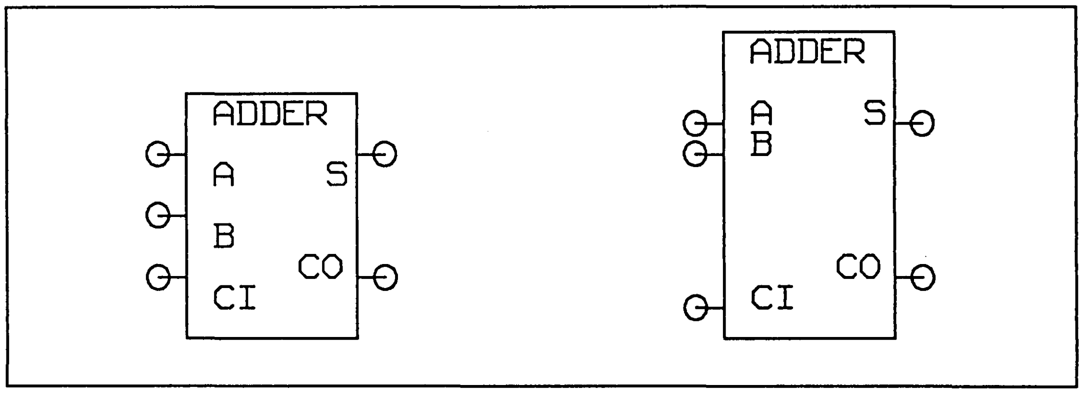


Figure 4–14. Versions 1 and 2 of FULL ADDER.BODY

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 a hierarchical design, 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 backannotated.

For those designers who create a structured or hierarchical drawing but want to produce a conventional flat drawing, the optional ValidFLAT/Transcribe Schematic Flattener is available. This tool transforms a structured and/or hierarchical design into a flat drawing that shows every pin and part explicitly.

The cross-reference listings generated by the Packager contain physical information for every part 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 in the cross-reference listings.

Comparing Design Techniques

The design methodologies discussed in this section (flat, structured, and hierarchical) are all appropriate for solving different of design problems. You must weigh the benefits and considerations of each technique before deciding which method to use.

In addition to using the individual methods, there is no restriction against combining them in design drawings. 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 re-use modules Designs that do not use buses
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 re-use modules Large designs that can be organized into separate components
Benefits:	Shortened design cycle Fewer data entry errors Easy-to-read print sets Cross reference listings Effective debugging capability Print sets organized in logical (top down) flow of design
Considerations:	Partially back annotated print sets Additional training required for design troubleshooters

5 Producing a Hardcopy

This section explains:

- Using the **hardcopy** command within GED
- Producing a hardcopy drawing from the operating system
- Creating plot files

The **hardcopy** command sends drawings to a plotter to produce a hardcopy of a drawing.

Use the **hardcopy** command to:

- Send a drawing to a plotter attached to your system.
- Send a drawing to a plotter attached to another system.
- Create a plot file that can be printed at a later time.

Several brands and sizes of plotters are supported. Although direct connections to all plotters are not supported on all platforms, network facilities allow you to do remote plotting.

Using the Hardcopy Command

See Creating Plot Files, *page 5–5, for information about creating plot files that can be printed at a later time or transported physically to another system.*

The **hardcopy** command plots files on the specified printer as long as your system has been properly configured. Refer to the appropriate *Guide to Operations* or *System Manager's Manual* for more information.

If necessary, issue the **set local_plot** command (default). This command automatically plots the drawing on the correct plotter. The plotter can be either directly connected to your workstation or on the network, as long as your system is configured properly.

Issue the **set** command to specify the device where the drawings are to be sent. The **set** command can be issued during the editing session or placed in the *startup.ged* file.

For example:

set plotter w11versatec	Selects the 11–inch versatec plotter.
set plotter b9429	Selects a Benson plotter.
set plotter calcomp1043	Selects a Calcomp plotter.
set plotter hp7475 *or* **set plotter hp7580**	Selects the specified Hewlett–Packard plotter.

Refer to the *Guide to Operations* manual for your system for a listing of supported plotters. Refer to the *ValidGED Command Reference Manual* for more information on the **set** command.

After you tell the system which plotter is being used, the **hardcopy** command requires a *scale factor* entry and a *drawing name*.

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 default size. If you specify a drawing name, you MUST specify a scale factor (a real number or A - E).
drawing name	If no *drawing name* is given, the current drawing is plotted. 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:

hardcopy Return	Plots the current drawing at the drawing's default scale.
hardcopy a	Plots the current drawing on an A size page.
hardcopy c *.logic*	Plots all LOGIC drawings in the current directory on C size pages.
hardcopy .5 hyper mux	Plots all drawing types for the drawing "hyper mux" (BODY, LOGIC, SIM, etc.) at half size.

Creating Plot Files

GED provides facilities which allow you to create a plot file that can be printed at a later time, physically transported to another system, or transferred to to the Interleaf™ publishing system for incorporation into a document. This plot file contains all the graphic information about the drawing.

To create a plot file:

1. Use the **set** command to change to **spooled_plot** mode from the default **local_plot** mode.
2. Use the **set plot** *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 named *vw.spool,* which is specific to the plotter you defined with the **set** command. You can then transfer the file by diskette, tape, or over the network to a system that supports the specified plotter.

See the appropriate Guide to Operations *or* System Manager's Manual *for more information about network plotting with your workstation.*

On a SCALDsystem or Sun workstation, you can use either the *vpr* or *vpl* command to plot the *vw.spool* file on a local or remote Versatec plotter. Your system must be configured properly for the command to work.

To plot the *vw.spool* file, type the command:

```
vpr vw.spool
```

If you want to include a GED drawing in an Interleaf document, follow these steps:

1. Create a *vw.spool* plot file.
2. Transfer the *vw.spool* file to an Interleaf desktop. It appears on the desktop as an ASCII file.
3. Open the *vw.spool* image, copy it, and place it into an Interleaf document.

   The GED drawing is in vector format and is fully-editable by the Interleaf software.

Standalone Hardcopy

To produce hardcopies of GED drawings directly from the operating system (without entering GED), you use the *hpf* command.

SYNTAX

```
hpf [−option outputfile] [-2 otherinputfile] inputfile
```

−option	Can be either **−f**, **−a**, or **−v**.
−f	Writes the data to a new version of *outputfile*.
−a	Appends data to an existing *outputfile*; if *outputfile* does not exist, creates a new file.
−v	Writes a vector format file to a new version of *outputfile*.
outputfile	On the VMS operating system, if *outputfile* is not specified, output is written to the file P.SPL, and a DCL command file (see the *Guide to Operations*) is used to send the file to a plotter. If the *outputfile* is specified as a dash (—), output is written to SYS$OUTPUT. The dash output file can be used with the **-f** and **-a** options. On UNIX, if no *outputfile* is specified, *hpf* pipes the output to lpr.
otherinputfile inputfile	Either one or two input files can be specified. The first (perhaps only) contains the **hardcopy** header. If there is only one input file, the plot data must immediately follow the four line header. Libraries cannot be specified. If two input files are specified, the first contains the **hardcopy** header and any library specifications, and the second contains the plot data.

Hardcopy Header File Format

The **hardcopy** command can plot ASCII vectorized format files, ASCII component (body) files, or GED binary format files. The file that **hardcopy** reads has a special header at the top containing information about the type of plotter, line width, scale, and the format of the graphical information. In addition, if the file is binary, the header contains a list of rooted paths of SCALD directories to search to find bodies.

All the fields of the header format file are followed by a newline character.

The format of the **hardcopy** input file follows.

Type of plotter

Refer to your system's *Guide to Operations* manual for a listing of supported plotters.

Line weight

Line weight can be:

1	NORMAL_WEIGHT lines
2	HEAVY_WEIGHT lines

The default is 2.

Font type

The **font type** parameter appears on the same line as the **line weight** parameter. **Font type** is optional. The available **font types** are:

- **vector_font** *(Default)*
- **valid_font**
- **milspec_font**
- **gothic_font**
- **cursive_font**
- **greek_font**
- **symbol_font**
- **native_font** *(For built-in plotter fonts)*

Scale

Scale can be:

- An ASCII positive real number string
- A capital letter drawing page size (A, B, C, D, or E)

Illegal scales default to 1.0.

Coordinates-per-inch

Coordinates-per-inch is an optional parameter which appears on the same line as the **scale** parameter. **Coordinates-per-inch** is an ASCII integer. If the coordinates-per-inch parameter is not included, any real number scale is assumed to be pre-compensated for the target plotter's bits per inch coordinate system. (**hardcopy** multiplies the incoming coordinates by the scale to get correct plotter coordinates.)

Encoding type

Encoding type can be:

V	Vectorized
B	Binary
C	Component or body

An illegal type causes the **hpf** program to abort.

Directory paths

A list of **rooted** SCALD directory paths follows the encoding type *only* if the encoding type is binary (**B**).

Encoded drawing

This is usually put in the second input file, but can be put here, directly following the four-line header. On VMS, if there are libraries specified, you *must* use the **-2** option.

Several different header types are shown in Figure 5-1.

Header	Comments
hp7580 1 milspec_font D V	*HP 7580 pen plotter* *NORMAL lines, milspec font* *scaled to D size* *Vectorized format file*
vers11 2 gothic_font B B /usr/valid/pdh/pdh.wrk /usr/valid/lib/standard/standard.lib /usr/valid/lib/lsttl/lsttl.libl	Sun workstation *11-inch Versatec* *HEAVY lines, Gothic font* *scaled to B size* *binary format file* *directories to seach for bodies*
vers11 1 A B /u0/tlh/tlh.wrk#scald /u0/lib/standard/standard.lib /u0/lib/lsttl/lsttl.libl	SCALDsystem *11-inch Versatec* *NORMAL lines, default font* *scaled to A size* *binary format file* *directories to search for bodies*
calcomp1043 1 E B $disk2:[jao]jao.wrk scald$root:[libraries.standard]standard.lib scald$root:[libraries.lsttl]lsttl.lib	VAX workstation *CalComp 1043 pen plotter* *NORMAL lines, default font* *scaled to E size* *binary format file* *directories to search for bodies*

Figure 5-1. Examples of Hardcopy Headers

Note: Comments *cannot* appear in a **hardcopy** header. They are included in this table for your information only.

Standalone Hardcopy Procedures

Note: There must not *be any extra lines after the last SCALD directory path or* **hardcopy** *will not work properly.*

The information in **hardcopy** input file format allows you to create hardcopies without entering GED. Follow this procedure to use the *hpf* program:

1 Create a header file with all the information on plotter, font, scale, file format, etc. Figure 5-2 shows an example header file called **bsizehp7580.header**.

VAX Workstation	hp7580 1 B B $disk2:[jao]jao.wrk scald$root:[libraries.standard]standard.lib scald$root:[libraries.lsttl]lsttl.lib
Sun Workstation	hp7580 1 B B /usr/valid/jao/jao.wrk /usr/valid/lib/standard/standard.lib /usr/valid/lib/lsttl/lsttl.lib
SCALDsystem	hp7580 1 B B /u0/jao/jao.wrk /u0/lib/standard/standard.lib /u0/lib/lsttl/lsttl.lib

Figure 5-2. Example Header File

2 Enter the name of the binary drawing file that you want to plot. In most cases, it is easy to determine the name of the directory that contains your drawing; for example, the drawing MY LOGIC would be in the directory *mylogic*. If you have many similarly named drawings, look in your SCALD directory (*user*.wrk file) for the mapping between GED drawing names and the corresponding system directories.

Each drawing directory contains a file which is the binary format GED file. The file is called:

UNIX: **logic_bn.***version.page*

VMS: **logic_bn$***version*$*page*.**dat**

In this example, the binary file is:

UNIX: **/usr/valid/jao/mylogic/logic_bn.1.1**

VMS: **$DISK2:[JAO.MYLOGIC]LOGIC_BN$1$1.DAT**

3 Use one of the following procedures to create and plot the plot file.

VMS

Create the plotter file with the following command:

```
$ HPF -F PLOTTER.OUT -2 BSIZEHP7580.HEADER -
  $DISK2:[JAO.MYLOGICDRAWING]LOGIC_BN$1$1.DAT
```

If $PLOTTER and SYS$PLOT are set up as described in "Local Plotting" in the *Guide to Operations*, you can plot this file with the command:

```
$ PRINT/DEVICE=SYS$PLOT PLOTTER.OUT
```

UNIX

Use one of the following commands:

```
% cat bsizehp7580.header mylogicdrawing/logic_bn.1.1 | hpr
```

or

```
% cat bsizehp7580.header mylogicdrawing/logic_bn.1.1 > hpr.infile
% hpr hpr.infile
```

6 Adding Physical Information

This section explains:

- Back annotation
- Manual physical part assignments
- The *chips_ prt* file

After completing the logical design, you can add information about the physical part assignments to the drawing. You can include part reference numbers (U–numbers) and pin numbers on the drawing.

To do this, use the **backannotate** command to automatically add information generated by the Packager and the physical design system. You can also manually add physical information with GED commands.

Back Annotating the Design

Back annotation brings physical design information from the Packager and adds it to the logic 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.

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

1 **GED**	Schematic capture: Create the logical design.
2 **PACKAGE**	Analyze the design further and prepare for use by a physical design system.
3 **BACKANNOTATE**	Add physical design information generated by the Packager to the design.
4 *physical interface program*	Format the design for the physical design system.
5 *physical design system*	Produce the circuit from the design and prepare for manufacturing.
6 **PACKAGE**	Reassign physical parts based on feedback files from the physical design system.
7 **BACKANNOTATE**	Update the GED drawing 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 comments that are not output to the connectivity file. That is, back annotated information does not appear in the SCALD system database; 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 typically adds $LOCATION and $PN properties to the drawing. The next time you back annotate 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 back-annotation file, named *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.

Follow this procedure to use the **backannotate** command. Before you begin, if you wish to save the original *pstback.dat* file, copy it to a file called *backann.cmd.*

1 Enter GED and edit the required drawing.

NOTE: If you copied the pstback.dat *file to* backann.cmd, *you do not have to specify the name of the file; GED automatically reads* backann.cmd *by default.*

2 Type the command, followed by the name of the file containing the physical part assignments. For example:

```
BACKANNOTATE pstback.dat [Return]
```

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

Displaying Pin Numbers

The **set** command provides several options for controlling the display of pin numbers on the drawing:

near_pn **far_pn**	Controls the placement of pin numbers on the drawing. **far_pn** (the default) places the pin number to the upper–right of the connection if the connection is to the right of the origin of the body, and places the pin number to the upper–left if the connection is on the left side of the body. **near_pn** attempts to place pin numbers as close to the body as possible.
rotate	Determines whether pin numbers annotated to vertical pins are rotated. The default is **rotate on.**
pin_size	Changes the size of added pin numbers. The default is 0.80 times the default text size (0.082 inches). The scale can be set to any real number; for example, 0.75.

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 back annotate 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 should only be attached 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.

The Packager directive HARD_LOC_SEC OFF allows the Packager to override manual assignments. See the ValidPackager Reference Manual *for more information. Otherwise, to change a manual assignment, you must delete the information or manually assign new information.*

When you assign physical part information to the logical parts of a GED drawing, you 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 usually override the assignments.

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 after packaging a design, but then 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** or **zoom** command to enlarge the appropriate part of the drawing so that the part is clearly visible.

2. Type:

   ```
   section Return
   ```

3. Point to the origin of the body you want to assign a section and press the yellow 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, type the pin number, then point to a pin. If you know exactly which section is required, this can save the time of cycling through the various sections.

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.

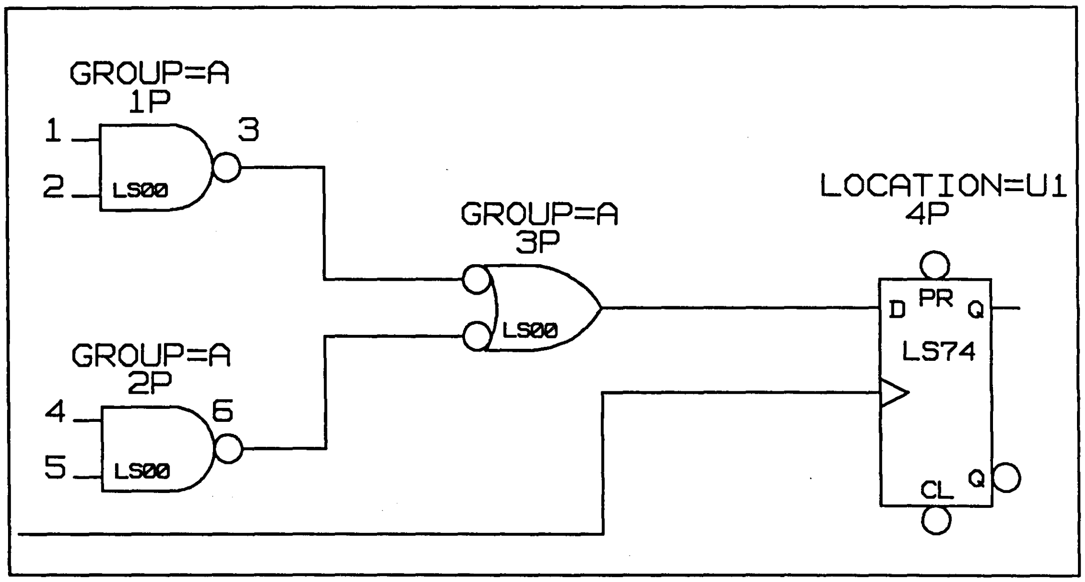


Figure 6-1. Adding Physical Information

In Figure 6-1, the properties that have been added have the following effects.

LOCATION

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

GROUP

The LS00 parts 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 ValidPackager Reference Manual *for more information on grouping parts.*

SEC

The **section** command was used on two of the LS00 parts 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 affected by the **section** and **pinswap** commands. Only pins having the pin_group property in a *chips_ prt* files can be affected by the **pinswap** command. 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. In UNIX, the chips file is called *chips_ prt*. In VMS, this file is called *chips_ prt.dat.*

SCALDsystem or PC AT

On a SCALDsystem or PC AT, use the following command to divide a library chips file:

/usr/bin/makechipsfiles *library_chips_file SCALD_directory_file*

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 100171 device.

Sun Workstation

On a Sun workstation, use the following command to divide a library chips file:

/usr/valid/tools/bin/makechipsfiles *library_chips_file SCALD_directory_file*

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

```
% cd /usr/valid/tools/libraries/100k
% /usr/valid/tools/bin/makechipsfiles 100k.prt 100k.lib
```

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

```
/usr/valid/tools/libraries/100k/100171/chips_prt
```

is the individual chips file for the 100171 device.

VAX Workstation

On a VAX workstation, use the following command to divide a library chips file:

$MAKECHIPSFILES *library_chips_file SCALD_directory_file*

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

```
$ SET DEFAULT SCALD$ROOT:[LIBRARIES.100K]
$ MAKECHIPSFILES 100k.prt 100k.lib
```

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

SCALD$ROOT:[LIBRARIES.100k.100171]chips_prt.dat

is the individual chips file for the 100171 device.

To use the **section** and **pinswap** commands, installations must have the following program:

SCALDsystem or PC AT:	**/u0/scald/section/section**
Sun workstation:	**/usr/valid/tools/section/section**
VAX workstation:	**SCALD$ROOT:[SECTION]SECTION.EXE**

This is the program that computes the section and pin assignments for the various parts. The files are included with GED.

Installations must also include the following files:

SCALDsystem or PC AT:	**/u0/scald/section/secassign.sh** **/u0/scald/libtools/makechipsfiles**
Sun workstation:	**/usr/valid/tools/section/secassign.sh** **/usr/valid/tools/libtools/makechipsfiles**
VAX workstation:	**SCALD$ROOT:[LIBTOOLS]MAKECHIPSFILES.COM**

7 Mixing Text and Graphics

This section explains:

- Adding drawings to existing text files
- Creating documents interactively
- Changing an existing document
- Printing a document with GED
- Using font styles

You can create mixed text and graphics documents interactively using the GED set of graphics tools. You can also add graphics to existing text. The need to physically "paste-up" drawings into text is eliminated.

Adding Drawings to Existing Text Files

The GED **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 called DOC drawings in GED. 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 Compiler.

Refer to the appropriate reference documentation for more information on the text editors.

The text file should be created with a text editor such as *vi* or EDT using a text formatter such as nroff or Digital Standard Runoff. However, the default page length is too long for the font created by GED. To create pages with 59 lines, use a formatting command to set the page length to 59 at the beginning of the file.

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

Remember to use the complete drawing name syntax: name.type.version.page.

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

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

3 On the next line type the formatting command to insert the number of blank lines you specified. This allows GED to properly format each page.

EXAMPLE

```
& AN EXAMPLE.LOGIC.1.1
12
.sp 12
```

4 Save your text file and then issue the **format** command:

`format` *filename* Return

drawing_name Return

GED accesses the drawings specified in the 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 to the **format** command.

The arguments to the **format** command are:

- The name of the text file
- A Return
- The drawing name

format then creates a document called *drawing_name*.DOC. The text file has been pre-processed by a text formatting program (for example, nroff or Runoff). Each page of the text file is turned into a page in a SCALD drawing. A page ends after 60 lines or after 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 the GED 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 plotted output to the correct size.

