

# Introduction to GEM™ Programming



	page
Introduction . . . . .	1
The GEM Programmer's Toolkit . . . . .	1
Documentation Description . . . . .	1
Software Description . . . . .	2
Technical Support Services . . . . .	3
The Development Environment . . . . .	4
Hardware Requirements . . . . .	4
Software Requirements . . . . .	4
An Overview of GEM System Software . . . . .	5
GEM Virtual Device Interface (VDI) . . . . .	5
NDC vs. RC Coordinate System . . . . .	5
Raster Text Fonts . . . . .	6
The GEM Metafile . . . . .	6
GEM Application Environment Services (AES) . . . . .	7
Memory Management . . . . .	7
Constructing A GEM Application . . . . .	7
Using IconEdit . . . . .	8
The GEM IconEdit Screen . . . . .	8
GEM IconEdit Concepts . . . . .	10
GEM IconEdit Menus . . . . .	12
Desk Menu . . . . .	12
File Menu . . . . .	13
Operation Menu . . . . .	14
Using the Resource Construction Set . . . . .	16
The GEM RCS Screen . . . . .	16
GEM RCS Menus . . . . .	18
Desk Menu . . . . .	18
File Menu . . . . .	19
Options Menu . . . . .	20
Global Menu . . . . .	21
Partbox Window Contents . . . . .	22
Tree Operations . . . . .	23
Objects . . . . .	25
Object Operations . . . . .	28
Tips And Concepts . . . . .	33
Visual Hierarchy: Relationship between Trees and Objects . . . . .	33
Notes on Making Icons and Bit Images . . . . .	34
Memory Limits . . . . .	34
Name Conflicts with Merge . . . . .	34
Getting Started, a Sample Session with the RCS . . . . .	35
Using GEM SID . . . . .	37
Additions to the E Command . . . . .	38
The Y Command (output to 1 or 2 screens) . . . . .	39
The N Command . . . . .	40
The Q Command (Quit) . . . . .	40
The SR Command (Search) . . . . .	40
The ? Command (Help) . . . . .	41
The ?? Command (Help) . . . . .	41
Using GEMSID with MAP files . . . . .	41
The Sample GEM Application and GEM Desk Accessory . . . . .	41
Porting To Other Environments . . . . .	44

## 1: Introduction

Welcome to "Introduction to GEM(TM) Programming". PLEASE REVIEW THIS DOCUMENT BEFORE EXAMINING THE REST OF THE MATERIALS IN THE GEM PROGRAMMER'S TOOLKIT(TM). We hope this document will answer many of your initial questions. You have received an early release of this software so be aware that there may be future changes in both the software and documentation. There will be notice of these updates on the DR SIG that the DR Support Center maintains on CompuServe(R). More on that later.

This document gives an overview of the GEM software and refers you to the appropriate areas of the documentation provided with the GEM Programmer's Toolkit. It also contains explanations on how to use the sample programs, tools, and utilities. In addition, it has suggestions about how to optimize your code and prepare for other environments. Finally, we will introduce you to the Digital Research(R) Support Center which provides access to GEM Programmer's Support while you work on getting your GEM software application up and running.

At DRI(TM) we are very excited about the power GEM software gives the applications programmer and we want to provide you with the most complete set of tools and support available. Please contact us directly with any questions or problems.

## 2: The GEM Programmer's Toolkit

The GEM Programmer's Toolkit is the set of Digital Research tools and utilities that enables an application program developer to use the GEM Virtual Device Interface (VDI) and the GEM Application Environment Services (AES). This toolkit is designed for use under PC DOS, version 2.0 or higher on an IBM(R) PC/XT, but any applications written with it may be moved from environment to environment with a minimal amount of effort.

You will use the Toolkit with a compiler, such as the LATTICE C compiler, and a set of debugging tools. For debugging tools you may use GEM SID with DR Assembler Plus Tools(TM), which includes LINK-86(TM), RASM-86(TM), and LIB-86 or, if you prefer, you can use GEM SID with any compiler that outputs a Microsoft(R)-type .MAP file after converting it to a .SYM file with a special utility that we provide.

### 2.1: Documentation Description

The documents that are provided with the GEM Programmer's Toolkit, DRI product code number 5047, include:

- Introduction To GEM Programming (this document)
- GEM Programmer's Guide, volume 1: VDI
- GEM Programmer's Guide, volume 2: AES
- GEM SetUp Guide

- GEM Desktop User's Guide
- End User License
- cover letter and supplemental notices

## 2.2: Software Description

The software provided as a part of the GEM Programmer's Toolkit consists of 6 disks. The disks consist included are:

Disk 1 of 6

- GEM SYSTEM MASTER ...containing GEM and GEM VDI in addition to the BATCH files used to install the system

Disk 2 of 6

- GEM DESKTOP MASTER ...containing the DESKTOP, OUTPUT and additional fonts

Disk 3 of 6

- GEM DEVICE DRIVER DISK #1 ...containing GEMSETUP and additional device drivers

Disk 4 of 6

- GEM DEVICE DRIVER DISK #2 ...containing various fonts

Disk 5 of 6

- GEM TOOLKIT TOOLS DISK ...containing GEMSID, ICONEDIT, the RESOURCE CONSTRUCTION SET and MAP2SYM

Disk 6 of 6

- GEM TOOLKIT MS SOURCE DISK ...containing the sample GEM application and GEM Desk accessory

To install the software on your IBM PC/XT:

- Load DOS into your computer.
- Place the GEM SYSTEM MASTER in drive A, type GEMPREP and press ENTER. Follow the instructions carefully, answering any questions that you are asked.
- When you return to the DOS prompt, copy all files on the two GEM Device Driver Disks to your GEMSYS sub-directory.
- Finally create a sub-directory called TOOLS and copy the GEM TOOLKIT TOOLS DISK and the GEM TOOLKIT MS SOURCE DISK to that area. It is in this area that you will examine our sample application and accessory before proceeding with the development

of your own application.

It is necessary to install any application before you can execute it. To install an application select it by Clicking on it once, then choose "Install An Application" from the Options Menu. This must be done with each of the applications provided with the GEM Programmer's Toolkit

NOTE: THIS SOFTWARE HAS BEEN SPECIALLY MODIFIED TO EXECUTE ONLY ON AN IBM PC. IT WILL NOT RUN ON ANY OTHER HARDWARE.

### 2.3: Technical Support Services

The GEM Programmer Support (GPS) program provides one year of technical support for one contact person in addition to the GEM Programmer's Toolkit. This program consists of a variety of technical support services streamlined to provide top quality response to problems reported by programmers writing GEM application programs. Through GPS a programmer may communicate with engineers well-versed in the problems that a programmer may encounter when using the Toolkit.

The methods of communicating with the DR Support Center are:

- US mail or any over-night courier for written SPR's (software performance reports),
- CompuServe for communicating with DR engineers or other programmers writing programs with Digital Research Development Tools,
- or directly by telephone, using the telephone and access number provided by DR when you return your completed GPS agreement.

We suggest that you familiarize yourself with the DR SIG on CompuServe, as many of the questions that you might have will be answered there. Additionally, there will be ideas from other application programmers and information on the latest changes to both the software and documentation that will not be available in any other location.

During the prerelease period GPS will provide priority support through CompuServe. Time permitting, we will also be available directly by telephone. If you leave your phone number with your message on CompuServe we will contact you by phone if necessary. We have set up a special procedure to insure the privacy of the messages that you leave for us and that we send to you.

To enter the DR SIG type "GO PCS-13" from the prompt after you have logged on to CompuServe. Log on procedures vary depending on whether you connect directly to CompuServe or through an

alternative communications system. For information on initiating your membership and logging on refer to the "CompuServe IntroPak".

The first time that you log on you will not be able to use the GPS area because we need your CompuServe number to give you access. Following the instructions listed below will enable us to get your CompuServe number and let you into the GPS area.

To send us a private message:

- 1- Log on to DR SIG
- 2- Enter the : Ask The SYSOP area (choose #3)
- 3- Leave a Message (choose L)
- 4- Address the message to "\*SYSOP"

A message left in this manner will not be available to any other person to read. This will create a "thread" of messages that only you can read. We will respond to your message, you will respond to ours. As long as you follow this procedure the messages will remain private. Any message you leave addressed to "\*SYSOP" will be read only by us no matter which area you leave it in.

We will be logged on to the DR SIG throughout the day and often in the evenings so please try to make use of this service.

### 3: The Development Environment

#### 3.1: Hardware Requirements

The recommended hardware for the GEM Programmer's Toolkit(5047) is an IBM PX/XT with 512K RAM, a graphics card and a mouse.

Supported graphics cards include:

- Hercules Monochrome Graphics Card
- IBM Color Graphics Card (in monochrome mode)
- IBM Enhanced Graphics Card

Supported mice are PC Mouse by Mouse Systems and the Microsoft Mouse.

#### 3.2: Software Requirements

In addition to the GEM Programmer's Toolkit you will need this additional software:

- PC DOS version 2.0 or higher
- the LATTICE "C" compiler
- an assembler, linker, and librarian such as those provided in DR Assembler Plus Tools (RASM-86, Link-86, and Lib-86)

#### 4: An Overview of GEM System Software

GEM software provides a unique graphics environment for personal computers, allowing users to work more effectively by manipulating graphic images such as icons, pop-down menus, and windows.

The GEM VDI supports graphics call portability across physical hardware such as graphics screens, peripheral input devices, and output devices. The GEM AES supplements the GEM VDI graphics I/O calls with functions that manage graphics-based user input to application environment metaphors. These include icons, pop-down menus, forms and user-manipulated menus.

##### 4.1: GEM Virtual Device Interface (VDI)

The GEM VDI consists of two main components: the Graphics Device Operating System(GDOS) and the device driver including font information. The size of the GDOS is approximately 6KB while the screen driver and default font can be 32KB to 36KB. It should be noted that the device driver contains much of the output functionality and the size can vary from device to device depending on what functions have been implemented. It is because the functionality is located in this area that programs written to GEM are portable. The programmer merely talks to all devices as though they are one and the same and data is returned from these devices that will indicate to the programmer the limitations of that particular device. It is the responsibility of the application to interpret this data. One of these limitations is the aspect ratio which is returned when a program performs an "OPEN WORKSTATION", see your GEM Programmers Guide, Volume 1. After the aspect ratio of the of the device is returned the programmer must use this information to modify the data sent to the device so that it looks the same on any device regardless of aspect ratio. Through careful, non-machine specific, programming you can write one program that will port to other systems quite easily.