Creating Documents Interactively

Creating a document with GED requires being able to add both text and drawings. To add text lines, use 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 the sample document shown in Figure 7–1:

1. Use a text editor to create a file called *mux.txt*.
2. Enter the following 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:

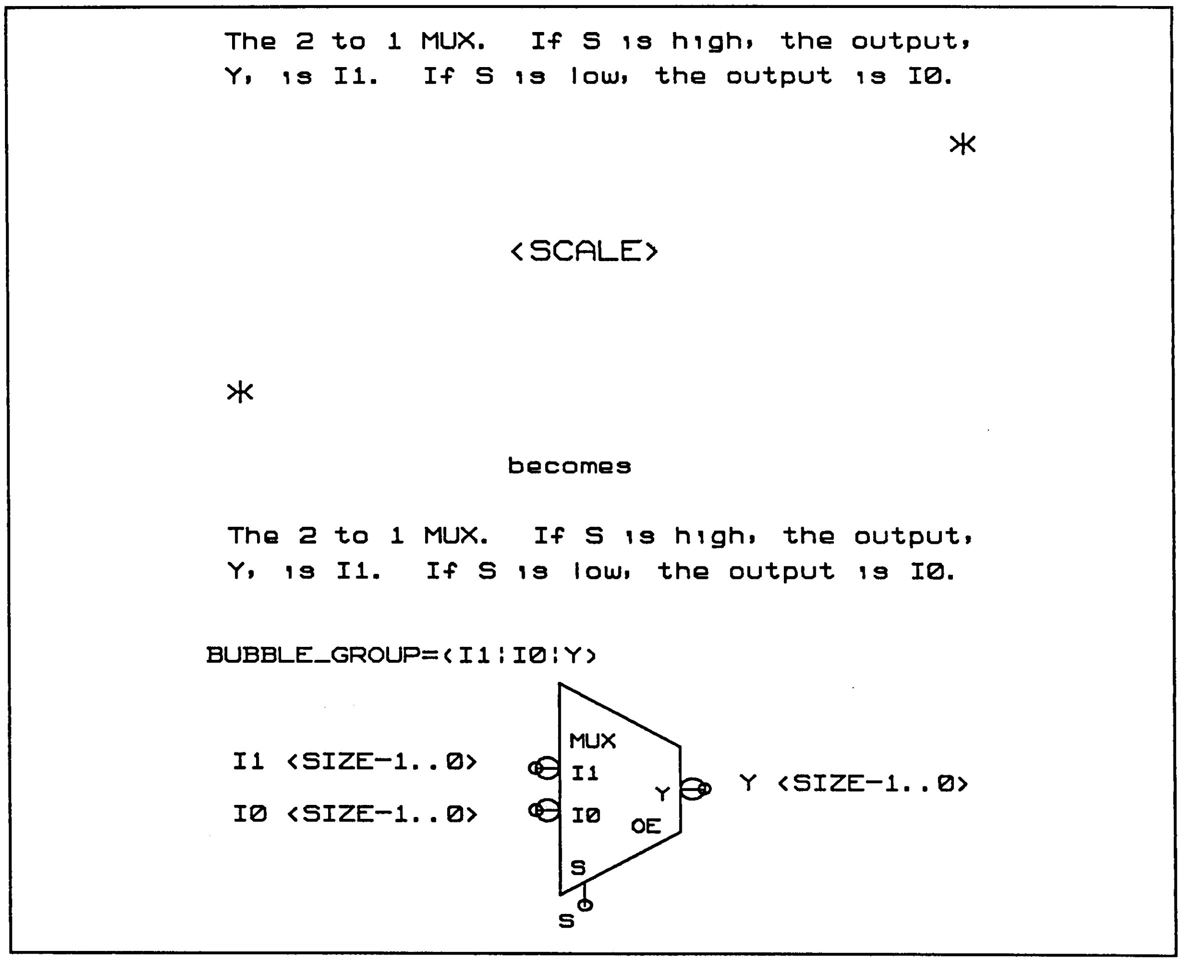
   ```
   filenote mux.txt
   ```

   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.

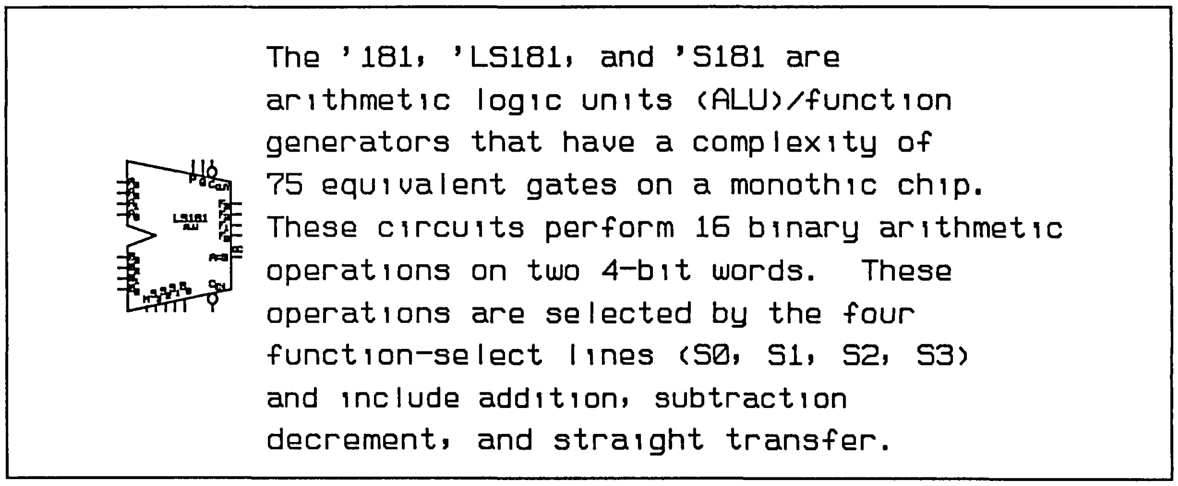
The 2 to 1 MUX. If S is high, the output, Y, is I1. If S is low, the output is I0.

Figure 7-1. The Example DOC Drawing

Changing an Existing Document

Once a document is created, either using the **format** command or interactively using the **filenote** and **scale** commands, it can be edited using GED. You may want, for instance, to rescale figures or make simple changes to lines of text. Modifications can be made using regular GED commands such as **move**, **copy**, **change**, **wire**, and **group**.

For example, Figure 7-2 is an existing document that needs several changes made.



The '181, 'LS181, and 'S181 are
arithmetic logic units (ALU)/function
generators that have a complexity of
75 equivalent gates on a monothic chip.
These circuits perform 16 binary arithmetic
operations on two 4-bit words. These
operations are selected by the four
function-select lines (S0, S1, S2, S3)
and include addition, subtraction
decrement, and straight transfer.

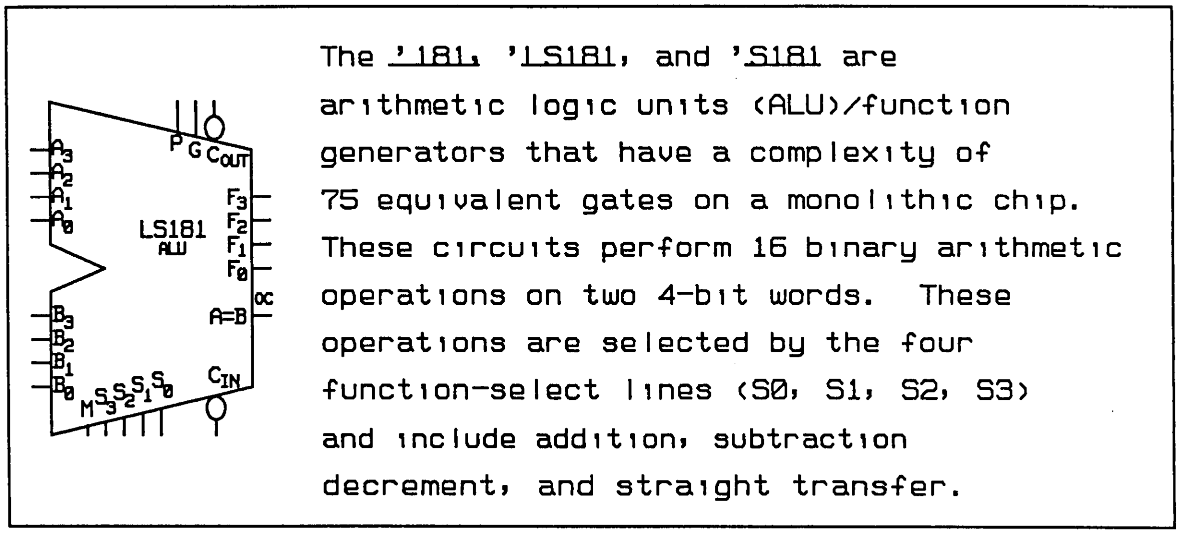
Figure 7-2. An Existing Document to Edit

1 The word "monolithic" is misspelled in the third line. This can be corrected with the **change** command.

2 Use the **wire** command to underline the chip names.

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 changed drawing is shown in Figure 7–3.


P
G
C_{OUT}
A_3
A_2
A_1
A_0
LS181
ALU
F_3
F_2
F_1
F_0
OC
A=B
B_3
B_2
B_1
B_0
M
S_3
S_2
S_1
S_0
C_{IN}

The '181, 'LS181, and 'S181 are arithmetic logic units (ALU)/function generators that have a complexity of 75 equivalent gates on a monolithic chip. These circuits perform 16 binary arithmetic operations on two 4-bit words. These operations are selected by the four function-select lines (S0, S1, S2, S3) and include addition, subtraction decrement, and straight transfer.

Figure 7–3. The Existing Document Changed

Printing a Document With GED

When a document is created using the **format** command, tick marks are placed at the corners of the page. These tick marks serve two purposes. First, they allow you to hardcopy at the default scale (**hardcopy 1**) and get an 8 1/2 by 11 inch page. Second, the tick 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 tick marks on the sides of the document or place text outside the tick marks.

If the marks are deleted or the page is created manually, type the following to create the tick 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, and the page will always be plotted correctly.

Using Font Styles

GED supports a variety of printing font styles that 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. All text strings in the design are plotted in the same font.

8 Drawing Maintenance

This section explains:

- Crash recovery procedures
- Updating drawings

Temporary Files and Crash Recovery

When you edit a second drawing without writing out the first one, GED 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 workstation and window. The ? is a letter designating the number of the temporary file stored in your operating system directory. Temporary files are not written into your SCALD directory.

These temporary files are deleted from the operating system if GED terminates normally. However, when GED exits abnormally, all drawings except the last one can be restored to the versions saved in the temporary files.

If GED or the system crashes, it is possible to recover the drawings that were being edited while GED was running. Every time GED is called to the screen after a system crash, a query appears as the first message from the editor asking if you want to recover files. In the event of a crash, you should recover files by answering "yes" to the question about recovering 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, and so on. If a *restore.wrk* SCALD directory already exists, it is overwritten. A warning message is printed regarding 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 reverse order. If there are drawings RESTORED1, RESTORED2, RESTORED3, RESTORED4, and RESTORED5, start editing RESTORED5 and work back to RESTORED1.

3 **write** the edited drawings with their correct names to the appropriate SCALD directory.

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 GED 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 the operating system 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 a part has been added or deleted, that property is added or deleted on the drawing. In addition, if you have modified a part property value, that value overrides any default value.

Dependency Files

The dependency file lists the bodies used by a drawing and the operating system directory from which the parts came. GED writes a dependency file in addition to the ASCII, binary, and connectivity files for each drawing. When you run the update facility, the date of 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

You enter the *gedupdate* command at the system prompt to update the drawings in your directory.

SYNTAX

gedupdate *argument*

The command arguments are:

Return (*no arguments*)

Find all drawings in the current directory that need updating and remake them. This option deletes the binary files and then reads in the ASCII versions of the drawings so that changes to properties are handled properly.

-n

Find all 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. This option 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 option (no argument), 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 enclosed in quotation marks and fully specified; wildcards are not allowed. For example:

```
gedupdate -a "SIZE SHIFTER.LOGIC.1.1"
```

-f "*part name*"

This option finds and lists all the drawings that use the named part. Quotation marks enclose 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, enter:

```
gedupdate -f "3 MERGE"
```

If you type in an unknown argument, a help message lists the arguments for **gedupdate** and the meaning of each.

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

A GED Files

This section contains information about the files and file structures used by GED, including:

- System initialization files
- ASCII files
- Body files
- Connectivity files
- Dependency files
- Back annotation file
- Vector plot format

System Initialization Files

Several system files are used to initialize GED when you enter the system.

The Startup.valid File

The system-wide initialization file, *startup.valid*, defines the location of the master library file and some of the directories referenced by GED.

This file also defines the default function key assignments for the workstation keyboard. On Valid SCALDsystems, *startup.valid* establishes the names of connections that can be used by the **connect** command. The connections specified are LED, CONCORDE, SIMULATOR, and COMPARE.

The system location of this file is:

SCALDsystem and PC AT:	**/u0/editor/startup.valid**
Sun Workstation:	**/usr/valid/tools/editor/startup.valid**
VAX Workstation:	**SCALD$ROOT:[EDITOR]STARTUP.VALID**

The System Startup.ged File

The following script is run by the system-wide initialization file. Valid does not install or modify this file. The system administrator at individual sites uses this file to provide system-wide commands without changing the Valid-supplied file, *startup.valid*.

The system location of this file is:

SCALDsystem and PC AT:	**/u0/editor/startup.ged**
Sun Workstation:	**/usr/valid/tools/editor/startup.ged**
VAX Workstation:	**SCALD$ROOT:[EDITOR]STARTUP.GED**

The Softkeyassign File

The *softkeyassign* file contains the default values of the function keys. Section 1 contains a listing of the default function keys. Refer to the **assign** command in the *ValidGED Command Reference Manual* for more information about programming the function keys.

The system location of this file is:

SCALDsystem and PC AT:	**/u0/editor/softkeyassign**
Sun Workstation:	**/usr/valid/tools/editor/softkeyassign**
VAX Workstation:	**SCALD$ROOT:[EDITOR]SFTKEYASSIGN.DAT**

The Config.dat File

The *config.dat* file is used by GED to define the library format for signal name processing. This file defines the characters that are used for signal name syntax (such as the asterisk to indicate a low asserted signal). See the *SCALD Language Reference Manual* for more information on the *config.dat* file.

The system location of this file is:

SCALDsystem and PC AT:	**/u0/editor/config.dat**
Sun Workstation:	**/usr/valid/tools/editor/config.dat**
VAX Workstation:	**SCALD$ROOT:[EDITOR]CONFIG.DAT**

The Master.lib File

The *master.lib* file contains the name translations for the Valid part libraries. The file entries contain the abbreviated names for the GED **library** command and the file system location of the libraries.

For example, an entry in master.lib on the SCALDsystem appears as:

```
"sttl"  '/u0/lib/sttl/sttl.lib';
```

In VMS, the entry is:

```
"sttl"  'SCALD$ROOT:[LIBRARIES.STTL]STTL.LIB'
```

This entry makes a permanent "alias" for the STTL library so that it always refers to the pathname or file specification. Consequently, you can type:

```
library sttl
```

instead of typing the **use** command with the entire pathname or file specification.

The system location of this file is:

SCALDsystem and PC AT:	**/u0/lib/master.lib**
Sun Workstation:	**/usr/valid/lib/master.lib**
VAX Workstation:	**SCALD$ROOT:[LIBRARIES]MASTER.LIB**

ASCII Files

One of the design database files that GED creates when a drawing is written is an ASCII file. An ASCII file is a script file that is used to represent all drawings except for BODY drawings. It is a specific type of text file that consists of commands to add each object in a drawing. GED "recreates" a drawing by reading the commands in the ASCII file; you can edit the ASCII file to make changes in your drawing.

GED internal coordinates are 0.002 inches per unit by default. Points are represented in ASCII files by their coordinates, enclosed in parentheses and separated by a space. Thus, the point x=100, y=200 becomes (100 200).

Angles are represented by a number from zero through seven:

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 consist of an identification line, commands to represent the type and location of each object in the drawing, and a QUIT statement. The file components are described in the following sections.

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 the line:

```
QUIT
```

Object Definitions

Each type of object in a GED drawing has a specific definition format.

Bodies

A body definition uses up to four lines in the ASCII file, in the following format:

forceadd *name*
[**R** *angle*]
pt ;
[**paint** *color pt*]

name	The body name includes the version number.
R *angle*	The rotation of the added body. The *angle* definition is optional.
pt ;	The placement point of the body.
paint *color pt*	The optional **paint** command is included if the body is other than the default color.

forceadd is used so that a place holder is created if the body is not found.

Wires

A description of a wire in the ASCII file consists of a single line in the format:

wire *linetype pattern pt1 pt2* ;

linetype — This numeric argument includes the line color and thickness definition. 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.

pattern — The fill pattern of the line. If *pattern* = -1, the line is filled. There are six defined wire patterns in GED:

1	-1	**4**	2175
2	273	**5**	3135
3	682	**6**	4383

pt1 pt2 — The begin and end points of the wire.

Dots

Dots are described in the following format:

dot *type pt* ;
[**paint** *color pt*]

type — If *type* = 0, the dot is open; if *type* = *1*, the dot is filled. If the type is not 0 or 1, the dot is assumed to be open.

pt — The location of the dot on the drawing.

paint *color pt* — The optional **paint** command is included if the dot is other than the default color.

Circles and Arcs

Circles and arcs are described in the following format:

> **circle** *pt1 pt2* ;
> [**paint** *color pt*]
>
> *or*
>
> **circle** *pt1 pt2 pt3* ; (for arcs)
> [**paint** *color pt*]

pt	Points are represented by the X–Y coordinates that describe a location on the drawing.
paint *color pt*	The optional **paint** command is included if the circle or arc is other than the default color.

Notes

The **forcenote** command is similar to the **note** command in the editor except that the **forcenote** command terminates after reading one note. Notes are described in the following format:

> **forcenote** *contents pt angle*;
> [**display** *size pt* ;]
> [**paint** *color pt*]

contents	The text of the note.
pt	The location of the note on the drawing.
angle	The rotation of the added note.
display *size pt* ;	This line is included if the note is not the default size. This command makes the text the correct size.
paint *color pt*	The optional **paint** command is included if the note is other than the default color.

Properties

The **forceprop** command is similar to the **property** command except that it takes a *default_status* argument. Property definitions occur directly after the object to which they are attached. The format is:

forceprop *default_status* **last** *name value*
[R *angle*]
[J *justification_type*]
pt ;
[**display** *size pt* ;]
[**paint** *color pt*]

default_status

This argument is necessary for the correct handling of changes to properties on library bodies. The *default_status* flag can have three values:

0	The status of the property is unknown (an undefined variable whose status is determined when the body definition is searched).
1	The property is known to be default (one that comes from the body definition).
2	The property is known to be non-default (one that a user added to the ASCII logic drawing).

last

This argument indicates that the property is to be attached to the last object or wire entered.

If the property is a PIN property, the **last** argument is replaced by the **lastpin** argument followed by a *pt* that describes the location of the pin in absolute coordinates.

R *angle*	The rotation of the added property. The *angle* definition is optional.
J *justification_type*	The justification of the added text. There are three possible values: **0** The text is left justified. **1** The text is center justified. **2** The text is right justified. If no justification is included, the property is created with the current default justification. If an illegal justification is given, the system uses left justification as the default.
pt	The location of the property on the drawing.
display *size pt* ;	If the property does not have the standard visibility, the **display** command sets the visibility of the name and value.
paint *color pt*	The optional **paint** command is included if the note is other than the default color.

Bubbled Pins

Bubbled pins for an object are described in the format:

forcebubble *pt* ...

All pins that are not in their default bubbled state are listed.

Binary Files

Binary files contain the same information as the ASCII file described, but in a binary format that is quicker to read and write. This format is proprietary and is not described in this document.

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
- Body properties
- Pin properties
- Bubble groups

See the **set** *command in the* ValidGED Command Reference Manual *for more information on default spacing.*

As in the ASCII files, GED internal coordinates are 0.002 inches per unit by default.

Line Definitions

Lines require one 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*

x1 y1 x2 y2	The line's endpoint coordinates; the line runs from (x1 y1) to (x2 y2). The coordinates are separated by spaces.
pattern	This optional argument identifies the line style (solid, broken, etc.) as a bit pattern. For example, if *pattern* is -1, the line is solid and if *pattern* is 682, the line is dotted. See the pattern values listed with the description of wires in the ASCII file description.
color	The internal GED color number. 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.

Arc Definitions

Arcs require one line each in the body file. The line has the format:

A *x y radius start_angle stop_angle color*

x y radius	The center and radius points of the arc.
start_angle stop_angle	Floating point numbers that measure the angles, in degrees, counterclockwise from the X axis.
color	The internal GED color number.

Text String Definitions

Text strings require two lines each in the body file. The first line gives the specification of the text; the second gives the text string. The format is:

T *x y angle slant size over inv just font Nch color string*

x y	The origin point for the text string.
angle	The angle of the text on the drawing. The allowed angles are:

- 0.00
- 90.00
- 180.00
- 270.00

size	The height of the characters.
just	The justification of the added text. There are three possible values:

0	The text is left justified.
1	The text is center justified.
2	The text is right justified.

Nch	The number of characters and spaces in the string.
color	The internal GED color number.

The text definition arguments *slant, over, inv,* and *font* are not currently implemented.

Connection Definitions

Connections require one line each in the body file. The contents of the line depend on whether a pin can be bubbled or not. The format is:

```
C x y "name" dispx dispy bubbleable [default_state x2 y2 x3 y3] f size angle just
```

x y	The location of the connection.
"name"	The name of the connection. The name must be enclosed in quotation marks.
dispx dispy	The location of the name.
bubbleable	Whether or not the pin is bubbleable. There are two possible values: **0** False **1** True
default_state	Whether or not the pin is bubbled. There are two possible values: **0** False **1** True If the *default_state* is 1 when a body is initially added, the pin is bubbled.
x2 y2 x3 y3	The endpoints of the bubbleable pin. These arguments are present only if the pin is bubbleable.

f	Whether the connection is a filled or open dot. **0** Open **1** Filled
size	The size of the name string. The default is 41.
angle	The angle of the pin name string attached to the connection: **0** 0 degrees **1** 90 degrees **2** 180 degrees **3** 270 degrees
just	The justification of the string. There are three possible values: **L** The text is left–justified. **C** The text is center–justified. **R** The text is right–justified.