##### 4.1.1: NDC vs. RC Coordinate System

The GEM VDI supports two coordinate systems for the description of graphic space, the Normalized Device Coordinate(NDC) system and the Raster Coordinate(RC) system. The NDC system addresses the graphics display independent of the device coordinate size while the RC system addresses the device in actual device units. These systems are described in detail in section 1 of the GEM Programmer's Guide, Volume 1. In order to make full use of the raster type operations available WE SUGGEST THAT YOU USE THE RASTER COORDINATE SYSTEM.



### .1.2: Raster Text Fonts

The device driver when loaded contains one or more system fonts. Additional fonts are loaded when requested by the application using the "LOAD FONTS" call. The fonts that are loaded are associated with the device driver as specified in the ASSIGN.SYS file. The number of fonts in the ASSIGN.SYS file are determined by the choices made by the user when executing SETUP. The fonts are described in detail in Appendix G of the GEM Programmer's Guide, Volume 1.

There are two different modes when using fonts; Point mode and Absolute mode. Point mode supports text of two sizes, the 1x and 2x. Absolute mode provides for the scaling of text at any value between 0 and 2x. These modes are a function of the device driver and we have placed the capabilities for both modes into our screen drivers while the printer drivers supplied by Digital Research support both modes but round down to the nearest size.

One special consideration when working with text in the work area of a window is that it may be necessary for your application to determine how much text will fit into your window if you are going to allow for the changing of the length of character strings. I.E.: You can send an English language string to a window for one version of a product and switch to a French language string for another, but the length of the string required to convey the same idea in French would be longer and require the placement of text into a different area of the window. In order to facilitate these types of situations DRI has included a set of VDI calls (see the Programmer's Guide volume 1, section 8) specifically for inquiring about the text attributes. To determine the amount of text that will fit within a window you would perform an "INQUIRE CHARACTER CELL WIDTH" and use that value in conjunction with the size of the window, returned by performing a WIND\_GET on the work area of the window. This would not be a problem with Alert Boxes or Dialog Boxes since the string length is determined at the time that they are designed using the Resource Construction Set.

For complete information on TEXT functionality consult your VDI Programmer's Guide, Volume 1.

### 4.1.3: The GEM Metafile

The GEM Metafile is a file used by GEM OUTPUT to send information to peripheral devices. It is also used to send information to other GEM applications. It is constructed of object oriented inscriptions of pictures from GEM applications or bit image files from paint programs and digitizers. The most important consideration for the applications programmer when constructing a GEM Metafile is that you; (1) define the Physical Page Size of the area to be output, and (2) define

the Coordinate System that you are using.

Additional information on the GEM Metafile can be found in the GEM Programmer's Guide, Volume 1, Appendix C.

#### 4.2: GEM Application Environment Services (AES)

The Application Environment Services (AES) consists of several primary elements including the Subroutine Library, the Screen Manager and the Dispatcher. The library provides for windowing, management of objects, mouse movement and more, see the GEM Programmer's Guide, volume 2, section 1.5. The Screen Manager is responsible for the mouse whenever it is outside of the application window and also sends alerts to the application, I.E.: telling it to redraw. The dispatcher is the core of the limited multi-tasking capabilities of GEM. It sets up a "ready list" to rotate processor time to multiple processes rotating control only when the application is making a system call to the AES.

Although the present development system is a single tasking system, PC DOS, if your application can live with the desk accessories then you are all ready for a future multitasking environment.

#### 4.3: Memory Management

For effective memory management a programmer need concern himself with only two procedures:

First, you must link PROSTART to your application at link time. This routine performs two functions; it (1) determines the amount of memory that your application requires shrinking the memory allocated at execution time to that amount, and (2) makes sure that GEM is resident before actually executing your application program to avoid a "hung" system. There are two versions of PROSTART included, PROSTART.A86, for use with an assembler that generates Digital Research format OBJ files, and PROSTART.ASM for use with an assembler that generates Microsoft format OBJ files, I.E.: Lattice "C".

Second, if you need temporary memory you should use the standard PC DOS calls for memory allocation and memory release.

#### 5: Constructing A GEM Application

The first step when designing a GEM application program is to create the menus and dialog boxes using the GEM Resource Construction Set. This is also the time to create the icons or bit images that you will be using. After you have completed this you can develop the

code that will use the resource file.

To write a GEM application program we recommend that you compile your code with LATTICE "C" after creating the source code with any wordprocessor in the non-text mode. It is possible to write code in any language but we are distributing only the bindings for LATTICE "C" at this time. We expect to be providing bindings for PASCAL shortly and other languages after that. It is possible for you to write your own bindings although you might have to adjust the include files produced by the Resource Construction Set.

After you have your code functioning, you can use GEM-SID to debug it. GEM-SID provides the capability to save your graphics screen while doing your debugging and to restore the graphics to the screen. It also makes possible for you to direct your text to one monitor and the graphics to another.

The following instructions will help you in working with the GEM development tools.

## 5.1: Using IconEdit

GEM IconEdit is a tool you use to create icons that represent your applications on the GEM Desktop. The icon you create subsequently appears in a directory window. You can also create icons for use in Dialog and Alert Boxes. To start your application from the GEM Desktop, your end-user can:

1. Double-click on the icon, or
2. Select the icon and then choose the "Open" command from the File Menu.

After you create an icon, you can store it as a disk file. You subsequently use facilities provided by the GEM Resource Construction Set to cause the icon to appear on the GEM Desktop.

### 5.1.1: The GEM IconEdit Screen

This section describes the components you see on the GEM IconEdit screen.

```
=====  
Menu Bar  
=====
```

The menu bar is the top line of your screen. It lists menu titles, which give you access to commands you can use with GEM IconEdit.

You use the GEM IconEdit menu bar in the same way you use the menu bar on the GEM Desktop. When you touch one of the menu titles with the pointer, a menu drops down beneath it. The

drop-down menu lists the commands available from the menu title you display.

Each of the drop-down menus and their associated commands are explained later in this documentation.

```
=====
Editing... Window
=====
```

The Editing... window is your work area. You design the data and mask portions of your icon inside this window.

The window title tells you which of your other windows is currently active. The title reads "Editing Data" when your Data window is the active one; it reads "Editing Mask" when your Mask window is active.

NOTE: You must pay particular attention to which of your windows is active. When you instruct GEM IconEdit to save your icon design in a disk file, only the design from your active window is saved. Likewise, when you recall a previously designed icon from disk to make changes to it, GEM IconEdit places the icon in your active window.

Note the grid inside the Editing... window. Each rectangle in the grid represents a single pixel in your icon. To draw your icon, blacken and whiten the appropriate pixel rectangles to form the image you want.

The pixels toggle between black and white as you touch them with the pointer and click or press the mouse button. To blacken a white pixel, place the pointer on it and click the mouse button. Click again on the same pixel and it reverts to white.

To blacken a series of pixels, place the pointer on a white pixel, press the mouse button, and drag the mouse. To whiten a series of pixels, use the same technique, but begin by placing the pointer on a black pixel.

```
=====
Size of Editing... Window
=====
```

When you start GEM IconEdit, the size of the grid in the Editing... window measures 32 pixels wide by 32 pixels high. This ratio is suitable for most icons that appear on a high-resolution screen; for example, a screen that measures 720 x 348 pixels.

When you are designing icons for screens with a different resolution, you may want to change the ratio of pixels in your

icon. You can do this by choosing the "Size Icon..." command from the Operation Menu. The command is described later along with detailed recommendations about what size of icon you should use for common screen resolutions.

```
=====
Data Window
=====
```

When the Data window is active, whatever you draw in the Editing... window is duplicated in the Data window on a smaller scale. To make the Data window active, place the pointer anywhere inside the window and click the mouse button.

```
=====
Mask Window
=====
```

When the Mask window is active, whatever you draw in the Editing... window is duplicated in the Mask window on a smaller scale. To make the Mask window active, place the pointer anywhere inside the window and click the mouse button.

```
=====
Icon Window
=====
```

When you place the pointer in the Icon window and click the mouse button, GEM IconEdit combines the designs from your Data and Mask windows to show you what your finished icon will look like.

#### 5.1.2: GEM IconEdit Concepts

```
=====
Icon Names
=====
```

You must name icons you create with GEM IconEdit when you want to store them on disk. Icon names must conform to PC DOS filename conventions.

You can use a filetype extension with icon filenames. You can use any extension you wish, but we recommend you use the characters "ICN".

```
=====
Recommended Icon Naming Conventions
=====
```

This section recommends conventions for naming your icon file: when you store them on disk. These are only recommendations and you are free to use any names you wish. The recommended conventions are shown in the following table.

Position	Value	Interpretation
1	I	Identifies an icon file.
2	x	Identifies type of icon: A = Application icon D = Document icon G = Generic icon
3-6	xxxx	Descriptive abbreviation. Use any four characters that adequately describe what the icon represents. See the examples that follow.
7	x	Data or Mask identifier: D = Data portion of icon M = Mask portion of icon
8	x	Resolution identifier: H = High-resolution L = Low-resolution

=====  
Examples of Icon Naming Conventions  
=====

IGTRSHDL.ICN

A generic icon (IG) representing a trash can (TRSH). It is the data portion of the icon (D) for low-resolution screens (L).

IADRAWDH.ICN

An application icon (IA) for a draw program (DRAW). It is the data portion of the icon (D) for high-resolution screens (H).

IDGRPHML.ICN

A document icon (ID) for a graphing program (GRPH). It is the mask portion of the icon (M) for low-resolution screens (L).

=====  
Icon Masks and Data  
=====

The basic purpose of an icon mask is to make sure your icon does not blend into the desktop and "disappear" when it is high-lighted. The GEM Desktop highlights an icon by showing it in reverse video when the icon is selected.