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"	The name of the property. The name must be enclosed in quotation marks.
"value"	The default value of the property. The value must be enclosed in quotation marks.
x y	The reference point for the property.
angle	The angle of the property.
size	The height of the characters.
just	The justification of the property. There are three possible values: **0** The text is left justified. **1** The text is center justified. **2** The text is right justified.
NV	The visibility of the property name. There are two possible values: **0** The name is invisible. **1** The name is visible. The name is visible by default.

VV	The visibility of the property value. There are two possible values: **0** The value is invisible. **1** The value is visible. The value is visible by default.
IP	Whether or not the property is a parameter. There are two possible values: **0** False **1** True
color	The internal GED color number.

The body property definition arguments *slant, over, inv,* and *font* are not currently implemented.

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.

Connectivity Files

Connectivity files describe all the bodies on a drawing. The information includes:

- The names of the bodies
- The names of the signals tied to their pins (with bubble state)
- 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
- Comments
- The NET section
- INVOKE commands

Each connectivity file has the form:

> **FILE_TYPE = CONNECTIVITY;**
> {**GED_Release:** *date and number*}
> [*expression property*]
> [*nets*]
> [*invokes*]
> **END.**

The first line is the header, the second line is a comment, the third is the expression property from the drawing body, the fourth line begins the net section, the fifth line begins the invoke command section, and the sixth is the END statement. The expression property, 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>.

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*;

EXAMPLE

```
EXPR="SIZE=10";
```

Nets

Each time GED writes a connectivity file, it numbers all the nets. The NC net is always net 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*];

constant	The net number.
"*net_name_string*"	Either the signal name for the net or the unnamed signal string created by GED. The *net_name_string* must be enclosed in quotation marks.
property_list	An optional argument, the *property_list* has the format: *property_list* ::= {*identifier string*}
identifier	The property name. The name must begin with a letter and can contain only: • Letters • Digits • Underscore (_) There are two reserved identifiers: FILE_TYPE and END.
"*string*"	The quoted string.

Two sample net entries follow.

EXAMPLE

```
2"UN$1$2P$A";
3"ANWC"LOAD"37"CONNECTED_TO"PAGE 4";
```

Invoking 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; }
```

"invoke_name_string"

The name of the component. The string must be enclosed in quotation marks.

"version_str"

The body version number; the string must be enclosed in quotation marks. This property is always output. If this property doesn't exist, the null string (" ") is used.

xy_str

The coordinates of the body on the page. This property is always output. If this property doesn't exist, the null string (" ") is used.

"rotation"

The rotation of the body; the string must be enclosed in quotation marks. This property is always output; if it doesn't exist, the null string (" ") is used. *rotation* can be one of the following values:

0	0 degrees rotation
1	90 degrees 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

directory_str	The name of the directory or library where the body originated, for example, lsttl.lib. The full pathname or file specification is not required. If this property doesn't exist, the null string (" ") is used.
path_str	The path property. If this property doesn't exist, the null string (" ") is used.
parameter_property_list	An optional list with the same format as a property list.
property_list ;	The *property_list* is optional; the semicolon is required.
"*pin_name_string*"	The pin name string must be enclosed in quotation marks.
constant	The number of the net that is attached to the *pin_name_string*. The net numbers are assigned in the net section.

EXAMPLES

```
% "LS00"                                   {body name}
"1","(100,345)","0","lsttl.lib","2P"; {body info}
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;
```

```
% "LS02"                                   {body name}
"2","(500,1234)","3","lsttl.lib","";{body info}
;                             {parameter property list}
COLOR"RED";                        {body property list}
"A"23;                                     {pin names}
"B"5;
"Y"OUTPUT_LOAD"(50.0,-50.0)"3;
```

Dependency Files

Dependency files list the operating system filenames for each body added to a drawing. These files are used by the GED update procedure, which allows drawings to be updated if any bodies are out of date. There are dependency files for all drawings except BODY and DOC drawings.

The first line of a dependency file is the drawing's logic file name, a colon (:), and a list of body file names separated by blanks. The names are all complete UNIX pathnames or full VMS file specifications.

For example, a sample dependency file for the logic drawing MY EXAMPLE.LOGIC.1.1 is:

SCALDsystem and PC AT:

```
myexample/logic.1.1: \
    /u0/lib/lsttl/ls00/body.1.1: \
    /u0/lib/lsttl/ls03/body.1.1 \
    adder/body.1.1 \
    shifter/body.1.1
```

Sun Workstation:

```
myexample/logic.1.1: \
    /usr/valid/tools/lib/lsttl/ls00/body.1.1: \
    /usr/valid/tools/lib/lsttl/ls03/body.1.1 \
    adder/body.1.1 \
    shifter/body.1.1
```

The \ (followed by a Return) is used to continue across lines. Subsequent lines begin with a Tab. 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 to enter only shifter/logic.1.1.

However, parts that are added from SCALD directories not in the current UNIX directory are given a full file specification name. The entire path name must be written out; no wildcards are allowed.

VAX Workstation:

```
[.MYEXAMPLE]LOGIC$1$1.DAT: \
  SCALD$ROOT:[LIBRARIES.LSTTL.LS00]BODY$1$1.DAT  \
  SCALD$ROOT:[LIBRARIES.LSTTL.LS03]BODY$1$1.DAT  \
  [.ADDER]BODY$1$1.DAT \
  [.SHIFTER]BODY$1$1.DAT
```

The \ (followed by a Return) is used to continue across lines. Subsequent lines begin with a Tab. Files are referenced from the VMS directory containing the SCALD directory file that holds the logic drawing. For example, from the directory SCALD$ROOT[CLASS], you need to enter only [.SHIFTER]LOGIC$1$1.DAT. However, parts that are added from SCALD directories not in the current VMS directory are given a full file specification name. The entire path name must be written out; no wildcards are allowed.

The last line in the dependency file is:

SCALDsystem and PC AT: **/u0/editor/MakeAddToList** "*drawing_name.extension.version.page*"

Sun Workstation: **/usr/valid/tools/editor/MakeAddToList** "*drawing_name.extension.version.page*"

VAX Workstation: **SCALD$ROOT:[EDITOR]MAKEADDTOLIST** "*drawing_name.extension.version.page*"

The *drawing_name* is quoted, and all parts of the name (extension, version, and page) must be specified.

Back Annotation File

This section discusses the format for the file read by the Graphics Editor's **backannotate** command and generated by the Packager. If you do not use the back annotation file generated by the Packager, there is no guarantee that the annotated information is consistent with the physical design.

The back annotation file contains physical information grouped by drawing. The back annotation file is named *pstback.dat* by the Packager.

The first line is of the file is:

FILE_TYPE = BACK_ANNOTATION;

The last line in the file is:

END.

The information in the file includes:

- Drawing name
- Pin names
- Body names
- 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.

Drawing Names

The drawing name line has the following syntax:

```
DRAWING = "SCALD dwg_name.extension.version.page";
```

The drawing name must be enclosed in quotation marks.

Body Names

Body names are 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 is:

```
BODY = "name","path_property";
```

If properties are to be attached, the statement ends with a colon and is followed by property name/value pairs, separated by commas.

```
BODY = "name","path_property": prop1 = "value1", ... propN = "valueN";
```

Property values are enclosed in quotation marks, but not property names.

There **MUST** be spaces around any equal sign (=).

Pin Names

Pin names include 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 is enclosed in quotes. For instance:

```
PIN = "pin_name": prop1 = "value1", ... propN = "valueN";
```

Property names must be 15 characters or less. Property values are enclosed in quotes, but not property names. There MUST be spaces around any equal 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. Both scalar and vectored nets can be annotated. The form is:

NET = "*net_name*": *prop1* = "*value1*", ... *propN* = "*valueN*";

Sample Back Annotation File