By properly coordinating the settings of each pixel rectangle in the data and mask portions of your icon, you can ensure

that the icon appears correctly when it is highlighted. Furthermore, you can achieve special effects by this coordination. Experiment with this feature to get the results that you desire.

GEM IconEdit uses Boolean algebra when merging the data and mask portions of your icon. It applies the same Boolean algebra rules when placing the merged icon on the GEM Desktop. The following table shows you what results to expect, on a pixel by pixel basis, in the non-selected states for both monochrome and color monitors.

Data	Mask	MONOCHROME Non-selected
Black	White	Black
Black	Black	Black
White	Black	White
White	White	Clear*

\* - If the pixel in the data portion is white and the matching pixel in the mask portion is also white, this has the effect of creating a "hole" in the icon. Consequently, you are able to see through the icon to whatever is beneath it; which is normally the Gem Desktop background.

### 5.1.3: GEM IconEdit Menus

This section describes the commands you see on the GEM IconEdit menus.

#### 5.1.3.1: Desk Menu

The Desk Menu gives you access to your desk accessories and other useful information.

```
=====
"IconEdit Info..."
=====
```

When you choose this command, a dialog box appears. Information inside the dialog box tells you:

1. The version number of the GEM IconEdit you are using.
2. The date your version of GEM IconEdit was created.
3. The name of the DRI engineer who wrote GEM IconEdit.

## 5.1.3.2: File Menu

The File Menu lists commands you can use to create, save, and modify your icons.

```
=====
"New"
=====
```

Use this command to create a new icon. If you have been working on an icon when you choose the "New" command, GEM IconEdit abandons all work you have done in any window.

Every data and mask icon you create with GEM IconEdit must have a name. But, GEM IconEdit does not ask you to provide a name until you save the data or mask. When you choose "Save" or "Save as..." from the File Menu, the Item Selector box appears for you to type the name under which you want to save the file.

```
=====
"Open..."
=====
```

Use this command to make changes to a pre-existing data or mask icon. After you choose this command, you see the Item Selector box on your screen. Select the file you want and GEM IconEdit places it in the Editing... window for you to make whatever changes you wish.

Note that GEM IconEdit also places the icon in your active window. Therefore, you should click on the desired window (Data or Mask) before you choose the "Open..." command.

```
=====
"Save"
=====
```

Use this command to store the icon from your active window on disk. The icon remains in the active window so you can continue making changes.

If you are creating a new icon and this is the first time you have saved it on disk, GEM IconEdit displays the Item Selector box for you to enter a file name.

If you are changing a pre-existing icon, GEM IconEdit stores it under its original name.

```
=====
"Save as..."
```



=====

Use this command to store the icon from your active window on disk, but under a name different from its original one. The original icon also remains on the disk under its original name.

GEM IconEdit displays the Item Selector box for you to enter the name under which you want to store the revised icon. After you save the icon, GEM IconEdit leaves it in your active window so you can continue working on it.

=====  
 "Quit"  
 =====

This command stops GEM IconEdit and returns you to the GEM Desktop.

#### 5.1.3.3: Operation Menu

The Operation Menu lists commands you can use to manipulate the windows on your GEM IconEdit screen. It also contains commands that help you create icons more quickly and efficiently.

=====  
 "Grid On/Off"  
 =====

Alternately turns the grid lines ON and OFF in the Editing... window. The command has no other effect.

=====  
 "Clear Icon"  
 =====

Erases the icon from the Editing... window and active window and returns all pixel rectangles to their original color, i.e., white.

=====  
 "All Black"  
 =====

Blackens every pixel rectangle in the Editing... window and the active window.

=====

"Invert Icon"  
 =====

Changes every pixel rectangle in the Editing... window and active window. That is, all black pixel rectangles change to white, and vice versa.

=====  
 "Size Icon"  
 =====

Use this command to change the number of pixel rectangles in the height and width of your icon. This is a necessary procedure when you design icons for screens with different resolutions. For example, if you design an icon that is 32 pixels high by 32 pixels wide, it may appear proportionally correct on a screen of one resolution but incorrect on a screen of different resolution

The following table lists common screen resolutions and the recommended icon size for each.

Resolution	Recommended Icon Size
720 x 348	32 x 32
640 x 200	48 x 24
640 x 400	32 x 32

Any time you start GEM IconEdit, the size of the grid in the Editing... window is 32 x 32. If you want to change the size, choose the "Size Icon..." command and the "Icon Size:" dialog box appears on your screen. Choose the width and height for your icon by clicking on the appropriate values. Then click on the "OK" exit button or press the Enter key.

The Icon Size dialog box disappears and GEM IconEdit changes the size of the windows on your screen to match your specifications.

Note that there are other sizes available in the Icon Size dialog box than those listed in the preceding table. The additional sizes are available because you can use GEM IconEdit to create images other than icons. For example, you can create an image of your company logo so you can show it in the dialog boxes you display as part of your software. This is done with the GEM Resource Construction Set.

=====  
 "Data to Mask"

=====

Copies the design from your data window into your mask window. This command is a useful "shortcut" for creating special visual effects between the data and mask portions of your icon.

=====  
 "Mask to Data"  
 =====

Copies the design from your mask window into your data window. It works exactly opposite of the "Data to Mask" command but can be used to achieve similar effects.

## 5.2: Using the Resource Construction Set

The RCS is the tool you use to make resources such as drop-down menus, dialog boxes, and alert boxes. The RCS also takes the icons and bit images you draw with IconEdit and makes them usable as resources. You save resources in files, which contain trees of objects such as text strings, icons, and exit buttons.

### 5.2.1: The GEM RCS Screen

The RCS screen comprises three parts:

- the View Window, in which editing takes place
- the Partbox Window, containing trees or objects
- the Desktop, containing the clipboard and trash can

Only one window is selected at a time. You can perform operations in the View or Partbox windows only when they are selected. To select a window, click within it. When a window is currently selected, the title bar is striped.

If one window is atop another, the topmost is currently selected. To bring the bottom window to the top, click anywhere within it. This operation is called "topping."

You can size and move the Partbox and View windows (but not the Desktop). To size the Partbox or View Window, first make sure it is selected. Touch the mouse pointer to the size boxes in their lower right corners and drag (the drag operation is described in your GEM User's Guide Glossary). To move a window, touch the pointer to the title bar, hold the mouse button down, and drag.

Most trees and objects are moveable within the View Window by selecting and dragging. (Note: in this doc, "tree" means a

tree icon; "object," means one of the things you add to a tree once you open the tree icon.) You can also size objects within the View Window. When you are editing a tree, objects contain an implied size box, or "handle" in the lower-right corner, in the sameplace as size boxes for windows. You cannot move around the contents of the Partbox Window; they are static. In the Desktop, you can rearrange the Trashcan and Clipboard. Note: certain trees and objects carry restrictions on movement and sizing. These will be covered later.

=====  
 Mouse Operations  
 =====

Most of the mouse operations used in the RCS are described in your GEM User's Guide Glossary. A couple of these operations have special meanings in the RCS.

- click always selects an object
- a double-click always opens an object
- the effect of a drag depends on where you drag it.
  - \* drag a tree or object from the Partbox Window to make a copy in the View Window.
  - \* drag within the View Window to move a part
  - \* drag from the View Window to the clipboard to cut a tree or object
    - \* dragging a tree or object to the Clipboard cuts it from the View Window and stores it temporarily in the Clipboard. If you cut a tree, a small T appears in the Clipboard; if an object, a small O appears. The clipboard can contain only one tree or object at a time. If you cut two items, you can paste only the second clipping; the first is lost. The clipboard clears every time you save or load a file.
    - \* drag from the clipboard to the View Window to paste a tree or object. When you paste a tree, the Name Dialog appears so you can change its name or type.
    - \* to delete an tree or object in the view window, drag it to the Trashcan. Deleted items are irretrievable.
- <SHIFT>-drag modifies a drag by making nondestructive copies. Drag the object you want to copy while holding down the <SHIFT> key.

\* <SHIFT>-drag in the View Window to make duplicates of an object or tree

\* <SHIFT>-drag from the View Window to the Clipboard to place a duplicate object or tree in the Clipboard

\* <SHIFT>-drag from the Clipboard to View Window to paste a duplicate item on the View Window without clearing the Clipboard

- To size an object in the View Window, place the pointer on its size handle (its lower right corner) and drag.

- the <CTRL> key modifies the operations above by taking you up one level in the visual hierarchy of your object tree. All <CTRL> operations apply to both the selected object and its progeny.

\* <CTRL>-click selects the parent of the object selected. If, for example, you have an object covering the sizing handle of the box that contains it, you cannot size the containing box; when you try to select the sizing handle, you get the object instead. According to the rules of visual hierarchy, the containing box is the parent of the object it contains. Therefore, <CTRL>-clicking on the sizing handle of the box (or rather, on where the handle would be if you could see it) selects the parent box rather than the child object. (Visual hierarchy is explained later in this reference section.)

\* <CTRL>-drag moves the parent of the object selected (moving a parent moves all its progeny as well).

\* <CTRL>-<SHIFT>-drag duplicates the parent of a selected object and all the progeny of that parent

- All copy, move, and delete operations affect both the selected object and its progeny

## 5.2.2: GEM RCS Menus

At the top of the RCS screen are four menus, as follows:

### 5.2.2.1: Desk Menu

```
=====
About  the R.C.S
=====
```

- Displays the version number, author, and copyright date of the RCS.

- other entries appear if you have Desktop accessories installed

#### 5.2.2.2: File Menu

=====  
New  
=====

Clears the View Window of trees and objects. A warning message appears to remind you to save your workspace.

=====  
Open  
=====

Opens a file, bringing up the Item Selector Dialog so that you can enter a filename. If a tree or object is currently selected, Open opens that instead. If the workspace is already in use, you are warned.

=====  
Merge  
=====

merges a second file with the file currently open. Brings up the Item Selector Menu for entry of the second filename.

=====  
Close  
=====

Closes the most recently opened file or tree.