```
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": PNN=GLOBALXOUT
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": $XRF=4A5
END.
```

Vector Plot Format

This section describes the format of the plot file produced with GED'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. The first character of the line specifies the type of the primitive. All units are nominally 0.002 inches.

Line Primitive

Lines require one line each in the file. The format is:

L *x1 y1 x2 y2 [pattern] color*

x1 y1 x2 y2	The line's endpoint coordinates; the line runs from (x1 y1) to (x2 y2). The coordinates are separated by spaces.
pattern	This optional argument identifies the line style (solid, broken, etc.) as a bit pattern. For example, if *pattern* is –1, the line is solid and if *pattern* is 682, the line is dotted. See the pattern values listed with the description of wires in the ASCII file description.
color	The internal GED color number. The line type describes both the color and thickness of the line. When the integer is converted to a binary value, bit zero defines the thickness (0 = thin), and the seven most significant bits define the color.

Arc Primitive

The format of the arc primitive is:

A *x y radius start_angle stop_angle*

x y radius	The center and radius points of the arc.
start_angle stop_angle	Floating point numbers that measure the angles, in degrees, counterclockwise from the X axis.

Text String Primitive

Each text string primitive consists of the following four lines; each line is terminated by a line feed character.

T *x y*
angle slant size overbar inverse_video
justification font
string

x y — The origin point of the text string.

angle — The angle of the text on the drawing. The allowed angles are:

- 0.00
- 90.00
- 180.00
- 270.00

justification — The justification of the added text. There are thee possible values:

0	The text is left justified.
1	The text is center justified.
2	The text is right justified.

string — The text string. No quotation marks are required.

The text definition arguments *slant, overbar, inverse_video,* and *font* are not currently implemented.

If supported on your system, refer to the source in the following file 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.

SCALDsystem or PC AT	**/u0/editor/lib/hpfilter.c**
Sun Workstation	**/usr/valid/tools/editor/lib/hpfilter.c**

This program is run by the **hardcopy** command when you have **set mono_hpplot.**

B Hardcopy Fonts

This section identifies the fonts supported by GED:

- **vector_font**
- **valid_font**
- **milspec_font**
- **gothic_font**
- **cursive_font**
- **symbol_font**
- **greek_font**
- **native_font**

The font name in parentheses is the *font_name* argument for the **font** option of the **set** command (**set font** *font_name*).

To use these fonts on a drawing, issue the **set** command to specify the required font and then **hardcopy** your drawing. Only one font style can be active at any time, and the active font affects all drawings plotted while it is active.

Native Font uses a font that is built into the plotter where applicable.

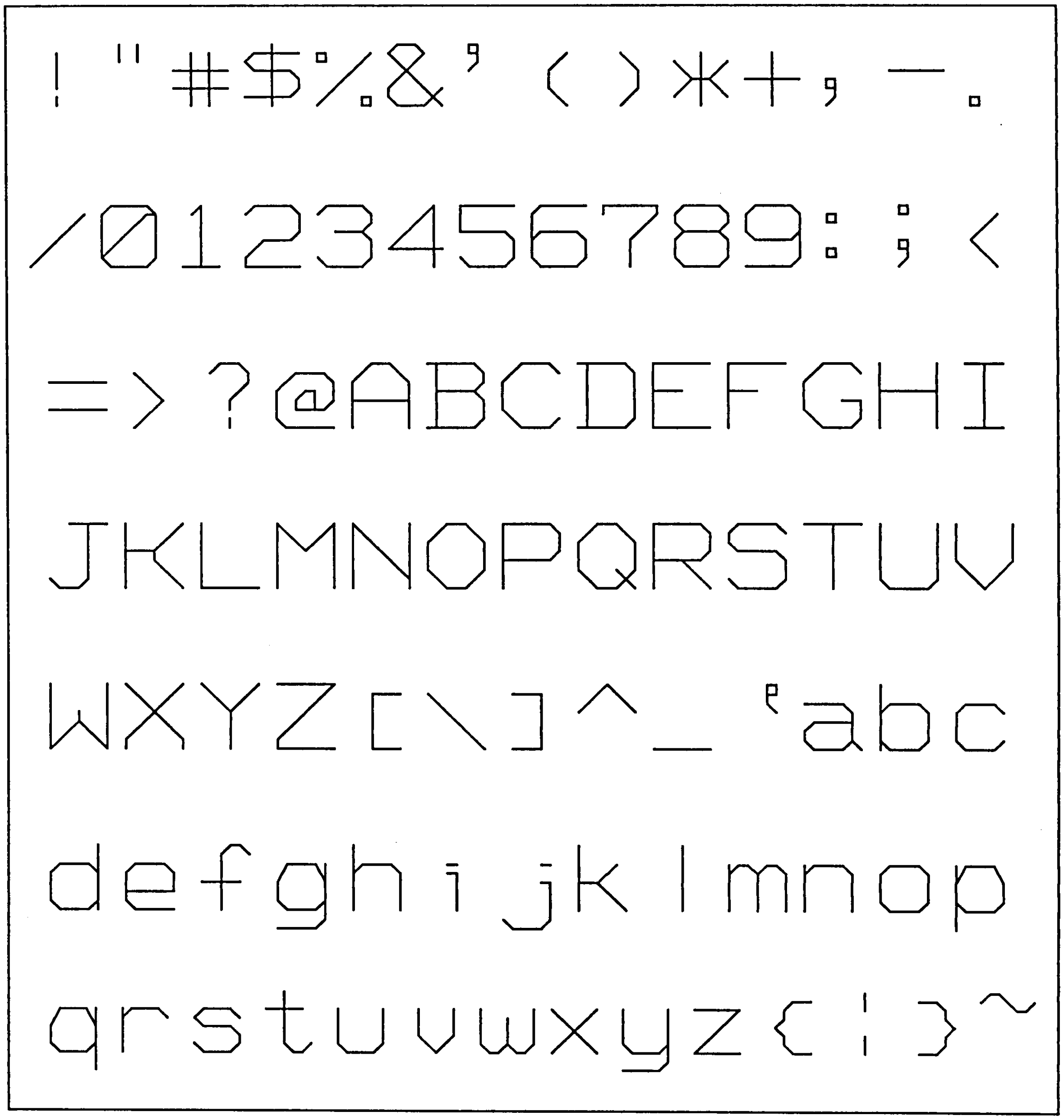

! " # $ % & ' ( ) * + , - .
/ 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I
J K L M N O P Q R S T U V
W X Y Z [\ ] ^ _ ` a b c
d e f g h i j k l m n o p
q r s t u v w x y z { | } ~

Figure B-1. Vector Font (Default)

33 !	34 "	35 #	36 $	37 %	38 &	39 '	40 (	41)	42 *	43 +
44 ,	45 -	46 .	47 /	48 0	49 1	50 2	51 3	52 4	53 5	54 6
55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?	64 @	65 A
66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L
77 M	78 N	79 O	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _	96 `	97 a	98 b
99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m
110 n	111 o	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x
121 y	122 z	123 {	124 ¦	125 }	126 ~					

Figure B–2. Vector Font ASCII Codes

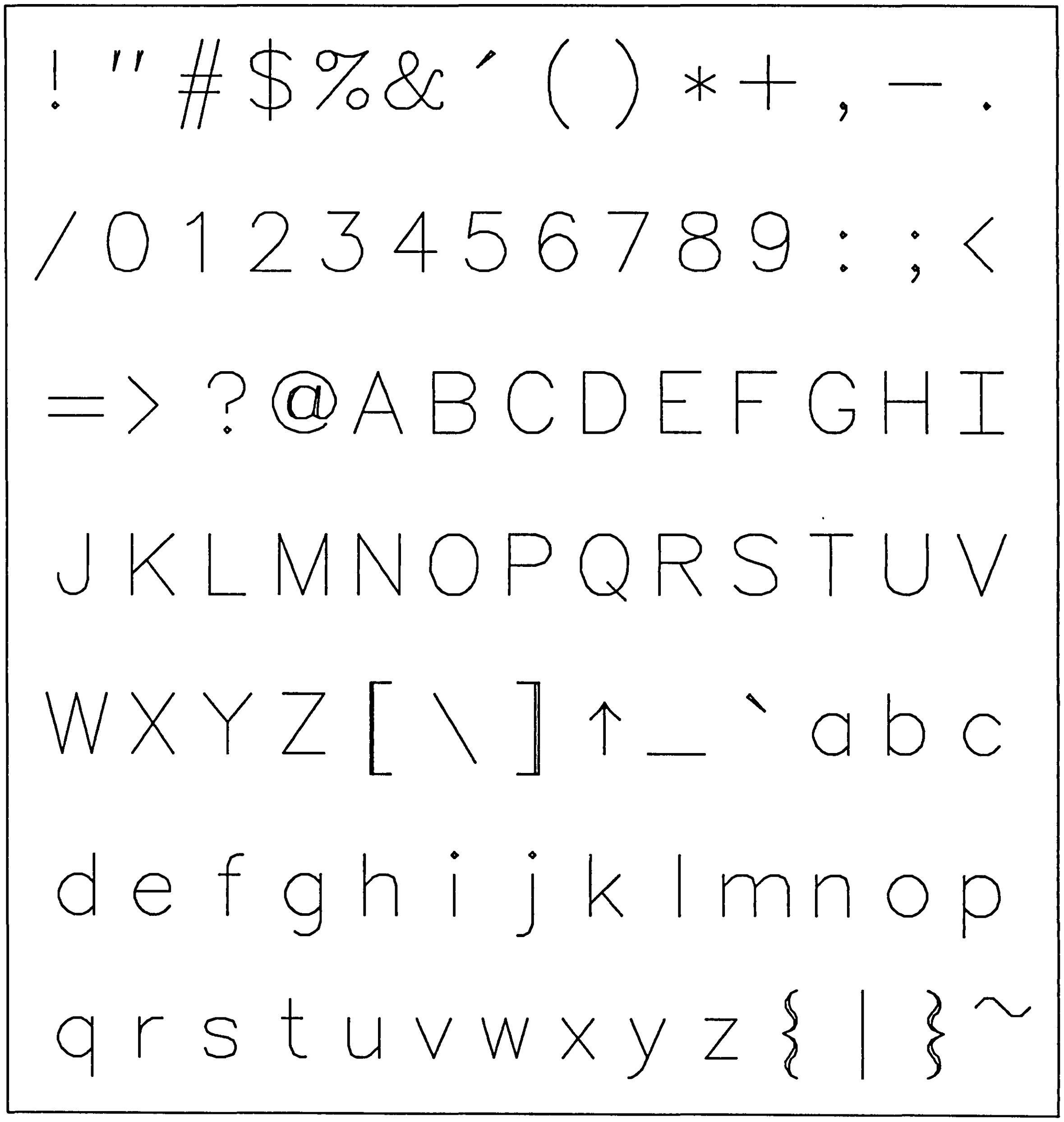

! " # $ % & ' ( ) * + , - .
/ 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I
J K L M N O P Q R S T U V
W X Y Z [\ ] ↑ _ ` a b c
d e f g h i j k l m n o p
q r s t u v w x y z { | } ~

Figure B–3. Valid Font

33 !	34 "	35 #	36 $	37 %	38 &	39 '	40 (	41)	42 *	43 +
44 ,	45 –	46 .	47 /	48 0	49 1	50 2	51 3	52 4	53 5	54 6
55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?	64 @	65 A
66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L
77 M	78 N	79 O	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ↑	95 _	96 `	97 a	98 b
99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m
110 n	111 o	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x
121 y	122 z	123 {	124 |	125 }	126 ~					

Figure B-4. Valid Font ASCII Codes

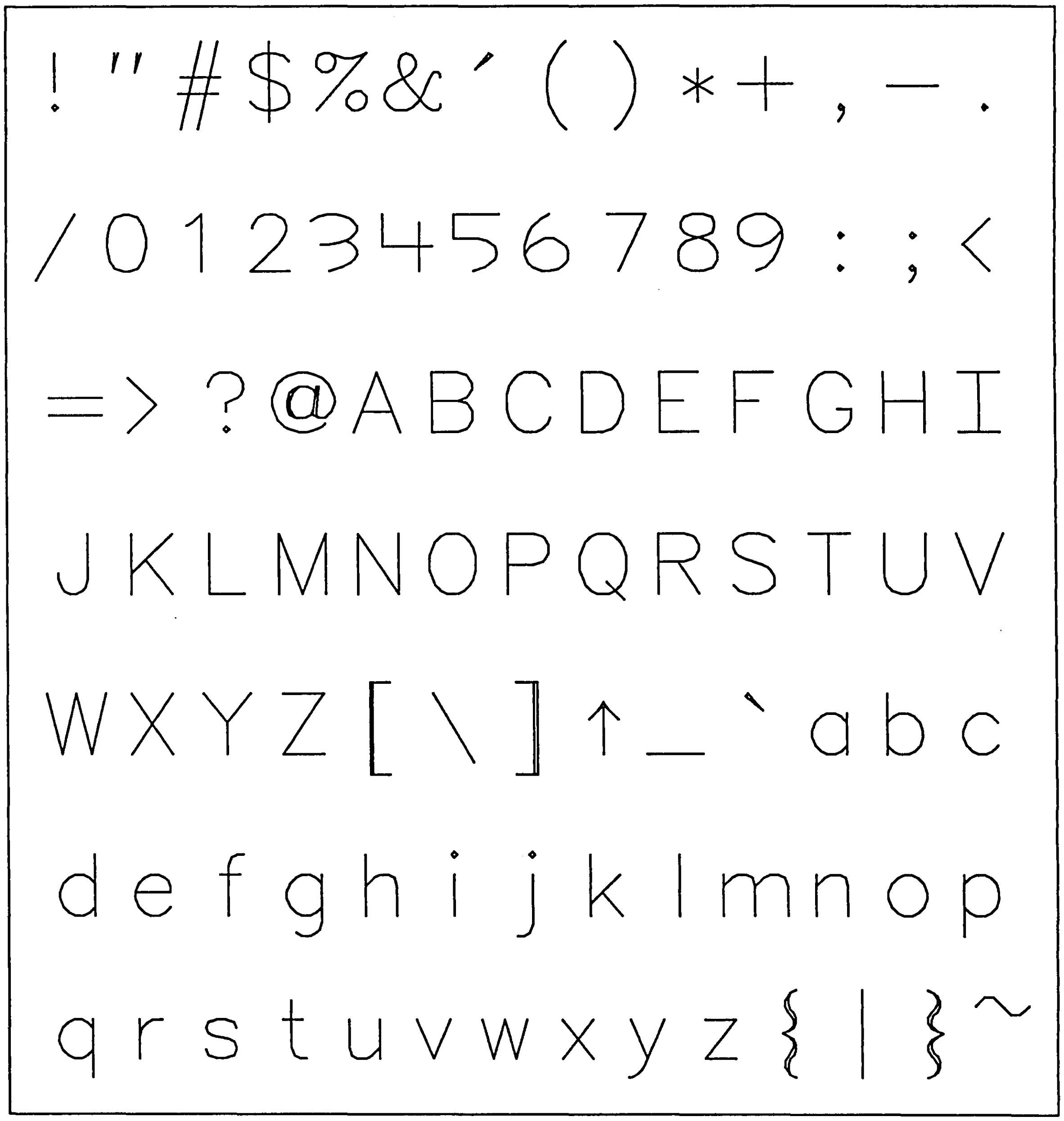

! " # $ % & ' ( ) * + , - .
/ 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I
J K L M N O P Q R S T U V
W X Y Z [\ ] ↑ _ ` a b c
d e f g h i j k l m n o p
q r s t u v w x y z { | } ~

Figure B-5. Milspec Font

33 !	34 "	35 #	36 $	37 %	38 &	39 ´	40 (	41)	42 *	43 +
44 ´	45 −	46 .	47 /	48 0	49 1	50 2	51 3	52 4	53 5	54 6
55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?	64 @	65 A
66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L
77 M	78 N	79 O	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ↑	95 _	96 `	97 a	98 b
99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m
110 n	111 o	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x
121 y	122 z	123 {	124 |	125 }	126 ~					

Figure B–6. Milspec Font ASCII Codes

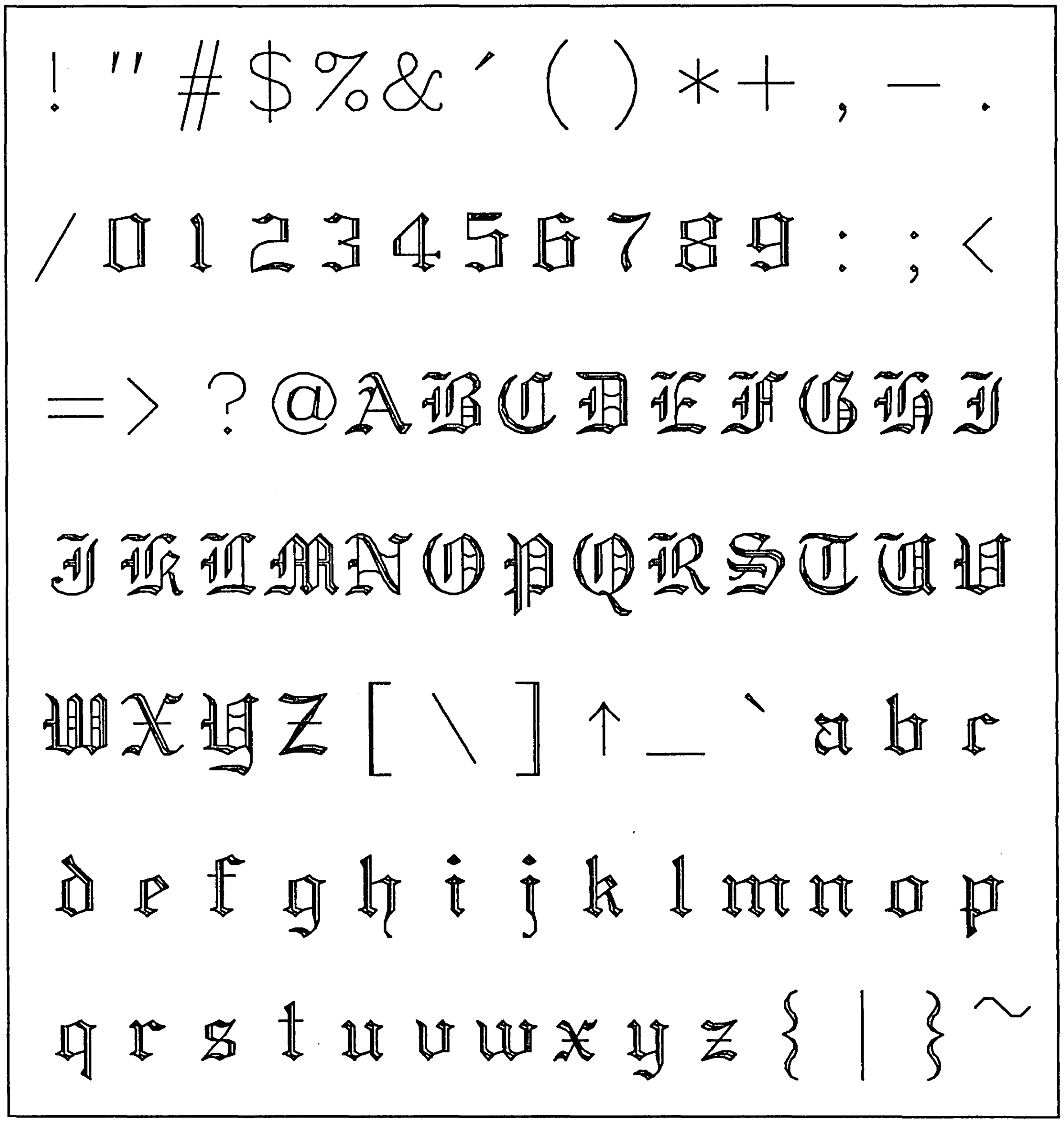


Figure B–7. Gothic Font

33	34	35	36	37	38	39	40	41	42	43
!	"	#	$	%	&	'	(	)	*	+
44	45	46	47	48	49	50	51	52	53	54
,	–	.	/	0	1	2	3	4	5	6
55	56	57	58	59	60	61	62	63	64	65
7	8	9	:	;	<	=	>	?	@	A
66	67	68	69	70	71	72	73	74	75	76
B	C	D	E	F	G	H	I	J	K	L
77	78	79	80	81	82	83	84	85	86	87
M	N	O	P	Q	R	S	T	U	V	W
88	89	90	91	92	93	94	95	96	97	98
X	Y	Z	[	\	]	↑	_	`	a	b
99	100	101	102	103	104	105	106	107	108	109
c	d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119	120
n	o	p	q	r	s	t	u	v	w	x
121	122	123	124	125	126					
y	z	{	|	}	~					

Figure B-8. Gothic Font ASCII Codes

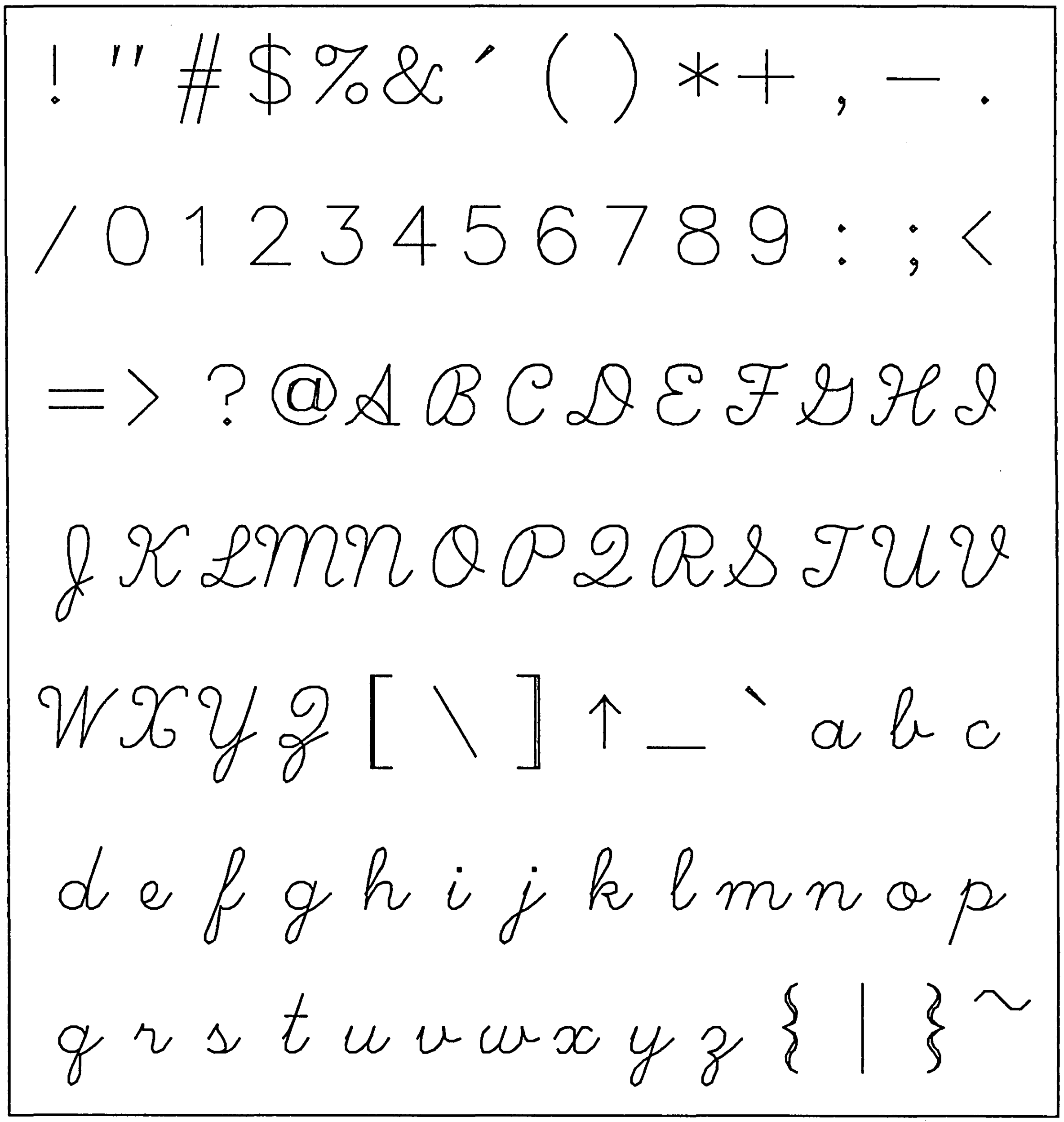

! " # $ % & ' ( ) * + , - .
/ 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I
J K L M N O P Q R S T U V
W X Y Z [\ ] ↑ _ ` a b c
d e f g h i j k l m n o p
q r s t u v w x y z { | } ~

Figure B–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
,	-	.	/	0	1	2	3	4	5	6
55	56	57	58	59	60	61	62	63	64	65
7	8	9	:	;	<	=	>	?	@	A
66	67	68	69	70	71	72	73	74	75	76
B	C	D	E	F	G	H	I	J	K	L
77	78	79	80	81	82	83	84	85	86	87
M	N	O	P	Q	R	S	T	U	V	W
88	89	90	91	92	93	94	95	96	97	98
X	Y	Z	[	\	]	↑	_	`	a	b
99	100	101	102	103	104	105	106	107	108	109
c	d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119	120
n	o	p	q	r	s	t	u	v	w	x
121	122	123	124	125	126					
y	z	{	|	}	~					

Figure B-10. Cursive Font ASCII Codes

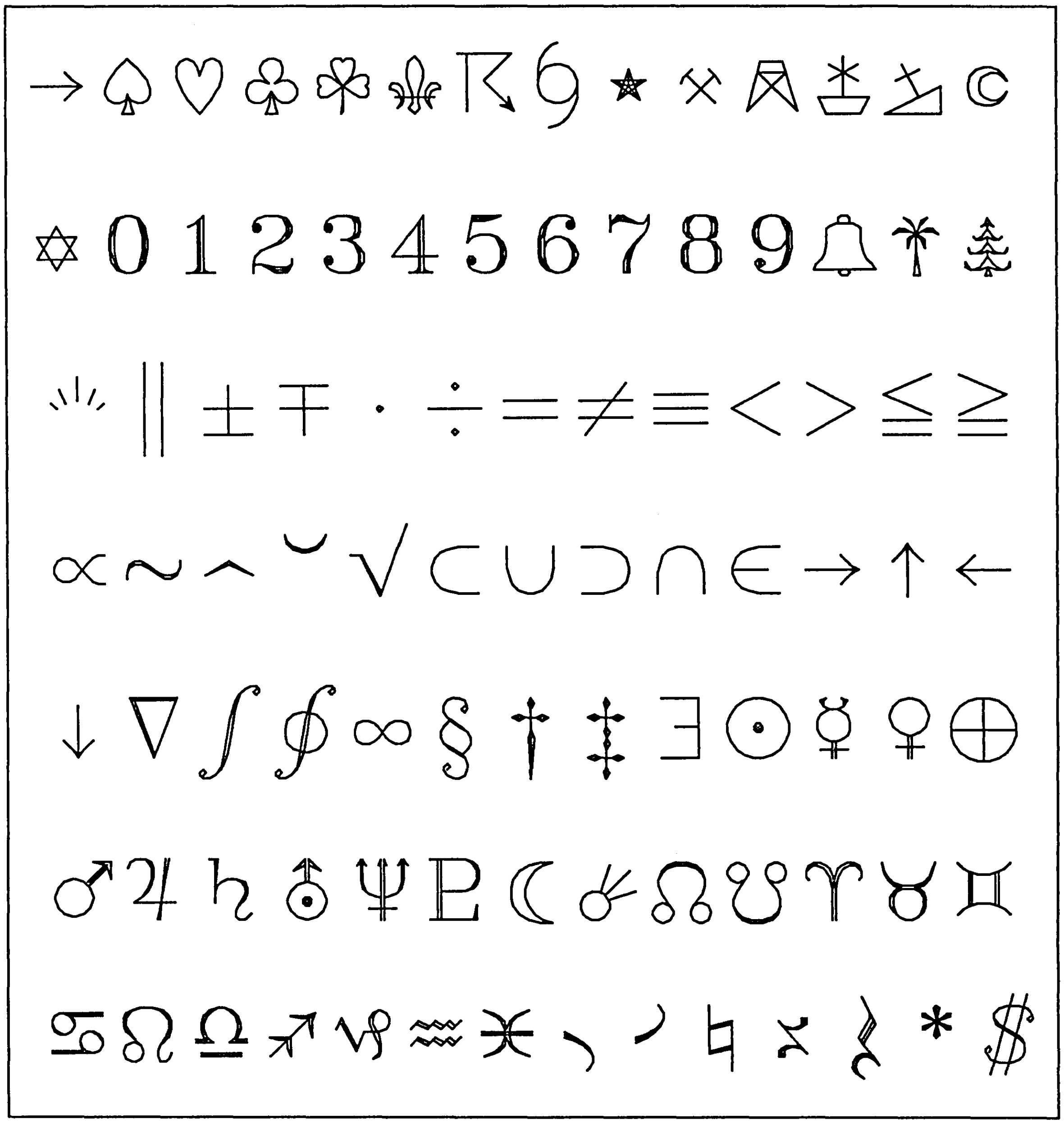


Figure B–11. Symbol Font

33	34	35	36	37	38	39	40	41	42	43
→	♤	♡	♧	♧	♣	R	ɕ	★	⚒	A
44	45	46	47	48	49	50	51	52	53	54
R	⚓	●	✡	0	1	2	3	4	5	6
55	56	57	58	59	60	61	62	63	64	65
7	8	9	🔔	🌴	🎄	⺌	‖	±	∓	·
66	67	68	69	70	71	72	73	74	75	76
÷	=	≠	≡	<	>	≤	≥	∝	~	⌢
77	78	79	80	81	82	83	84	85	86	87
⌣	√	⊂	∪	⊃	∩	∈	→	↑	←	↓
88	89	90	91	92	93	94	95	96	97	98
∇	∫	∮	∞	§	†	‡	∃	☉	☿	♀
99	100	101	102	103	104	105	106	107	108	109
⊕	♂	♃	♄	⛢	♆	♇	☾	☄	☊	☋
110	111	112	113	114	115	116	117	118	119	120
♈	♉	♊	♋	♌	♎	♐	♑	♒	♓	𝄾
121	122	123	124	125	126					
𝄿	♮	𝄪	𝄽	✱	𝄋					

Figure B–12. Symbol Font ASCII Codes

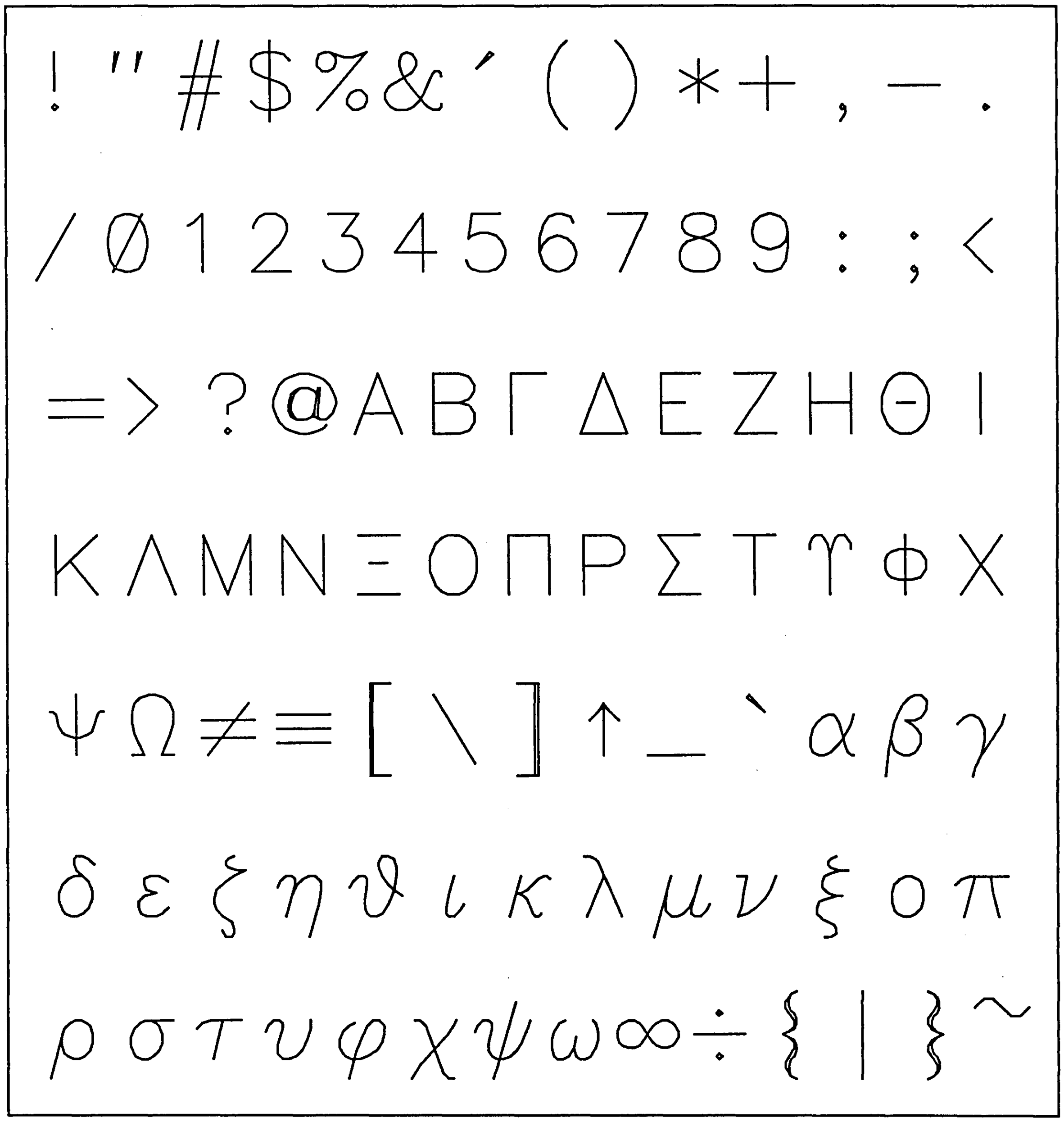


Figure B-13. Greek Font

33 !	34 "	35 #	36 $	37 %	38 &	39 '	40 (	41)	42 *	43 +
44 ,	45 –	46 .	47 /	48 0	49 1	50 2	51 3	52 4	53 5	54 6
55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?	64 @	65 A
66 B	67 Γ	68 Δ	69 E	70 Z	71 H	72 Θ	73 I	74 K	75 Λ	76 M
77 N	78 Ξ	79 O	80 Π	81 P	82 Σ	83 T	84 ϒ	85 Φ	86 X	87 Ψ
88 Ω	89 ≠	90 ≡	91 [	92 \	93]	94 ↑	95 _	96 `	97 α	98 β
99 γ	100 δ	101 ε	102 ζ	103 η	104 ϑ	105 ι	106 κ	107 λ	108 μ	109 ν
110 ξ	111 o	112 π	113 ρ	114 σ	115 τ	116 υ	117 φ	118 χ	119 ψ	120 ω
121 ∞	122 ÷	123 {	124 |	125 }	126 ~					

Figure B–14. Greek Font ASCII Codes

C Batch and Non-Graphical GED

This section explains:

- NGED (Non-graphical GED)
- Redirecting GED input and output

NGED

Graphical GED:

- Only runs on graphics workstations
- Always draws graphics on the screen
- Only runs in the background under Suntools

NGED allows the editor to run in a non-graphical mode. This allows you to run GED:

- Without a graphics terminal
- In the background

This is useful for running a large batch process, such as **hardcopy** or **backannotate**, without having to enter GED.

To run nongraphical GED, type the command:

nged *drawing_name*

Like the **ged** command, this command accepts an initial drawing name as a command line argument.

Redirecting GED Input and Output

GED's input and output can be redirected to or from a file. You can choose to redirect either or both the input and output. If GED or NGED reads from a file, it operates about the same as when reading a file with the **script** command. If GED or NGED writes to a file, it stores the text messages that are also printed on the screen.

In UNIX, use one of the following commands to redirect the input and/or output for GED or NGED:

ged < *input* > *output*

or

nged < *input* > *output*

input	The file that contains the input commands.
output	The file that contains the output commands.

In VMS, use the following command to read the input to GED from a file:

DEFINE/USER_MODE GED$INPUT *INPUT.DAT*

INPUT.DAT is the name of the file that contains the input commands. GED$INPUT is a VMS logical name.

To divert GED's output into a file in VMS, type:

DEFINE/USER_MODE GED$OUTPUT *OUTPUT.DAT*

OUTPUT.DAT is the name of the file that contains the output commands. GED$OUTPUT is a VMS logical name.

To run GED in VMS batch mode, submit a DCL script (similar to the example in Figure C-1) to the batch queue.

```
$   SET DEFAULT DRC9:[MYDIR.SUBDIR1.SUBDIR2]
$   DEFINE/USER GED$INPUT GEDIN.DAT
$   DEFINE/USER GED$OUTPUT GEDOUT.DAT
$   NGED
```

Figure C-1. Sample DCL Script

The following variables are used:

- **DRC9:[MYDIR.SUBDIR1.SUBDIR2]** is the directory where GED runs.
- **GEDIN.DAT** is the input file to GED.
- **GEDOUT.DAT** is the output file for GED.

Index

ValidGED™

Command Reference Guide

January 15, 1989

P/N: 900-00577

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, California 95134
(408) 432-9400 Telex 371 9004
FAX 408 432-9430

MANUAL REVISION HISTORY

Rev	Date	Software Release	Reason for Change
A	1/15/89	GED 9.0	Initial Release

Table of Contents

Preface

The *ValidGED Command Reference Guide* contains an alphabetical listing of GED commands. You can enter GED commands in several ways:

- Type the command in the message window and press Return.

- Use the cursor controller to select the command from the on-screen menu.

- In some cases, press a pre-defined function key.

Command Entry

Whether you use the keyboard, menus, or buttons to enter a command, every command and its arguments are echoed within the message window.

GED is not case-sensitive; it recognizes commands typed in either uppercase or lowercase letters. However, file names and text added to drawings are case-sensitive.

Command Abbreviations

The minimum entry for each command is underlined in the command syntax line. Typing either the full name or the abbreviation executes the same command.

SYNTAX

<u>com</u>mand Return

EXAMPLE

<u>dir</u>ectory Return

Command Syntax

Commands that require arguments and optional entries follow the general syntax shown below.

SYNTAX

command [*operands...*] [*points...*] ... (Return)

operands

Variable arguments for each command are explained below the syntax line. You must substitute the appropriate value for the command argument(s) when you enter the command. The description of the argument provides potential values for the argument. Square brackets surrounding any operand indicate the argument is optional. You do not type the square brackets.

points

Any points required by the command can be entered by pressing the appropriate cursor controller (mouse or puck) button or by typing the coordinates at the keyboard. Coordinates can be entered as (x,y) or (x y).

EXAMPLE

```
hardcopy c *.logic* (Return)
```

Documentation Conventions

Figure 1 lists the documentation conventions used in this manual.

Figure 1. Documentation Conventions

Convention	Meaning	Example
bold font	Literal keyboard input	**set path =**
[optional]	Optional user input; brackets are for document clarity only and should not be entered	[-options]
keyname	Name of key or button the user should press	Return
sans serif italic font	Variable; must be replaced by specific values supplied by the user	*user_name*
pt	Use the cursor controller to select a point on the screen	

GED Commands

This section contains descriptions of all GED commands. Note the following information:

- For all commands requiring a cursor point, the point entry is abbreviated as *pt*.
- For all commands requiring a directory name, the directory variable is abbreviated as *<dir>*. The angle brackets are required.
- The use of ellipsis (...) indicates that the preceding 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. For example: (*pt1 pt2*) ...

add

Adds a specified body to a drawing.

SYNTAX

add *body_name*[.[**body**][.[*version*]]] *pt* [*pt* ...] ...

body_name The name of the body drawing to be added.

body The type of body can be specified, but is not required.

version 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.

Each SCALD directory in the current search stack is searched until the drawing with name *body_name*.body is found. GED does not allow TIME or SIM parts in logic drawings. Similarly, SIM parts cannot be added to TIME drawings, and so on. The **directory** <*> command tells whether any of the SCALD directories in a user's list are of the wrong type for the currently-edited drawing.

To add a body to a drawing:

1 Type:

add *body_name* Return

A copy of the body is attached to the cursor.

This step is optional.

2 If necessary, press the white button to rotate the body.

3 Move the body to the required position in the drawing and press the yellow button.

This step is optional.

4 If necessary, press the yellow button to add another copy of the same body to the drawing and position the copy as described above.

The **add** command remains active until another command is entered or the semicolon 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 Return.

add remembers the body that you added last and attaches it to the cursor if you enter the **add** command and press any button. The cursor must be within the drawing area.

The **window** and **zoom** commands can be nested within the **add** command.

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. If you receive this error message:

- Check the spelling to make sure that you typed the name of the body correctly.
- Make sure that the required component library is specified. **library** Return lists the available libraries.
- Make sure that the specified part is included in the library. List the contents of the library directory.

EXAMPLES

```
add ls74
```

Adds version 1 of the part LS74 to the drawing.

```
add addr..2
```

Adds version 2 of the part ADDR to the drawing.

See Also

directory	Lists the contents of a library.
library	Adds a library to the active search list.
replace	Substitutes one body for another.
version	Selects an alternate version of a body, if available.

arc

Draws arcs and circles.

SYNTAX

<u>ar</u>c *pt1 pt2* [*pt3*] ...

The **arc** command facilitates bodies that contain arcs and circles. To draw an arc:

1 Type:

```
arc Return
```

and enter two points to indicate the ends of the arc.

A circle is drawn with the two initial points forming its diameter. A flexible wire running between the initial points is attached to the cursor.

Select the semicolon (;) or the **arc** *command to place a circle on the drawing rather than an arc.*

2 Position the arc as required and enter a third point to determine the curvature of the arc. The arc passes from the first point, through the third point, and ends at the second point.

- Press the yellow button to position the curve of the arc at the nearest screen pixel.
- Press the white button to position the curve of the arc at the nearest grid intersection.

To make a semi-circle, place the third point anywhere on the circle itself.

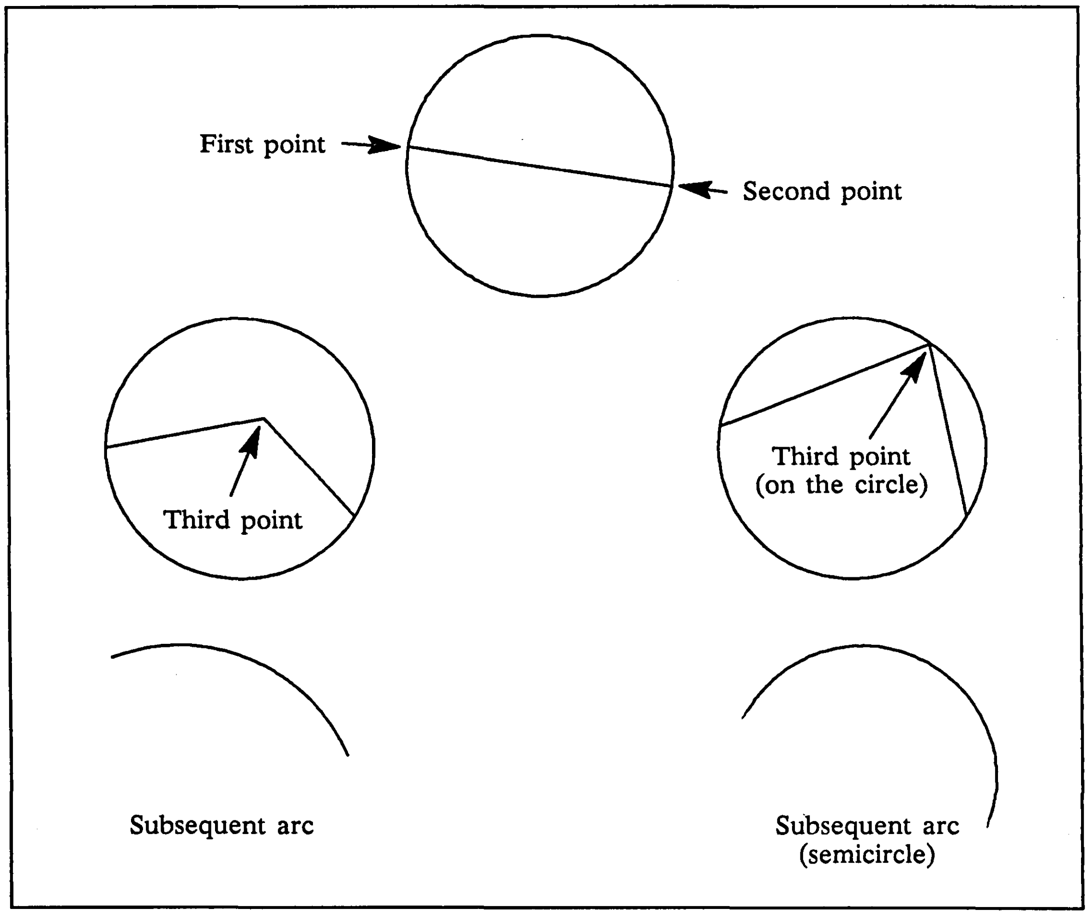


Figure 2. Creating an Arc

See Also

circle	Draws circles and arcs using a center point and a radius rather than the diameter to specify the size of the circle or arc.

assign

Assigns a GED command or operation to a programmable function key.

SYNTAX

assign *function_key "command text"*
assign *key_name "command text"*

function_key	After you type **assign**, you can press the function key to be assigned and then type in the command text to be assigned to that key.
key_name	Rather than pressing the function key, you can type the function key name and then enter the command text to be assigned to that key.
"command text"	The GED command and its arguments to be assigned to the function key.

The **assign** command assigns text, such as a GED command, to a function key so that you can press the specified key instead of typing the text. 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 GED. These values are stored in the *softkeyassign* file in the system editor directory.

The current function key assignments can be displayed with the **show keys** command. The soft key assignments can be tailored for the entire system by defining additional function key values.

The recommended method for changing the function key assignments is to place **assign** commands in individual *startup.ged* or script files. These values then take effect whenever you use GED. Alternately, you can issue the **assign** command during an editing session to make function assignments that last only during that session.

To use the **assign** command:

1. Type:

   ```
   assign
   ```

You can also type assign *and the function key name (for instance,* LF2*).*

2. Press the required function key.

3. Type the GED command to be assigned.

   The command and its arguments must be enclosed in quotation marks.

4. Press Return.

You can also define keys by putting **assign** statements in your *startup.ged* file. You identify the function key by typing the name of the key. The command text must be enclosed in quotation marks. A Return is automatically appended to the end of the assigned string.

The name of the key corresponds as closely as possible with the text printed on the keyboard. Letters can be either uppercase or lowercase.

On SCALDsystems, the keys are named:

- LF2 - LF9
- RF1 - RF12
- TF1 - TF6

On Sun workstations, the keys are named:

- F1 - F9
- R1 - R9

The L1 - L10 keys cannot be assigned.

The number keys on the numeric keypad are preceded with K. *The hyphen on the numeric keypad is named* K_. *The names of keys with two words are joined into one word with an underscore character. For example,* Next Screen *becomes* Next_Screen.

On VAX workstations, the keys are named:

- F6 - F20
- K1 - K9

On the PC AT workstation, the keys are named F1 - F10.

To increase their usefulness, various control keys can be pressed in combination with function keys to create more command possibilities. These control keys and abbreviations are:

SHIFT	**S**	
CONTROL	**C** or **CTRL**	
ALT	**A**	(PC AT)
SUPER (LF10)	**SU**	(SCALDsystem)
HYPER (LF11)	**H**	(SCALDsystem)
META (LF12)	**M**	(SCALDsystem)

To use a key combination with the **assign** command, type the name of the modifier key followed by a dash and the name of the function key.

EXAMPLES

```
assign (press LF6) "display 2.0"
```

This example assigns the command **display 2.0** *to the LF6 function key on a SCALDsystem.*

```
assign K6 "display 2.0"
```

This example assigns the command **display 2.0** *to the number 6 key on the numeric keypad of a VAX workstation. This version uses the key name (K6) to identify the function key.*

```
assign SHIFT-LF2 "window 2.0"
```

This example assigns the command **window 2.0** *to the shifted LF2 key (the LF2 key is pressed simultaneously with the shift key) on a SCALDsystem.*

See Also

Section 1 (*GED Overview*) of the *ValidGED User's Guide* describes the default function key assignments.

auto

Performs the global addition or deletion 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. **auto undot** automatically removes all dots from the drawing except at the intersections of four wires.

SYNTAX

<u>au</u>to <u>d</u>ots
<u>au</u>to <u>p</u>ath
<u>au</u>to <u>u</u>ndot

dots

Places a dot at each wire 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.

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.*

path

Makes 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.

undot

Removes all dots from the drawing except those at the intersections of four wires.

See Also

set

Allows you to specify the style of dots to be displayed (open or filled).

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains more information about the PATH property.

backannotate

Annotates designs with physical information from the Packager.

SYNTAX

backannotate [*annotation_file*]

annotation_file

GED reads a schematic 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, or you can 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:

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 *ValidPackager Reference Manual* and the description of the **set** command in this manual more information.

This step is optional.

2 If necessary, rename *pstback.dat* to *backann.cmd*. The file *pstback.dat* is generated by the Packager. The *backann.cmd* file must be in the current directory.

3 Enter GED and type:

```
backannotate [Return]
```

If you did not rename *pstback.dat* to *backann.cmd*, specify the name of the backannotation input file on the command line. For example, enter:

```
backannotate pstback.dat [Return]
```

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.
set	Several options control the placement of pin numbers on the drawing.

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains more information about properties, and Section 6 *(Adding Physical Information)* discusses backannotation and physical design information.

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.

GED supports IEEE bubbles. Following standard GED usage, objects that look like IEEE bubbles in body drawings are interpreted as bubbles in connectivities.

See Also

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains information about defining bodies.

busname

Places equally-spaced, single-bit vectored signal and pin names on the drawing.

SYNTAX

busname *bus_name pt1 pt2*
busname *pt1 bus_name pt2*
busname *pt1 pt2 bus_name*

bus_name	A simplified SCALD signal syntax name, such as A<0..3>.
pt1 pt2	The two points specify the location of the first two names. Remaining names are placed automatically, with spacing between each name defined by the first two points.

The **busname** command provides a convenient shorthand for naming signals in buses, or pins on bodies, whose names differ only in array subscripts. The name you specify is a simplified SCALD signal name, such as:

ADDRESS<7..0>

GED reads the system-wide *config.dat* file to determine the array subscript string, such as two dots (..) or a colon (:). Bus bit ordering (left-to-right or right-to-left) is ignored. Table 1 illustrates bus name syntax and the resulting signal names.

Table 1. Bus Name Syntax

Bus Name	Signal Name
`A<3..0>`	`A<3>, A<2>, A<1>, A<0>`
`A<0..3>`	`A<0>, A<1>, A<2>, A<3>`
`A<0>`	`A<0>`
`A<7..0:2>`	`A<7>, A<5>, A<3>, A<1>`

If the array subscript character is a colon (:), the field separator becomes a double colon (::). For example, `A<0:7::2>` becomes:

```
A<0>, A<2>, A<4>, A<6>
```

GED draws a bright line between the name and the wire to which the name is attached to verify that the signal names are attached correctly.

EXAMPLE

You can draw a 4–bit counter body with outputs shown as four separate, evenly spaced wires. To add signal names:

The \I *refers to the signal interface property. For more information on signal properties, refer to the* SCALD Language Manual.

1 Type:

```
busname A<3..0>\I
```

GED attaches the string A<3>\I to the cursor.

2 Place the name at the top wire and press the yellow button to enter the point.

GED attaches the string A<2>\I to the cursor.

3 Position the second name and press the yellow button.

This name and the remaining two signal names are placed on the drawing.

The bright lines are displayed so you can verify the attachment of signal names to the drawing.

See Also

signame

Attaches signal names to wires or pins.

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains information about properties and signal names.

The *SCALD Language Reference Manual* contains information about signal name syntax.

change

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

SYNTAX

change *pt ...*
change *group_name ...*

pt

You can choose text strings to change by pointing with the cursor and pressing the yellow button. You can select as many strings as necessary.

group_name

You can place strings to be edited in a group and then specify the group by name or by pressing the white button.

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

To use the **change** command:

1 Select the **change** command from the menu or type:

```
change Return
```

2 Select the text string(s) or group to be edited.

GED replaces the status line at the top of the display with the first line of text to be edited. You can now use the GED line editor or the system editor to change the selected text strings.

To use the system editor, enter Control-V. GED writes all the selected text strings to a file, one string to a line, and accesses the system editor. Use the editing functions to move around the file and make the required changes. If you add notes to the file, they are placed on the drawing below the last note you changed. You cannot delete lines from the file; use the **delete** command to remove the text strings from the GED drawing instead.

The system editor can be set with the **set user_editor** command. The default is *vi* on UNIX systems and EDT on VMS systems.

When you are finished, exit from the system editor. The changed text is repositioned on the drawing. Refer to the appropriate manual for more information about the system editor.

Note: To run the system editor from NGED, enter the **change** command and a group name to be changed:

```
change a Return
```

The **change** command will automatically access the system editor.

Using the GED Line Editor

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

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

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

Table 2. Line Editor Functions

Keys	Result
Control – A	Moves the cursor to the beginning of the line.
Control – B	Moves the cursor backward one character.
Control – D	Deletes one character to the right of the cursor.
Control – E	Moves the cursor to the end of the line.
Control – F	Moves the cursor forward one character.
Control – H	Deletes one character to the left of the cursor.
Control – K	Deletes the remainder of the line (right of the cursor).
Control – Q	Displays the **help** file for the line editor.
Control – R *character* Return	Searches to the left of the cursor for the specified character.
Control – S *character* Return	Searches to the right of the cursor for the specified character.
Control – U *n* Return *cmd*	Repeats the command *n* times. If no number is given, the default is four.
Control – V	Place all selected text strings (that have not yet been edited) into a file and run the system editor on that file.
Control – X	Repositions the text currently on the edit line and displays the next line of text to be edited. When all text has been edited, exits the line editor.
Control – Z	Aborts changes to the text currently on the edit line.

check

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

SYNTAX

```
check
```

The **check** command adds PATH properties 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. **check** looks for:

- 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
- Wires that come close to but do not contact pins
- Duplicate PATH properties
- Unmarked wire connections
- Wires overlapping a body
- Missing TITLE and/or ABBREV properties
- Bodies that are placeholders
- Pins located at the origin (0,0) in BODY drawings

- Multiple dots at the same location
- Hard properties with the ? value (placeholders)
- Objects partially outside the GED drawing boundaries
- Wires connecting the pins of a two-pin body.

check lists each detected error. 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**.
set	The **check_on_write** option determines whether **check** is automatically invoked every time you write a drawing.

circle

Adds circles and arcs to a drawing.

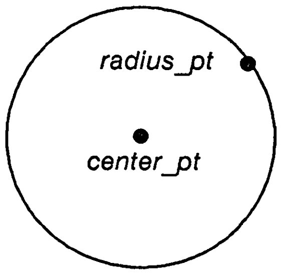
SYNTAX

circle *center_pt radius_pt* [*end_pt*] ...

center_pt	The center point of the circle.
radius_pt	The second point of the circle or arc. This point is used to determine the length of the radius.
end_pt	This third point, along the circumference of the circle, is only used when drawing an arc. This point determines the length of the arc.

The **circle** commandcan be 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:

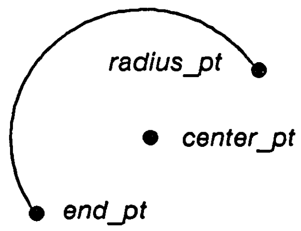


1 Select **circle** from the menu.

2 Select a point as the center of the circle.

3 Select a second point to determine 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:

```
circle
```

and select the 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 Position the cursor along the circumference of the circle and specify an ending point to determine the length of the arc. The arc is drawn from the starting point counterclockwise to the ending point.

See Also

arc Facilitates the creation of arcs on a drawing. Uses diameter rather than center point and radius to define the size of the circle.

connect

Provides a general interprocess communication facility for GED. (Not available on PC AT system).

SYNTAX

```
connect program
connect off
connect show
led
```

program — **connect** provides a general interprocess communication scheme for GED on SCALDsystem. A *program* connected to GED can send commands that GED can execute. Several Valid tools have built-in connections to GED. For example, **simulate** starts the split-screen Simulator; **connect simulator** connects to a Simulator process that is currently running in a different window.

off — Causes GED to disconnect the currently-connected process.

show — Lists the connections GED currently knows.

led — A synonym for **connect led.**

The connected process has commands that allow you to perform queries about signals and bodies in the currently edited drawing. For example, LED has the **show net** command. GED searches its database and reports the results to the connected process and in the GED drawing area.

The names of programs that can be used by **connect** are established by placing **connect** command lines in the system-wide startup files. Connections to LED, CONCORDE, ValidSIM, and COMPARE are specified in the system-wide *startup.valid* file. You must enter **connect** commands in both GED and the other process. If the processes fail to establish a connection within about 30 seconds, a message is displayed. GED can only connect to one process at a time.

EXAMPLE

In *startup.valid*, the command