=====  
Save  
=====

Saves the current file. Trees are saved in files with a filetype of RSC (for resource). Trees and the files they reside in are named separately. The names and types of trees and objects are saved in the DEF file. You can create other types of files, depending on the current Output setting (see the Global Menu).

=====  
Save As  
=====

Saves a file under a new name. Brings up the Item Selector Dialog. If you type in a new filename, include the filetype (the filetype RSC is recommended). There are no naming conventions for the RCS, but generally you name your RSC file after your program: for example, FOO.EXE and FOO.RSC.

=====  
Abandon  
=====

Reverts to the last-saved version of your file. If you are not satisfied with changes you made to a particular file, Abandon clears the changes and gives you a fresh copy.

=====  
Quit  
=====

Takes you back to the Desktop. If you have not saved your work, a warning message appears.

### 2.2.3: Options Menu

=====  
Info  
=====

Displays information about the file, tree, or object selected.

=====  
Name  
=====

Names or renames a tree or object. You may also be able to change the type of the tree or object.

=====  
Hide  
=====

Makes an object or portion of a tree disappear.

=====  
Unhide  
=====

Makes the hidden progeny of an object reappear. To expose

a hidden object, select its parent and choose Unhide.

\*\*\*\*\*  
Sort  
\*\*\*\*\*

Sorts and object's progeny by X and Y coordinates. Useful for achieving a tidy appearance and ensuring that objects are drawn in the order in which you arrange them on the screen.

\*\*\*\*\*  
Flatten  
\*\*\*\*\*

Removes the tree level selected, "flattening" the tree. Suppose a three-level tree of parent, child, and grandchild. If the child is selected, Flatten removes the child and promotes the grandchild to direct relationship under the parent.

\*\*\*\*\*  
Snap  
\*\*\*\*\*

Makes the position and size of an object in the View Window automatically align with the nearest coordinates on an invisible character grid. Used only with Free Tree, which does not have automatic snap. Without snap, it is difficult to get objects lined up exactly.

\*\*\*\*\*  
Load  
\*\*\*\*\*

Loads a bit image or icon.

#### 5.2.2.4: Global Menu

\*\*\*\*\*  
Output  
\*\*\*\*\*

Creates output files. The Output Files Dialog comes up, offering the following choices:

Application Binding Files		Source Files for Resource
C	(*H)	*.C
Pascal	(*O)	



Note that .H and .O files are included when compiling your C or Pascal application. They supply actual values associated with names you entered in the Name Dialog. A .C file is a C-language source of your resource, which you can hand edit and use with RSCREATE.

=====  
Safety  
=====

Sets the level of protection on your files, from heavily guarded to unwatched. When you select Safety, the Resource Editing Mode Dialog appears. In it you have the choice of three levels of protection: locked, in which tree structure cannot be changed, normal, which issues a warning before rearranging trees, and expert, in which anything goes.

### 5.2.3: Partbox Window Contents

The Partbox Window contains the object trees, in the form of icons, from which you build your resources. (The GEM Programmer's Guide explains object trees and their function in the GEM system.) The kinds of trees available are as follows:

=====  
Unknown  
=====

Serves as a place marker for trees of unknown type. If the DEF file of the tree you are working on was somehow lost (or never existed: for example, you created it with RSCREATE) use Unknown. If you have a hand-made tree containing non-standard objects, use Unknown.

=====  
Free  
=====

The most general and flexible kind of tree. Other tree types have their own editing constraints. The only rule for Free Trees is that visual hierarchy prevails. Free Trees are mainly useful for building system-specific trees; generally, they are not portable.

=====  
Menu  
=====

Specifically intended for application menus, as described in

the GEM AES Menu Library.

```
=====
Dialog
=====
```

Creates dialog boxes for user input. A stricter form of Free tree in that character snap is enforced. See Dialog Boxes in the Form Library section of the GEM AES manual.

```
=====
Alert
=====
```

Warns the user of impending consequences or that something is not right. A stricter form of Dialog tree. See Alerts in the Form Library section of the GEM AES manual.

#### 5.2.4: Tree Operations

This section describes the various things you can do with trees.

```
=====
Starting a New Tree
=====
```

To begin work on a new tree, select the Partbox Window. In the Partbox Window, select the kind of tree you want and drag a copy of it to the View Window. Drop the copy within the window. The new tree opens into the Name Dialog.

```
=====
The Name Dialog
=====
```

The Name Dialog appears when you create a new tree by dragging it from the Partbox Window, pasting from the clipboard, or duplicating a tree in the View Window by <SHIFT>-dragging. In the Name Dialog you can name your new tree and change its type. After you exit the Name Dialog, the tree closes and retreats to its place in the View Window. To continue work on the new tree, you must open it.

```
=====
Naming a Tree
=====
```

Trees must have names so that you can refer to them in your

code. When you drag a new tree to the View Window, it automatically takes a default name based on its order among the trees that already exist: for example, TREE1, TREE2, etc. (You can have up to 55 trees in the View Window.)

```
=====
Changing a Tree's Type
=====
```

The Name Dialog also accomodates changes in tree type. The current tree type is highlighted. Select the new type by clicking on it. Be careful about changing the type of a tree to Menu or Alert if you aren't absolutely sure of its nature. Change it to a Dialog first so you can open and look at it without undesirable consequences. (Trees mistakenly typed as Menus are especially prone to hanging the system.)

```
=====
Rearranging Trees
=====
```

You can rearrange trees in the View Window.