```
connect simulator unix "/tmp/gedsimS"
```

allows GED to connect to a program named SIMULATOR using /tmp/gedsimS. This command is specified by the system manager and identifies the UNIX socket, the *discipline*, and the number of the GED window. The discipline is platform-specific:

- UNIX is on SCALDsystem only.
- MSPMPX is on Sun3 and Sun4 only.
- VMSAD is on VAX only.

If you enter the **connect sim** command in GED and the **connect ged** command in the Simulator, the two programs can connect together. The Simulator **open** command uses this interface and allows you to select signals in the GED drawing.

copy

Copies objects, properties, and groups in the current drawing.

SYNTAX

```
copy [number] source_pt destination_pt ...
copy [number] group_name destination_pt ...
copy property_pt destination_pt attach_pt ...
```

number	The number of copies to place on the drawing. After the first copy is placed, the remaining copies are automatically added to the drawing. The second copy is offset from the first copy by the same distance as the first copy from the original. You can use this feature to copy single items and groups.
source_pt	Identify an object to copy.
property_pt	Identify a property to copy.
group_name	You can choose a group to copy by using the cursor controller and the white button, or you can type in the single–letter group name to identify the group.
destination_pt	The position of the new copy.
attach_pt	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 wire):

1 Select **copy** from the menu.

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

- The yellow button picks up a copy of the object at the grid point nearest the cursor.
- The blue 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 and wires.

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

- The yellow button places the copy on the grid point nearest the cursor.
- The blue 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** or **select** command to define a group.

2 Select **copy** from the menu.

3 Move the cursor to the group to be copied and press the white button. This selects the

nearest group and attaches it to the cursor relative to the cursor's position when the button is pressed.

or

Type the single-letter *group_name* and press Return.

4 Move the cursor to the location for the copy and press the yellow button. Groups of properties are not copied.

When you copy a group of objects containing properties, a warning message is displayed. When applicable, properties attached to objects are copied with the group.

To make *multiple copies*:

1 Issue the **copy** command and enter a number to specify the number of copies to be made.

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

- The yellow button selects an object.
- The white button selects a group.

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

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

To copy *properties*:

1 Select **copy** from the menu.

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

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

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

You cannot copy default body properties, pin properties, or 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. If a default body property on a body was changed, a copy of the body contains the changed value.

The **window** and **zoom** commands can be nested within the **copy** command.

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 then be copied).
select	Defines a group of objects (which can then be copied).

cut

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

SYNTAX

cut *pt*
cut *group_name*

pt

Selects an object to cut.

group_name

You can choose a group to cut by using the cursor controller and the white button, or you can type in the single-letter group name to identify the group.

The **cut** command, in conjunction with the **paste** command, allows objects and groups to be copied from one drawing to another. Use the **cut** command to place the specified object or group into a *cut buffer*. The cut buffer can contain only one group or object.

1 Type:

```
cut
```

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

or

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

The **cut** command highlights the selected object or group and also displays the number of bodies, wires, dots, circles, and notes from the group 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 the PATH property, properties generated by the **pinswap**, **section**, and **backannotate** commands, and pin properties. Wire properties are copied when a wire is cut to allow signal names to be transferred to the new drawing (unnamed signal names are not copied).

See Also

copy	Makes copies of objects and groups in the same drawing.
paste	Transfers objects from the cut buffer to the specified locations in the current drawing.

delete

Removes objects from a drawing.

SYNTAX

delete *pt ...*
delete *group_name ...*

pt

Selects an object to delete. To delete an object, point to any part of the object and press the yellow button. **delete** removes the object nearest to the cursor.

group_name

You can choose a group to delete by using the cursor controller and the white button, or you can type in the single-letter group name to identify the group. The group nearest the cursor is deleted.

Default properties on bodies and pin number properties 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.

diagram

Changes the name of the current drawing.

SYNTAX

diagram [*<dir>*] [*drawing_name*] [. [*type*] [. [*version*] [. [*page*]]]]

<dir>	The directory name where the drawing resides. If no directory is specified, the current directory is used.
drawing_name	The new name of the drawing. If no drawing name is specified, the current drawing name is used.
.type.version.page	The drawing type, version number, and drawing page number are optional. If not included, the current drawing type is used, and the version and page number default to 1.

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:

   ```
   diagram
   ```

   and the new name of the drawing.
3. Type:

   ```
   write
   ```

   to save a copy of the drawing under its new name.

The **diagram** command is also used to change the type of a drawing. For instance, when changing a SIM drawing to a TIME drawing, GED substitutes primitives from the TIME library for the SIM primitives where possible on the drawing.

EXAMPLE

```
edit test1.logic
diagram finaltest.logic
write
```

directory

Lists the contents of SCALD directories.

SYNTAX

directory [<*dir*>] [*drawing_name*] [. [*type*] [. [*version*] [. [*page*]]]]

<*dir*>	List the specified directory. If no directory is specified, the current directory is used.
drawing_name	List the specified drawing.
.type.version.page	Unless you specify type, version, and page parameters, the **directory** command displays just the drawing name. You can also list drawings by type or select only certain versions or pages to list.

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. After a screenful of text, the following prompt is displayed:

```
More (ync)
```

y *or* Return	Yes. Present more information.
n *or* q	No. Do not print any more output.
c	Continue. Print the entire message output without pausing for page prompts.

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

EXAMPLES

```
directory
```

Lists all drawing names in the current SCALD directory.

```
directory <*>
```

Lists all active SCALD directories (but no drawing names).

```
directory <time>*
```

Lists all drawing names (parts) in the TIME library.

```
directory ls*
```

Lists all drawing names beginning with ls *in the current SCALD directory.*

```
directory *.body*
```

Lists all BODY drawings in the current SCALD directory.

```
directory <*>*
```

Lists all drawing names in all active SCALD directories and libraries.

```
directory *.*
```

Lists the name, type, and version of each drawing in the current SCALD directory.

See Also

ignore	Excludes the specified directory or library from the active search list.
library	Specifies the component library to be accessed. With no arguments, lists all available libraries.
use	Specifies the current working directory or library on the active search list.

Section 2 *(The Editing Environment)* of the *ValidGED User's Guide* describes SCALD directories and their operation.

display

Changes the way objects are displayed on a drawing.

SYNTAX

display *option "group_name"*
display *option pt ...*

option

Several options can be specified with the **display** command. You can change the display of:

- Properties
- Text size
- Text justification
- Wire size
- Dots

Individual options are listed following the syntax explanations.

"group_name"

You can choose to change the display of a group by using the cursor controller and the white button, or you can type in the single-letter group name to identify the group. The group name specified must be quoted. The group can contain any type of object.

pt

You can choose to change the display of a single object by using the cursor controller and the yellow button. The **display** command selects the appropriate object to change.

Group names, options and point entries may be included in any order and in any combination, except that the first argument **must** be a command option.

Properties

The options **name**, **value**, **both**, and **invisible** determine the way properties are displayed on the drawing. Although a property consists of a name and value pair, usually 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, enter the command and 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.

EXAMPLE

You can define a body with a default property SIZE = 1B and suppress the display of the property with the **display invisible** command. When that body is added to a logic drawing, the property SIZE = 1B does not appear. Use the **display value** option and point to the location of the property to make it appear on the drawing. The command **show properties** displays the name and value of all properties (including invisible ones) on the drawing.

Text Size

The **default** and *scale_factor* options determine the size of text displayed on the drawing.

default

Displays text on the drawing at the default size, 12 characters per inch.

scale_factor

Enlarges or reduces the size of the text on the drawing by the amount specified by *scale_ factor*. Negative scale factors are treated as inverses to allow a simple method of undoing a text size change. If you select a *scale_factor* larger or smaller than GED's limit, GED sets the size of the text to the maximum or minimum value.

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 hardcopy of the drawing without taking up more space than necessary.

To change the size of a string of text:

1. Type:

   ```
   display scale_factor
   ```

   to indicate the factor by which the size of the currently displayed text is to be multiplied.

2. Use the cursor to select the text string to change.

The **set size** *command allows you to change the default size of added text.*

To return the text to the default size, type:

```
display default
```

and point with the cursor to specify the text.

Text Justification

By default, all user-added text is left-justified.

left_justified	Left-justifies selected text strings (default).
right_justified	Right-justifies selected text strings.
center_justified	Center-justifies selected text strings.

When the vertex of a right-justified string is selected (blue button), the cursor attaches to the right end of the string.

Wire Size

You can change the way an existing wire appears on a drawing.

heavy	Makes the wire thicker so it looks like a bus.
thin	Returns a heavy wire 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 net changes. In a BODY or DOC drawing, only the wire segment specified by the cursor is changed.

Dots

The **filled** and **open** options change the display of dots already added to the design.

filled	Displays solid dots.
open	Displays open dots (default).

Open dots scale when the **window**, **zoom**, or **scale** command is used; filled dots do not. The **set dots_filled** command makes dots added to the drawing filled by default.

EXAMPLES

```
display invisible "a"
```

Makes all properties in group A invisible. The quotation marks around the group name are required.

display both *pt*

Displays the name and the value for the selected property.

display 2 *pt*

Enlarges the selected text by two times.

display .5 *pt*

Makes the selected text half as large.

display −4 *pt*

Reduces the selected text four times. This is the same as **display .25.**

display filled *pt*

Makes the selected dot solid.

See Also

find	Allows you to define a group of text to be manipulated by the **display** command.
group	Allows you to define a group by drawing a closed polygon around the required objects.
select	Allows you to define a group of objects with a stretchable rectangle.
set	Allows you to change the default options used by GED. The **set** options that affect the same drawing elements as the **display** command are: • **dots_filled** • **dots_open** • **left_ justified** • **center_ justified** • **right_ justified** • **prop_display** • **size**
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

Adds dots to drawings to indicate connection points.

SYNTAX

<u>do</u>t *pt* ...

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

auto	**auto dot** places a dot at all connection points in a logic drawing. **auto undot** automatically removes all dots except at the intersections of four wires.
display	The **filled** and **open** options change the style of selected dots displayed on the drawing.
set	**set dots_filled** and **set dots_open** change the default dot type.
show connections	Temporarily displays all connection points in a logic drawing.

echo

Displays messages from a script file on the GED screen. This allows you to track the progress of a GED script, and is useful for debugging purposes.

SYNTAX

<u>ec</u>ho *message_line*

message_line	The message to be displayed.

edit

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

SYNTAX

```
edit [<dir>] [drawing_name] [. [type] [. [version] [. [ page]]]]
edit pt
```

<dir>

Search for the drawing in the specified SCALD directory. If no directory is specified, each directory in the list is searched until a drawing of that name is found.

dwg

The name of the drawing to edit. 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 when you write the drawing. If the drawing name is omitted, GED uses the name of the current drawing.

.type.version.page

If the drawing type is not specified, GED uses the default type specified by the **set push_type** command, initially set to LOGIC. To edit another type of drawing, include the drawing type after the drawing name.

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 drawing types, such as TIME, and SIM, can also have multiple versions and pages.

pt

The **edit** command allows you to examine the drawings associated with bodies on the screen. By default, the LOGIC drawing of a hierarchical body is edited when you select the body from the current drawing. You can also change the default drawing type by using the **set push_type** command.

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 yellow button. The current drawing is placed in temporary storage, and the drawing SUBTRACTOR.LOGIC is displayed and can be edited.

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 edit the first drawing again, **edit** displays the modified version from temporary storage.

EXAMPLES

```
ed test
```

Displays drawing test.type.1.1. *The* type *is determined by the* **set push_type** *command. The default is LOGIC.*

```
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 (the one named on the status line).

See Also

get	Replaces the current drawing with the version stored on the disk.
return	Returns to the previously-edited drawing.
set push_type	Changes the value of the default drawing type.
show history	Lists the drawings that have been edited during the current GED session.

endsim

Terminates a session with the split-screen simulator.

SYNTAX

<u>en</u>**dsim**

The **endsim** command terminates a session with the split_screen simulator. This is necessary if you need to **section** or **pinswap** parts.

See Also

simulate Accesses the split_screen simulator.

error

Locates and displays each error detected by the **check** command.

SYNTAX

error

The **error** command steps through the errors found by **check**. It 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.

exclude

Removes items or groups from a group.

SYNTAX

<u>ex</u>clude *pt* [*option*] ...
<u>ex</u>clude *group_name* [*option*] ...

pt	To remove individual objects from the current group, press the yellow or blue button. To remove previously-defined groups from the current group, enter the single-letter group name. To remove the contents of the entire group, press the white button.
group_name	The name of the current group. Any objects you specify are removed from the current group. Specify the current group by entering the single-letter *group_name*. If you do not specify the group, the most recently generated group is used. The *group_name* must be included on the same line as the command.
option	Several optional flags allow you to remove *types* of objects from a group:
<u>bo</u>dies	Remove all bodies from the group.
<u>co</u>nnections	Remove all body pins (but not the body origins) from the group.
group_name	Press the white button or enter the single-letter group name to exclude a previously-defined group from the current group.
<u>ne</u>ts	Remove all wires from the group.

properties	Remove all properties from the group.
wires	Remove all wires from the group.

See Also

find	Defines a group of items that match a specified pattern.
group	Allows you to define a group by drawing a closed polygon around the required objects.
include	Adds items to a group.
select	Provides a stretchable rectangle to specify the boundaries of a group.

exit

Allows you to leave the editor.

SYNTAX

exit

The **exit** command allows you to leave GED. 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 the 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

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

SYNTAX

<u>fil</u>enote *filename pt*

filename	The name of the text file to add to the drawing.
pt	The position in the drawing to add the text.

When the text file is added, each line in the file is converted into a note that can be individually moved, copied, deleted, or changed. Empty lines in the file are ignored. To include a blank line in the note, type a space on the line in the file.

See Also

note	Adds individual lines of text to a drawing.

find

Searches the current drawing and places all objects that match a specified pattern into a group.

SYNTAX

find *pattern*

pattern

A given pattern to match in the current drawing. The *pattern* can match:

- Body names
- Notes
- Property names
- Property values
- Signal names

You can use wildcard 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 uppercase and lowercase alphabetic characters in the pattern.

The command assigns all matching items to a group. The number of items in the group is displayed on the screen. GED operations such as **paint**, **show**, **delete**, and **display** can also be performed on the entire group.

All items found with the command are placed in a list. You can step though the list items using the **next** command. This command places an asterisk next to each item on the display so it can be changed or deleted.

EXAMPLES

```
find path=*
```

Locates all path properties.

```
find ls00
```

Locates all LS00 components on the drawing.

```
find un$*
```

Locates all unnamed signals.

See Also

exclude	Removes objects or groups from a group.
include	Adds objects or groups to a group.
next	Steps through the list of items located by **find**.

format

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

SYNTAX

format *text_file* Return *new_drawing_name*

text_file The name of the text file.

new_drawing_name The name of the new drawing.

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 Return.

3 Type the name of the new DOC drawing.

The text file can contain references to 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 inserted in the text file to mark the location of each drawing.

The **format** command can accept the output of text processors such as nroff (UNIX) or Digital Standard Runoff (VMS). It cannot accept output from typesetting programs such as troff (UNIX).

GED reads the named drawing, reduces it to its basic components, and 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 inches 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.

EXAMPLE

```
format design.dat Return
specification
```

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

See Also

filenote	Allows you to add a text file to a GED drawing.
scale	Includes a drawing into specified amount of space.
smash	Breaks a body into the separate objects that define it.

Section 7 *(Creating Mixed Text and Graphics)* of the *ValidGED User's Guide* contains more information about creating mixed text and graphics documents.

get

Replaces the current copy of a drawing with the version stored on the disk.

SYNTAX

```
get [[<dir>] [drawing_name] [.[type] [.[version] [.[ page]]]]]
```

<dir>	Retrieve the drawing from the specified SCALD directory. If no directory is given, each directory in the list is searched until the specified drawing is found. If the drawing is not found in the specified SCALD directory, the new drawing is assumed to belong to the current SCALD directory.
drawing_name	The name of the drawing to retrieve and display.
.type.version.page	The type, version number, and page of the specified drawing.

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 perhaps modified) version in GED. **get** is useful if, while editing a drawing, you want to discard current work and go back to the previous version.

To read in the disk copy of the current drawing, type:

```
get [Return]
```

See Also

edit	Displays a drawing to be edited.
remove	Deletes a selected drawing.
return	Returns to the previously-edited drawing.

grid

Alters the way the grid is displayed.

SYNTAX

grid *option* ;

GED uses a grid of locations to help you place objects and ensure alignment and connections. The purpose of the grid is to help you produce neat, attractive schematics and to facilitate the connection of wires to each other and to pins.

The **grid** command is used to specify the way the grid is displayed. The current values of the grid spacing are displayed on the status line at the top of the screen.

option	Individual options are listed below. End the **grid** command line with a semicolon (;) or by selecting any other command on the menu.
Return	Toggles the grid on and off.
on	Displays the grid on the screen.
off	Turns off the displayed grid.
grid_size	Specifies, in defined units of measure, the separation of the grid lines. The default size for editing LOGIC, TIME, and SIM drawings is 0.1, or one-tenth of an inch. The *grid_size* (a real number) must be a multiple of 0.002 inches, which is the smallest possible grid separation. Table 3 shows the default grid spacings.

Be extremely careful when changing the grid size. Bodies could be placed off grid and then, if the grid size is again changed, wires might not be connected even when they appear to be. This is also why the blue button shoud be used whenever possible to connect wires to pins and other vertices.

Table 3. Default Grid Spacing

Grid Type	BODY	DOC	Other
Decimal	0.05	0.166	0.1
Fractional	0.0625	0.208	0.125
Metric	1.25	4.15	2.5

"Other" drawing types include LOGIC, TIME, and SIM.

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.

dots

Displays the grid as dotted lines.

lines

Displays the grid as solid lines.

In decimal spacing, 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, GED uses 508 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.

Table 4 shows the default grid values.

Table 4. Default Grid Values

Grid Type	Units/inch	Units	Minimum Spacing
Decimal	500	inches	0.002
Fractional	400	inches	0.0025
Metric	508	mm	0.005

See Also

set	Changes the default values used by GED.

group

Combines selected objects into a group.

SYNTAX

group [*group_name*] [**all**] *pt* ...

group_name

The name of the group; do not enclose the group name in quotation marks. If you do not assign a group name, a single-letter name is assigned by GED.

all

Include the entire drawing in a group.

pt

Use the cursor controller to draw a polygon around the objects to be grouped. Press the yellow button to change the direction of the line. Close the polygon by pressing the blue button when the cursor is near the starting point. You can press the blue button to include other objects in the same group. You can also draw more polygons to include other objects in the same group.

The **group** command creates a group of objects on which you can perform many GED operations. The group is defined as a collection of vertices, so any object with a vertex within the group is affected by an operation on the group.

Objects are added to the same group until you use the **group** or **select** command to change the current group or enter a semicolon to end the command.

To define a group:

1 Issue the **group** command. If desired, specify the name of the group.

2 Draw a polygon around the required objects.

The screen displays the group name and the number of bodies, arcs, properties, notes, dots, and wires in the group.

Many commands allow you to operate on entire groups. See the individual command syntax definitions for information on which commands operate on groups.

See Also

exclude	Removes objects or groups from a group.
find	Allows you to define a group of objects by matching text strings.
include	Adds objects or groups to a group.
select	Provides a stretchable rectangle to specify the boundaries of a group.

hardcopy

Sends drawings to a plotter to produce a printed output.

SYNTAX

hardcopy [*scale_option*] [<*dir*>]*drawing_name* [.[*type*][.[*version*][.[*page*]]]]

scale_option	Specifies the size of the printed output. There are two types of options:
a,b,c,d,e	Specifies a page size to scale the drawing.
scale_factor	Specifies a number to scale the drawing from the normal size.
<*dir*>	Plot the specified drawing from the specified SCALD directory.
drawing_name	The name of the drawing to print. If a drawing name is given, a scale factor (number or page size) *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.
.type.version.page	The drawing type, version number, and page of the specified drawing.

Plots can be made on dot matrix, electrostatic, or pen plotters, including Epson, Versatec, Hewlett Packard, CalComp, and Benson models. The **set** or **set plotter** command specifies the type of plotter to use.

EXAMPLES

```
hardcopy 30
```

Plots the current drawing at the default scale of 1.

```
hardcopy a
```

Scales the current drawing onto an A-size page.

```
hardcopy c *.logic*
```

Scales all LOGIC drawings in the current directory onto C-size pages.

```
hardcopy 1 <100k>100112.body*
```

Plots all versions of the 100112 part from the 100k library.

```
hardcopy 1 hyper mux
```

Plots all drawing types for the drawing hyper mux *(BODY, LOGIC, and SIM) in the current directory.*

See Also

set Allows you to specify the type of plotter to be used with the **hardcopy** command and other plotting options.

Section 5 *(Producing a Hardcopy)* of the *ValidGED User's Guide* contains information about plotting drawings with your SCALDsystem.

help

Displays the on-line documentation for a specified GED command.

SYNTAX

help *command_name*

command_name

The name of the command help file to display. The help file briefly describes the syntax and the semantics of the selected command.

To display a list of topics on which help is available, use the command syntax:

help help

or

help ;

If necessary, you can use the **zoom** or **window** command to enlarge the size of the text or pan the help display. To exit from **help**, select another command (other than **zoom** or **window**) from the menu or enter a semicolon.

ignore

Causes a specified directory or library to be deleted from the active search list.

SYNTAX

```
ignore directory_name
ignore library_name
```

directory_name	The directory name to be deleted from the search list.
library_name	The library name to be deleted from the search list.

When you issue the **ignore** command, the system prompts you to enter a semicolon to proceed with the operation. When you ignore a directory or library, 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 as a place holder 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 or library with an equivalent part.

EXAMPLES

```
ignore [Return]
```

Ignores the current SCALD directory.

```
ignore lsttl
```

Ignores the lsttl *library.*

```
ignore practice.wrk
```

Removes the SCALD directory practice.wrk *from the active search list.*

See Also

library	Adds a library to the active search list.
use	Specifies the working directory on the active search list.

Section 2 *(The Editing Environment)* of the *ValidGED User's Guide* explains the file and directory structures in GED and includes a discussion of the active search list.

include

Adds items or groups to a specified group.

SYNTAX

include *pt* [*option*] ...
include *group_name* [*option*] ...

pt

To add individual objects to the current group, press the yellow or blue button. To add previously-defined groups to the current group, press the white button or enter the single-letter group name.

group_name

The name of the current group. Any objects you specify are included in the current group. Specify the current group by entering the single-letter *group_name*. If you do not specify the group, the most recently generated group is used. The *group_name* must be included on the same line as the command.

option

Several optional flags allow you to include *types* of objects in a group. The objects are included based on their association with objects already in the group.

bodies

Include all bodies that have properties already in the chosen group.

connections

Include all objects connected to the pins of any body already in the chosen group.

group_name

Press the white button or enter the single-letter group name to include a previously-defined group in the current group.

nets	Include all nets of wires attached to bodies or wires already in the chosen group.
properties	Include all properties attached to objects already in the chosen group.
wires	Include all wires that have properties already in the chosen group.

See Also

exclude	Removes objects or groups from a group.
find	Allows you to define a group of objects.
group	Allows you to define a group by drawing a closed polygon around the required objects.
select	Provides a stretchable rectangle to specify the boundaries of a group.

led

Compares signal names between logic design and layout drawings. (SCALDsystem only)

SYNTAX

le̲d

This command allows SCALDstar users to compare signal names on the logic design to the layout drawing. The **led** command is identical to **connect led.**

To use the command:

1. From the GED window, type:

   ```
   led (Return)
   ```

2. Go to the LED window and type:

   ```
   ged (Return)
   ```

   The two programs are connected.

   This operation must be completed within approximately 30 seconds.

When the **show net** command is used in either the GED or LED window, the net is found and highlighted in both places.

See Also

connect	Provides interprocess communications between GED and other processes.

library

Adds a specified library to a search list.

SYNTAX

library [*library_name*]

library_name

Adds the specified library to the active search list. This allows you to reference parts from the library and add them to your design.

If you enter the command without a library name, the command returns a list of the available library names.

EXAMPLES

```
library sttl
```

Adds the sttl *library of parts to your active search list of directories.*

```
library Return
```

Lists all possible libraries that you can reference with the **library** *command.*

loadmenu

Read in menu definition files. (Sun systems only)

SYNTAX

lo̲admenu [left | right] *menu_name*

left

Load the required menu on the left-hand side of the GED screen.

right

Load the required menu on the right-hand side of the GED screen. The right-side menu is the default.

If neither **left** or **right** is included with a new menu name, the new menu is loaded on the same side as the current menu position.

menu_name

The name of the file containing the desired menu. You must include the *.menu* filename extension.

The **loadmenu** command reads in predefined or custom menu definition files. The command can be typed in the GED message window, or it may be included in your *startup.ged* file so that a specific menu is loaded when you enter the editor.

loadmenu left or **loadmenu right** without the name of a new menu to load moves the current on-screen menu to the indicated side of the screen.

EXAMPLES

```
loadmenu left ged.menu

loadmenu right /usr/catie/mymenu

loadmenu /usr/valid/tools/editor/menus/pr.menu
```

See Also

menu

Selects the GED command to be displayed on the menu (non-Sun systems).

masterlibrary

Allows you to use a file of abbreviations for your SCALD directories.

SYNTAX

masterlibrary *filename*

filename — The name of the file containing abbreviations for the SCALD directories you want to access.

The **masterlibrary** command allows you to use abbreviations to refer to your SCALD directories. This allows you to specify SCALD directories outside your current working directory without entering the entire pathname or file specification from GED.

In your *startup.ged* file, enter the **masterlibrary** command followed by the name of the file containing the abbreviations.

The default file *master.local* in your login directory is provided for this use. You can, however, create an abbreviation table file with any name. In this file, specify enough information in the path or file specification so that GED can find the appropriate SCALD directory when you enter the **use** command.

For example, if you often require a SCALD directory located on another machine on the network, you should specify the machine-rooted path. If the SCALD directory is located in another one of your project directories, specify the relative path to the file.

See Also

ignore	Removes the specified SCALD directory from the search stack.
library	Adds a library to the search stack.
use	Adds a specified SCALD directory to the search stack.
write	Writes the drawing to the current or specified SCALD directory.

menu

Selects the GED command to be displayed on the menu. (Non-Sun systems only)

SYNTAX

menu *number ged_command_word*
menu *pt ged_command_word*

number	A number from 1 to 15 to specify the menu box position.
pt	Instead of specifying a menu box position, you can point to the required box and press any button.
ged_command_word	The GED command to display in the specified menu box position.

The **menu** command allows you to select the GED commands displayed in the menu. **menu** commands can be placed in *startup.ged* and script files. To change the displayed menu:

1 Issue the **menu** command.

2 Specify the menu box position with a number or by pointing to the required box and pressing any button.

3 Type the GED command (abbreviations are allowed) and press Return.

See Also

loadmenu	Reads in predefined or custom menu definition files (Sun systems only).

mirror

Creates a mirrored version of a selected body.

SYNTAX

mirror *pt ...*

The **mirror** command creates a mirrored version of a body, as opposed to a rotated version. This command mirrors all lines and arcs in a body drawing about the Y axis. 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 select the body to be mirrored with the yellow cursor button.

In a body drawing, **mirror** Return flips over the entire body definition. This procedure is useful for creating other versions of a body.

For instance, two versions might resemble those in Figure 3.

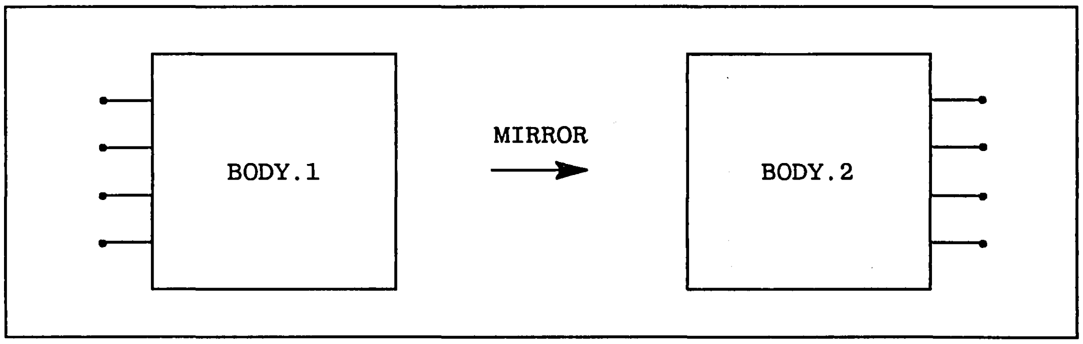


Figure 3. Mirrored Body Versions

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 *(Creating a Design)* of the *ValidGED User's Guide* contains additional information about adding bodies to a drawing, and Section 4 *(Design Techniques)* contains additional information about creating body drawings.

move

Moves objects from one position to another.

SYNTAX

move *source_pt destination_pt ...*
move *group_name destination_pt ...*

source_pt	The object to move.
group_name	The group to move.
destination_pt	The new position of the group or object.

The **move** command is used to move objects from one position to another on the drawing.

Properties (including signal names) attached to objects are moved with the objects. Properties can also 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.

To use the **move** command to move single objects:

1. Select **move** from the menu.
2. Position the cursor on the object that is to be moved and press the appropriate button.
   - The yellow button picks up the object at the grid point nearest the cursor.
   - The blue button picks up the object at the vertex nearest 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 yellow button places the object on the grid point nearest the cursor.
- The blue button attaches the object to the nearest vertex.

The **move** command also operates on defined groups. You can specify the name of the group or press the green or white button to select the group to be moved.

The **window** and **zoom** commands can be nested within the **move** command.

See Also

group	Defines a group of objects (which can then be moved).
select	Defines a group of objects (which can then be moved).
split	Separates objects with common vertices so they can be moved.
set	**set move_direct** and **set move_orthog** change the wire mode used by the **move** command.

next

Displays the items located by the **find** command.

SYNTAX

<u>ne</u>xt

The **next** command steps through the items found by the **find** command. It draws an asterisk at the location of the item's 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 the list once.

If you run the **check** command, **next** performs like the **error** command by finding the next error in the design.

See Also

find	Defines a group of items that match a specified pattern.

note

Adds text strings to a drawing.

SYNTAX

note (*text_line* ... *pt* ...) ... ;

text_line The text to add to the drawing.

pt Where each note is to appear on the drawing.

Notes are text strings that appear on the drawing; they do not affect the evaluation of the drawing by the SCALDsystem.

There are two ways to add notes to a drawing:

- Specify the points on the drawing where the notes are to be located and then type in the text. Press Return after each note to position each note on the drawing. As long as there are points remaining, GED interprets the text you enter as notes to the drawing.
- Type in each line of text and press Return. (You can enter several strings before placing them.) Then use the cursor and the yellow button to indicate where each note is to appear on the drawing.

Place quotation marks around notes beginning with an open parenthesis. Quoted notes are never interpreted as GED commands.

The **window** and **zoom** commands can be nested within the **note** command.

See Also

filenote Allows you to add a text file to a GED drawing.

paint

Assigns selected colors to specified objects.

SYNTAX

paint *color_name group_name*
paint *color_name pt*
paint default

color_name	The color to assign to a group or object.
group_name	The single-letter name of the group to color.
pt	To assign a color to an object, point to the object and press the yellow button. Select a group by entering the group name or by pointing to the required group and pressing the white button.
default	This command paints objects in their preset default colors. Use the **set** command to establish default colors for the objects in your drawings.

The **paint** command allows you to assign colors to the objects in your drawing. Even if you are working on a monochrome monitor, you can add color to drawings that can be printed on a color plotter or transferred to a color workstation.

You can use up to 16 colors in a drawing. The predefined colors are shown in Table 5.

Table 5. Predefined Paint Colors

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

If you have a color monitor, the objects are drawn in the actual colors you specify. If you use a monochrome monitor, you can 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 status line 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.

The **window** and **zoom** commands can be nested within the **paint** command.

You can establish default colors for the objects in your drawings with the **set color** commands.

EXAMPLE

The **find** *command places all* ls00 *parts it finds in the first available group name.*

To change the color of all the LS00 bodies in a design:

```
find ls00
paint skyblue "a"
```

If you have a color monitor, the colors are displayed on the screen.

See Also

set

The **set color** commands allow you to specify a default color to be used for each type of object in the drawing.

show

The **show color** command lists the color of the specified object.

paste

Copies the contents of a **cut** buffer to the current drawing.

SYNTAX

paste *pt ...*

The **paste** command, used with the **cut** command, allows an object or a group of objects to be copied from one drawing to another. To copy a group or object that has been **cut**, type:

```
paste [Return]
```

and then select the point to position the group or object.

To add more copies of the cut buffer, press the yellow cursor button, position the copy, and then press the yellow button again.

The **window** and **zoom** commands can be nested within the **paste** command.

See Also

cut	Copies an object or a group from a drawing to a buffer.

pause

Temporarily interrupts GED until you press a key.

SYNTAX

pause

This command allows you to temporarily interrupt GED until you press another key. This is useful for demos and scripts.

See Also

sleep	Allows you to temporarily stop GED from within a script.

pinswap

Swaps the pin numbers defined to be in the same pin group on a body.

SYNTAX

pinswap (*pt1 pt2*) ...
pinswap *pin_number pt* ...

pt1 pt2 — The two pins to be swapped.

pin_number — Rather than pointing to two pins, you can type in a new pin number and then point to the pin. The selected pin is swapped with the pin having the pin number you specified.

See the description of the PIN_GROUP property in the Library Reference Manual *for information about making pins swappable.*

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, use one of the following procedures:

- Type **pinswap** and point to the two pins to be swapped.
- Type **pinswap**, type in a new pin number, and then point to the 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

backannotate	Annotates the design with physical information from the Packager.
section	Displays different sections of a body with pin numbers.
set	**near_pin**, **far_pin**, **rotate**, and **pin_size** control the appearance of pin numbers on the drawing.

property

Attaches a property name and value to a specified vertex of an object.

SYNTAX

property (*attach_pt* (*name value location_pt*) ...) ...
property (*group* (*name value*) ...) ...

attach_pt	The vertex where the property is to be attached.
group	The name of a group. The group name can be typed in or a group can be selected by pressing the white button. The specified property is attached to each object in the group.
name	The name of the property. The property name must be less than 16 characters long.
value	The value of the property. The name and value can be separated by a space, an equal sign (=), or can be typed on separate lines.
location_pt	The location on the drawing where the text of the property value should appear.

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 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 described in the *SCALD 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 GED 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 (except the SIG_NAME property) must have a unique name. If a newly entered property has the same name as a property currently attached to that object, the new property value replaces the old property value.

To specify a property:

1. Select **property** from the menu.
2. Specify the object (vertex) or group 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 or an equal sign (=), or typed on separate lines.
4. Specify the location on the drawing where the text of the property value should appear. The placement is automatic when properties are assigned to groups.

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. The **set prop_display** command controls the display of added properties.

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 **pinswap**, **section**, and **backannotate** commands cannot be manipulated.

The **window** and **zoom** commands can be nested within the **property** command.

See Also

copy	Copies objects, properties, and groups in the current drawing.
delete	Removes objects from a drawing.
display	Changes the way objects are displayed on a drawing.
move	Moves objects from one position to another.
reattach	Reattaches properties from one object to another.
set prop_display	Changes the default values controlling the display of properties on the drawing.
show attachments	Displays the connections between properties and the objects to which they are attached.
show properties	Displays the name and value of the properties on the drawing.
signame	Attaches signal names to wires or pins.
swap	Swaps the position of two properties.

quit

Allows you to end the editing session.

SYNTAX

```
quit
```

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

Reattaches properties from one object to another.

SYNTAX

reattach *pt1 pt2 ...*

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
   ```

   and select the property to be moved. A line is drawn from the property to the current cursor position.

2. Specify the new attachment point for the property.

3. 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

move	Repositions the selected object.
show attachments	Allows you to verify that the properties you reattached are attached to the correct objects.

redo

Reverses the last **undo** command.

SYNTAX

redo

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 changes made to a drawing during the current design session.

remove

Deletes a drawing from a SCALD directory.

SYNTAX

remove [*<dir>*] [*drawing_name*] [. [*type*] [. [*version*] [. [*page*]]]]

<dir>	The SCALD directory where the drawing resides.
drawing_name	The name of the drawing to remove.
.type.version.page	The drawing type, version number, and page of the specified drawing.

This command deletes a drawing from a SCALD directory. Because you can specify only one argument with the **remove** command, repeat the procedure to delete additional drawings. 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 Return.

GED displays the names of the files to be deleted.

3 Select the semicolon from the GED menu or type:

; Return *(semicolon)*

The directory entries are deleted, and the files are purged.

To cancel the **remove** command, type:

```
abort [Return]
```

or select any command except semicolon from the menu. GED displays the message:

```
Nothing done
```

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 drawing name is specified, **remove** deletes all drawing types (BODY, LOGIC, SIM, etc.), versions, pages, and files (ASCII, binary, dependency, connectivity) of the specified drawing in the SCALD directory.

If no SCALD directory is given, **remove** searches for the specified drawing in the currently active SCALD directory.

EXAMPLES

```
remove drawing1
```

Deletes all drawing types (SIM, LOGIC, BODY) of drawing1.

```
remove drawing2.logic.*
```

Deletes only the LOGIC pages of drawing2.

```
remove drawing3.logic.*.1
```

Deletes only the first page of the LOGIC drawing, drawing3.

replace

Substitutes one part for another.

SYNTAX

re<u>**pl**</u>**ace** *body_name*[**.body**] [.*version*] *pt* ...
re<u>**pl**</u>**ace** *body_name*[**.body**] [.*version*] Return *group_name*

body_name

The name of the body drawing to be replaced.

.body

The type BODY can be specified, but is not required.

version

The version defaults to 1, but any existing version of a body can be replaced.

group_name

The name of the group to be replaced. This is useful for global changes if you have placed all of one type of part into a group.

The **replace** command is used to substitute one part for another. There are several ways to use the **replace** command:

- Enter the name of the replacement part, then use the cursor to point to the body or bodies to be replaced.
- 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.
- Use the **find** command to group all the occurrences of a body to be replaced, then use the

group_name option with the **replace** command to globally change all the occurrences of the body. A message displays the number of bodies that are replaced. When you replace the bodies in a group, enter a Return before you type the group name.

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 location. The rotation of the original body is preserved when the body is replaced.

See Also

add	Adds a specified drawing to the body.
find	Defines a group of items that match a specified pattern.
version	Selects an alternate version of a body.

return

Returns to the previously-edited drawing.

SYNTAX

return

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.
show return	Lists the drawings that the **return** command will return to in the order that they will be accessed.

rotate

Rotates a body or text string 90 degrees, with mirrors at 180 and 270 degrees.

SYNTAX

```
rotate pt ...
```

The **rotate** command creates rotated and mirrored versions of a selected body. When a body is rotated, all properties are also rotated. Text strings can be also rotated or justified independently.

To rotate a body or text string, type **rotate** and then point to the object to be rotated. Each time you press the button, the part rotates 90 degrees. In the 90 degree rotation, body notes are rotated 90 degrees and left in their original justification.

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).

For the mirrors, justified text is shifted from left to right or right to left, and no further rotation is done. Text rotations (properties and drawing notes) are actually rotated, not mirrored.

See Also

add	Bodies are rotated during the **add** command when you press the white button.
mirror	Creates a mirrored version of the selected body.
spin	Provides true rotations, not mirrors, of a body.

route

Draws a wire connecting two selected points.

SYNTAX

<u>rou</u>**te** *pt pt ...*

The **route** command connect two points by drawing a series of orthogonal line segments between them. If it cannot determine a route, it draws a diagonal line directly between the two points. **route** will not run a wire through any existing objects or vertices.

To select the nearest pin or wire vertex for a **route** point, use the blue button to select the point. Use any other button to select the nearest grid point.

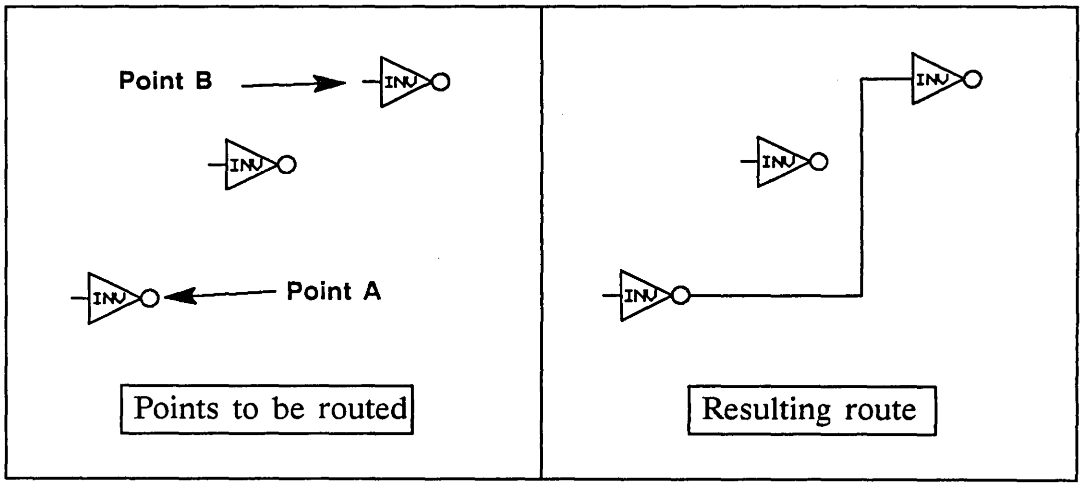


Figure 4. Route Command Operation

See Also

wire	Adds individual wires to a drawing.

scale

Smashes a drawing and includes it in the current drawing.

SYNTAX

<u>sca</u>le (*pt1 pt2*) *drawing_name*

pt1 pt2	Indicate the size of the rectangle where the smashed drawing will be placed.
drawing_name	The name of the drawing to smash and add to the current drawing.

The **scale** command adds a specified drawing to the current drawing in the rectangle indicated by two points. The drawing is smashed (all bodies are 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.
smash	Breaks a body into the objects that define it.

Section 7 *(Mixing Text and Graphics)* of the *ValidGED User's Guide* contains more information about creating mixed text and graphics documents.

script

Performs the GED commands listed in the specified text file.

SYNTAX

```
script filename
```

filename — The name of the script file to execute.

The **script** command allows you to specify GED commands in a script file. This allows you to operate in batch mode using the same syntax as if you typed in the command. You can use the cursor controller to enter points or you can specify the X–Y coordinates in the script file.

startup.ged is a good example of a script file. 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.

You can configure a script to accept input during execution by including *user input tokens* in a script. User input tokens must be placed at the beginning of a new line. There are two user input tokens:

$< — When GED encounters this token in a script, it prints from the token to the end of the text line as a prompt in the message window, then waits for one item of input. The input can be a typed line, a function key press, a cursor controller point, or a Control-C; you cannot use a Return as a response to a user input request.

$; — This token also prints from the token to the end of the text line as a prompt and awaits input, but this token accepts and interprets input until you enter a semicolon. If this token is included, GED follows the prompt with the message:

```
Type ; when done with user input.
```

When GED sees a user input token in a script, it highlights a menu button with the name of the GED command being executed.

To abort a script, enter Control-C. To abort at a user input token prompt, enter a semicolon.

EXAMPLES

```
add ls04
$<Place the LS04
```

Add a single LS04 to a drawing and use the mouse to position the part.

```
property
$<Choose the part to add a size to
size =
$<Type in the size you want
$<Place the property on the drawing
```

Add a size property to a part with a size specified at the time of entry.

```
rotate
$;Rotate the object until properly oriented
```

Rotate an object until the user enters a semicolon.

A more complicated script might contain a large number of **signame** commands and prompt the user for a point to place each SIG_NAME property.

section

Displays different pin numbers for different sections of a body.

SYNTAX

section [*pin_number*] *pt* ...

pin_number	The pin number that defines the section.

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 logical part selected can be assigned to a section, the pin numbers for the 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 avoids stepping through each section individually.

Currently, you can only section parts with SIZE = 1 or HAS_FIXED_SIZE characteristics. Assigning sections to a HAS_FIXED_SIZE part is accomplished by pointing to the pin of the section to be assigned.

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.

To remove section information from a part, use the **replace** command to replace the sectioned body with a new copy of the part.

See Also

backannotate	Annotates the design with physical information from the Packager.
pinswap	Swaps pin numbers on a body that are defined to be in the same pin group.
set	Several options control the placement of pin numbers on the drawing.

Section 6 *(Adding Physical Information)* of the *ValidGED User's Guide* contains more information about the *chips_ prt* file.

select

Provides a stretchable rectangle to specify the boundaries of a group.

SYNTAX

<u>sel</u>ect [*group_name*] [**<u>a</u>ll**] *pt1 pt2 ...*

group_name	A single-letter name for the group. Do **not** enclose the group name in quotation marks. The group name is optional.
all	Places the entire drawing into the group.
pt1 pt2	Defines the boundaries of a group. Use the cursor and the yellow button to position a stretchable rectangle around the objects to be included in the group. You can also press the blue button to include individual objects in the same group.

You continue to add objects to the same group until you enter another command or a semicolon.

The **window** and **zoom** commands can be nested within the **select** command.

See Also

exclude	Removes items from a group.
find	Defines a group of items that match a specified pattern.
group	Allows you to define a group of objects by defining a closed polygon.
include	Adds objects or groups to a group.

set

Establishes the default options for GED.

SYNTAX

set *option*

The **set** command modifies the default command options used by GED. The commands can be issued during an editing session or placed in the *startup.ged* file.

option	**set** has many options that allow you to tailor GED to your particular requirements:
set Return	Lists the current settings and options.
ascii **binary** **conn** **dependency** **noascii** **nobinary** **noconn** **nodependency**	Specifies the types of files that are written when a drawing is saved. Default files are: • ASCII • Connectivity • Binary • Dependency Unwanted files can be turned off temporarily and then reset.
capslock_off **capslock_on**	**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 uppercase regardless of how it is entered at the keyboard. Only affects text added to a drawing.
center_ justified **left_ justified** **right_ justified**	Sets the justification of text strings (properties and notes). The default is left-justified.

check_on_write off
check_on_write on

Determines whether the **check** command is automatically called by the **write** command. The default is **check_on_write on**.

color_arc *color*
color_body *color*
color_dot *color*
color_note *color*
color_prop *color*
color_wire *color*

Sets the default color for new objects of the specified type. The default for all objects is monochromatic. The *color* can be one of the following:

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

decimal
fractional
metric

decimal bases drawings on the decimal system, with 500 internal units per physical inch. The grid spacing on the status line of the display is in grids–per–inch. **fractional** sets the default internal division 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. **metric** bases drawings on the metric system with 508 internal units per inch, or 20 internal units per millimeter. The grid spacing on the status line of the display is in grids–per–millimeter. This remains compatible with Valid libraries since pins are on 2.5 mm centers.

default_doc_grid *options*
default_grid *options*

The **default_grid** command changes the default grid. **default_doc_grid** changes the default grid for DOC drawings. The available *options* for both commands are the same as those for the **grid** command:

- **on**
- **off**
- *grid_size*
- *grid_multiple*
- **dots**
- **lines**

See the **grid** *command for more information on default grid options.*

The default is grid size is 0.1. The grid for BODY drawings is 1/2 the default grid setting (initially 0.05). The initial grid for DOC drawings is 0.166.

direct_wire
orthog_wire

orthog_wire sets the wiring mode to orthogonal (right-angle bent) wires. Orthogonal mode is the default. The **direct_wire** option sets the wiring mode to non-orthogonal (diagonal) wires.

dots_filled
dots_open

dots_open displays added dots as small open circles (default). **dots_filled** displays filled (solid) dots.

double_width
single_width

Governs the darkness of plotted lines. The default is **double_width**, which prints two pixels instead of one.

far_pn
near_pn

Controls the placement of pin numbers on the drawing. **far_pn** (default) places numbers slightly further from the pin than **near_pn**.

font *font_name*	Allows you to specify the text font to be used in the hard copy of the drawing. The following fonts are available: • **vector_font** (default) • **native_font** • **gothic_font** • **milspec_font** • **cursive_font** • **valid_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.
go_at_pin **stop_at_pin**	**go_at_pin** continues the wire from a pin. You must press the yellow cursor button twice to end the wire. **stop_at_pin** ends a wire when the blue button is used or when the yellow button is used at a pin, dot, wire endpoint, or T-junction (default).
grid_off **grid_on**	Controls whether or not the grid is displayed when GED is entered and when new drawings are edited. The default is **off**.
hpf **vgb**	Plots using *hpfilter*. **vgb** plots using the monochrome S32 display.

local_plot **spooled_plot**	Determines whether plot 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.
mono_hpplot	Uses hpfilter to plot to a Hewlett-Packard plotter.
move_direct **move_orthog**	**move_orthog** uses orthogonal (bended) wires when an object is moved (default). **move_direct** uses diagonal wires when an object is moved.
nfs_file_locking off **nfs_file_locking on**	Allows the System Manager to determine whether GED honors network locks. **nfs_file_locking on** enables network-style locks. **nfs_file_locking off** tells GED to use flock locks only. The default is **off**.
pin_size *scale*	Changes the size of added pin numbers. The scale can be set to any real number. The default is 0.80.
plotter *plotter_name*	Allows you to specify the name of a plotter supported by *hpfilter*. The default is an 11-inch Versatec. You can also plot to a file that conforms to the graphics standards for the Interleaf publishing software. For example: • **w11versatec** (default) • **interleaftps** For a full listing of the plotters supported by your system, see the appropriate system administration manual.

prop_display *display*	Specifies the default visibility of properties you add to a drawing. The *display* options are: • **both** • **name** • **invisible** • **value** (default)
push_type *drawing_type*	Specifies the drawing type for **edit** when the drawing extension or type is missing. The default is LOGIC. This is also used by the **edit** and **get** commands when you specify a drawing to edit by selecting a body with the cursor.
rotate off **rotate on**	Determines whether numbers annotated to vertical pins are rotated. The default is **rotate on**.
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**.
sticky_off **sticky_on**	If a default property is deleted from a BODY drawing, **sticky_off** 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.
user_editor *editor_name*	Specifies the editor to be used in the **v** mode of the **change** command. The default editor is *vi* on UNIX systems and EDIT/EDT on VMS systems.
user_sim *simfile*	Allows you to specify the UNIX pathname or VMS file specification of an alternate Simulator executable.

show

Temporarily displays the specified drawing information.

SYNTAX

show *option*

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.

option	**show** has several options:
show Return	Displays a list of all the **show** options.
attachments	Displays the connections between properties and the objects to which they are attached.
body_name *pt*	Displays the name, version, angle, and SCALD directory of the indicated part.
color *pt*	Lists the color of the specified object.
coordinate *pt ...*	Displays the internal GED coordinates of an indicated point.
connections	Displays an asterisk at each wire connection in the drawing.
distance *pt1 pt2*	Displays the distance between two indicated points in the drawing area. If **set decimal** or **set fractional** mode is set, the unit of measure is inches; if

	metric mode is set, the unit of measure is millimeters. Use the yellow button to specify an actual location on the screen, the white button to specify the nearest grid point, or the blue button to specify the nearest vertex.
group [*pt*] **group** [*group_name*]	Causes the specified group to be highlighted. You can either select the nearest group with the cursor or type the name of 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 during this GED session and shows which are modified but unwritten. **show history** also lists the drawing you will edit with the **return** command.
keys	Lists the function keys and the corresponding text string that has been assigned to each key.
modified	Lists all the drawings you have modified but not written during this GED session.
net [*pt*] **net** [*net_name*]	Highlights all nets matching the name of the selected or specified net. The net can be specified by name or by pointing to a net with the cursor.
origins	Displays asterisks at 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 wires, the use of **show properties** displays the text `SIG_NAME=` with each signal name.
pwd	Lists the UNIX or VMS directory from which the current GED session originated.
release	Displays the release version information for GED.
return	Displays, in order, the previously-edited drawings you can visit with the **return** command.
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.
vertices	Displays asterisks at the vertices of all objects.

signame

Attaches signal names to wires or pins.

SYNTAX

signame (*signal_name ... pt ...*) ...

signal_name

The text for the signal name.

The **signame** command allows you to attach signal names to wires or pins. To attach a signal name:

1. Select **signame** from the menu.
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.
4. Press Return.

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. When **signame** is used to name pins in BODY drawings, they are stored as properties with the name `PIN_NAME`.

The **window** and **zoom** commands can be nested within the **signame** command.

See Also

busname	Provides a shorthand for naming signals in buses or pins on bodies.
property	Attaches a property name and value to an object.

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains more information about properties and signal names. The *SCALD Language Manual* contains details about signal name syntax.

simulate

Allows you to run the Simulator program.

SYNTAX

<u>sim</u>**ulate** [*root_drawing*]

root_drawing

When you run the **simulate** command, the default drawing (the one listed in your *simulate.cmd* file) is simulated unless you provide the name of another drawing to simulate.

The Simulator is optional software and may not be included with your system.

The **simulate** command is used to simulate the functional behavior of a drawing. The **simulate** command creates a Simulator window in the lower portion of the screen and establishes communication with the GED window. You can use the Simulator OPEN command and then select signal names from the GED window.

To exit from the simulator, type EXIT in the simulator window or **endsim** in the GED window.

See Also

The *ValidSIM Reference Manual* contains a thorough discussion of the Simulator program.

sleep

Stops GED for a specified number of seconds.

SYNTAX

sleep *seconds*

seconds — The number of seconds to pause.

The **sleep** command pauses GED for the specified number of seconds. This command is useful in scripts and demos.

See Also

pause — Stops GED until a key is pressed.

smash

Breaks a body into the objects that define it.

SYNTAX

<u>sm</u>**ash** *group_name ...*
<u>sm</u>**ash** *pt ...*

group_name

You can either select the nearest group with the cursor or type the name of the group.

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 use this command on bodies and groups of bodies in the drawing.

EXAMPLE

You can create a 3-input AND gate from a 2-input AND gate with the following commands:

1. `edit user 3and.body`
2. `add 2and` *pt*
3. `smash` *pt*
4. Attach the additional input pin and add pin_names and dots.
5. Write the drawing.

Normally, you cannot add a body to a body drawing. Using the **smash** command changes the 2AND body into its separate elements so that GED does not interpret it as a body when the USER 3AND drawing is written.

See Also

add	Adds a body to a drawing.
scale	Smashes a drawing and adds it to the current drawing.

spin

Provides true rotations, not mirrors, of a body.

SYNTAX

<u>sp</u>in *pt ...*

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 some bodies, which can cause errors in the drawing. **spin** also rotates text strings.

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

Adds a segment to an existing wire and separates objects with common vertices.

SYNTAX

split *pt1 pt2 ...*

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.

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. Select **split** from the menu and select a point along the wire with the yellow button.

   This adds a vertex along the wire segment 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 placed at the same vertex:

1 Select **split** from the menu and select the desired vertex with the cursor (press the blue button).

  The selection attaches one of the objects to the cursor so it can be moved 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 and begin the cycle again, splitting off each item in turn.

swap

Exchanges the position of two lines of text (properties or notes).

SYNTAX

swap *pt1 pt2 ...*

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**, **backannotate**, and **section** commands cannot be swapped.

See Also

note	Adds notes to a drawing.
pinswap	Exchanges the pin numbers assigned to a body by **section**.
property	Attaches properties to objects in a drawing.

system

Allows you to access the operating system.

SYNTAX

system [*command*]

command

A system command to execute from within GED.

The **system** command allows you to access the operating system on your system. If you just enter **system** Return, you are connected to the operating system and can run any operating system commands.

To exit from the operating system and return to GED, type:

UNIX: Control**-D** or **exit**

VMS: **logout**

This command is identical to the **unix** command.

undo

Undoes the operation of the previous command affecting the drawing.

SYNTAX

```
undo
```

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** reverse 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

Allows you to access the operating system.

SYNTAX

unix [*command*]

command	A system command to execute from within GED. After the command is executed, the system displays a prompt instructing you to press Return (unless the command is executed from a script).

The **unix** command allows you to access the operating system on your system. If you enter

```
unix Return
```

you are connected to the operating system and can run any operating system commands.

To exit from the UNIX shell and return to GED, type:

UNIX:	**exit**
VMS:	**logout**

This command is identical to the **system** command.

See Also

system	Accesses either the VMS or UNIX operating system from GED.

use

Specifies a working directory.

SYNTAX

use *directory_name*

directory_name

The name of the SCALD directory to use. If the SCALD directory you specify is in a directory other than the current directory, the UNIX pathname or VMS file specification must be given.

The **use** command allows you to specify a SCALD directory from which you can retrieve drawings and in which you can 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.

The **masterlibrary** command allows you to refer to SCALD directories by abbreviations. You place the pathnames or file specifications and abbreviations for SCALD directories in an abbreviation table file. Place the **masterlibrary** command and the name of the abbreviation file in your *startup.ged* file. Then when you **use** the file during a GED session, you can specify just the short file name.

EXAMPLES

```
use /u0/job/part1.wrk
```

Specifies the SCALD directory part1.wrk *in the UNIX directory* /u0/job.

```
use project1.wrk
```

Specifies the SCALD directory project1.wrk *in the current working directory.*

```
use connectors
```

Accesses the SCALD directory associated with the abbreviation connectors.

See Also

directory	List the contents of the active SCALD directories.
ignore	Deletes the specified directory or library from the active list.
library	Specifies the component library to be accessed.
masterlibrary	Specifies the name of an abbreviation file with the pathnames or file specifications and abbreviations of SCALD directories.

Section 2 *(The Editing Environment)* of the *ValidGED User's Guide* explains the file and directory structures of GED.

vectorize

Creates a file in vector plot format of the current drawing.

SYNTAX

vectorize

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 of the *ValidGED User's Guide* contains more information about vector plot format.

version

Selects an alternate version of a body.

SYNTAX

version *group_name ...*
version *pt ...*

group_name

You can use the **find** command to group all occurrences of a specified body, then issue the **version** command with the *group_name* option to globally change the drawing. The white button changes the version of the bodies in the group closest to the cursor.

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.

Note that size-wide versions of bodies are represented. The first version is *sizeable*; you can specify the number of bits the part can represent. This version is generally used in structured designs. The second version is a flat representation of the part; each pin on the drawing represents a pin on the physical package. This version is generally used in flat designs.

See Also

add	Allows you to add a specific version of a body directly to a drawing.
replace	Substitutes one device for another.

window

Changes the view of the current drawing.

SYNTAX

window [*option*] ;

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 from the on-screen menu with the cursor or type a semicolon followed by a Return.

option	The **window** command options are shown below:
window ;	If you issue the **window** command followed by a semicolon, GED redraws the image without changing the center or the scale. This option refreshes the screen when error messages cover part of the drawing.
window down	Reposition the center of the screen down below the drawing (move the drawing *up* on the screen).
window fit	Fits the drawing to the entire screen.
window in	Enlarge the size of the drawing on the screen.
window left	Reposition the center of the screen to the left of the drawing (move the drawing *right* on the screen).
window out	Reduce the size of the drawing on the screen.
window previous	Switch from the current window scale and position to the previous window scale and position.

window *pt*	The **window** command with an argument of one point pans the drawing and causes that point to become the center of a new screen display of the drawing. The scaling of the drawing remains the same. Use the blue button to enter the single point.
window *pt1 pt2*	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.
window *pt1 pt2 pt3*	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 *pt1* and *pt3* is greater than the distance between *pt1* and *pt2*, the items appear larger; if the distance is smaller, items appear smaller.
window right	Reposition the center of the screen to the right of the drawing (move the drawing *left* on the screen).
window *scale_factor*	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.
window up	Reposition the center of the screen up above the drawing (move the drawing *down* on the screen).
	Some of the **window** commands may be pre-assigned to function keys, depending on your system.

EXAMPLES

```
window 2
```

Makes the drawing appear twice as large.

```
window -2
```

Reduces the drawing by a factor of two.

```
window 1.5
```

Enlarges the drawing one and a half times.

```
window 0.5
```

Has the same effect as **window -2.**

See Also

zoom Also allows you to enlarge and reduce portions of the drawing.

Section 1 *(GED Overview)* of the *ValidGED User's Guide* contains more information about function key assignments, and Section 3 *(Creating a Design)* contains more information about window and display functions.

wire

Adds wires to a drawing.

SYNTAX

<u>wir</u>e *pt pt ...*

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.

To snap the wire to the nearest vertex, press the blue button. To end a wire at a pin, dot, or other wire, press the yellow button. To end a wire in a free space, press the yellow button twice at the final point.

The **set** commands **set stop_at_pin** and **set go_at_pin** allow you to specify the default method for ending wire 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. To bend a wire, press the yellow button. Press the white or green button to change the orientation of the bend. If the white button is pressed a second time, the wire becomes diagonal. A third press returns the wire to the first orthogonal position.

The **set direct_wire** command can be typed at the keyboard or added to your *startup.ged* file. In this mode, finishing a wire segment with the yellow or blue button creates a diagonal wire. Ending a wire with the white or green button creates orthogonal wire segments to the nearest grid point. You can return to the automatic orthogonal wiring mode by entering the **set orthog_mode** command.

To indicate wire connections, you can use the **dots** or **auto dots** 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.

The **window** and **zoom** commands can be nested within the **wire** command.

See Also

display	Allows the display of wires to include buses and patterned lines.
route	Automatically draws a wire between two selected points.
set	Allows you to set default wire options.
show connections	Temporarily highlights all wire connections in your design.

Section 3 *(Creating a Design)* of the *ValidGED User's Guide* contains more information about adding wires to a drawing.

write

Writes the current drawing onto the disk.

SYNTAX

write [[*<dir>*] [*drawing_name*] [. [*type*] [. [*version*] [. [*page*]]]]]

<dir>

The SCALD directory where the drawing resides. 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.

drawing_name

The name of the drawing to write. If no drawing name is specified, the drawing is given the drawing name specified on the status line at the top of the display. If you enter a drawing name and 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.

.type.version.page

The drawing type, version number, and page of the specified drawing.

EXAMPLES

```
write
```

Saves the current drawing named on the status line in the current SCALD directory.

```
write newname
```

Stores the current drawing named on the status line in the current SCALD 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 the* **write**.

```
write <project2.wrk>
```

Writes the current drawing into the SCALD directory project2.wrk.

See Also

diagram	Allows you to rename a drawing.
exit	Leaves the editor.
library	Specifies the component library to be accessed.
quit	Leaves the editor.
set	The **check_on_write** option automatically calls the check command when the **write** command is issued.
use	Specifies a working directory.

zoom

Reduces and enlarges portions of the drawing.

SYNTAX

zoom [*option*]

The **zoom** 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 from the on-screen menu with the cursor or type a semicolon followed by a Return.

option	The **zoom** command options are shown below:
zoom ;	If you issue the **zoom** command followed by a semicolon, GED redraws the image without changing the center or the scale. This option refreshes the screen when error messages cover part of the drawing.
zoom down	Reposition the center of the screen down below the drawing (move the drawing *up* on the screen).
zoom fit	Fits the drawing to the entire screen.
zoom in	Enlarge the size of the drawing on the screen.
zoom left	Reposition the center of the screen to the left of the drawing (move the drawing *right* on the screen).
zoom out	Reduce the size of the drawing on the screen.
zoom previous	Switch from the current zoom scale and position to the previous zoom scale and position.

zoom *pt*	The **zoom** command with an argument of one point pans the drawing and causes that point to become the center of a new screen display of the drawing. The scaling of the drawing remains the same. Use the blue button to enter the single point.
zoom *pt1 pt2*	The **zoom** 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.
zoom *pt1 pt2 pt3*	You can issue the **zoom** 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 *pt1* and *pt3* is greater than the distance between *pt1* and *pt2*, the items appear larger; if the distance is smaller, items appear smaller.
zoom right	Reposition the center of the screen to the right of the drawing (move the drawing *left* on the screen).
zoom *scale_factor*	You can specify an integer or a real number as the argument to the **zoom** command to scale the view of the drawing by the amount entered. The center of the window remains the same.
zoom up	Reposition the center of the screen up above the drawing (move the drawing *down* on the screen).

Some of the **zoom** commands may be pre-assigned to function keys, depending on your system.

EXAMPLES

```
zoom 2
```

Makes the drawing appear twice as large.

```
zoom -2
```

Reduces the drawing by a factor of two.

```
zoom 1.5
```

Enlarges the drawing one and a half times.

```
zoom 0.5
```

Has the same effect as **zoom -2**.

See Also

window	Changes the view of the current drawing.

Index