- To move a tree to a new position between two other trees, drag it between them and let go. The two trees part, allowing the newcomer to slip in.

- To move a tree to the end of a list of trees in your View Window, drag it straight down and let go.

- To move a tree to the front of the list, drag it so it touches the first tree and let go. The former first tree shifts right and the new tree takes it place.

```
=====
Opening a Tree
=====
```

Use either of the following operations to open a tree:

- double click on the tree
- select it and choose Open from the File Menu. When you open a tree, its name, whether the name you gave it or a default name, appears in the title bar of the View Window. In the Partbox Window, prototype objects appear where the prototype trees were formerly.

```
=====
Closing a Tree
=====
```

To close a tree, click on the Close Box at the far left of the title bar or pick Close from the File Menu. Closing a tree is not the same as saving it to disk. Trees are saved in RSC files, which must be named separately upon saving.

### 5.2.5: Objects

The objects that flesh out your tree appear in the Partbox Window when you open the tree. These available objects differ according to what kind of tree you are working on. Objects are dragged onto the View Window. The surface of the View Window, when the tree is opened, constitutes the highest level of the tree (the root object).

```
=====
Partbox Window Contents and Tree Type
=====
```

For Free Trees and Dialog Trees, the objects available in the Partbox Window are the same. Menu Trees and Alert Trees have their own individual Partbox contents. Unknown Trees do not have a corresponding Partbox. If you want to add objects to an Unknown Tree, you have to change its type first. The kind of tree you have open is reflected in the title bar of the Partbox window: if the tree is a Dialog, for example, the title reads "Dialog Partbox."

```
=====
Free Trees and Dialogs: Objects Available
=====
```

The following objects are available in Free Tree and Dialog Tree Partboxes. The drawings below are intended to suggest the way the prototype objects appear in the Partbox. The names these objects assume in the Object Library section of the GEM AES are given in parentheses.

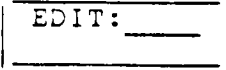


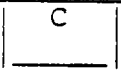
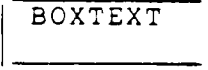

o 

BUTTON
--------

(G\_BUTTON) Boxed string that the end user selects to indicate his choice among alternatives

o STRING

(G\_STRING) Boxless character string, generally consisting of explanatory text for the user.

- o EDIT: \_\_\_\_\_  
(G\_FTEXT) Editable text field
  
- o   
(G\_FBOXTEXT) Editable text field with box
  
- o   
(G\_IBOX) Hollow box through which you can see the dither pattern or text beneath
  
- o   
(G\_BOX) Opaque box
  
- o TEXT  
(G\_TEXT) Formatted text, for which you can specify sizes, colors, fonts, and masking
  
- o   
(G\_BOXCHAR) Single character in a box, nonformattable
  
- o   
(G\_BOXTEXT) Same as TEXT, but with a box
  
- o   
(G\_ICON) Makes resource of icon made with IconEdit



(G\_IMAGE) Makes resource of bit image (data field only) made with IconEdit.

```
=====
Menus: Objects Available
=====
```

The objects in the Menu Partbox Window are limited to those components practical in menus. They are as follows:

o TITLE

--prototypical title for your menu bar

o ENTRY

--prototypical entry to be placed in a menu drop-down

o -----

--separator line. Divides entries into logical groups for user's convenience



--box. Used as placeholders for elements such as dither patterns and colors that you add at runtime. (Observe the GEM Draw menus.)

```
=====
Alert Trees: Objects Available
=====
```

Alerts generally employ the standard GEM system warning icons, a message, and away for the user to get out of the alert.

o 

BUTTON
--------

    --exit button, with characteristic heavy borders of exit buttons.

o Message Line

    --nonformattable

o NOTE icon

    --depicts a hand with upraised forefinger. A prototypical NOTE icon is already in place in the View Window when you open an Alert Tree, because NOTES are the most commonly used alerts.

o WAIT icon

    --depicts a question mark

o STOP icon

    --depicts a hand with palm exposed

#### 5.2.6: Object Operations

The following descriptions apply specifically to objects in Free and Dialog Trees and generally to Menu and Alert Trees. Exceptions in Menu and Alert Trees are noted.

=====  
Adding Objects to Trees  
=====

To move an object from the Partbox window to your tree, first make sure the Partbox window is selected. Drag a copy of the object into the View Window and drop it. Note that this is the same procedure you use to copy trees into the View Window. Remember that a <shift>-drag makes additional copies of objects. If you want to add three buttons to your tree, for example, you can drag one down from the Partbox and copy it twice rather than dragging down three separate buttons.

For Menu Trees:

=====

You can drag objects only to their appropriate places. Titles, for example, belong in the menu title bar after the default entries "Desk" and "File." You can drag down Entries, Separator Lines, and Boxes only when you have somewhere to put them: that is, after you select the title of the menu you want, and the white box that will contain your menu, drops down.

You cannot drag objects to or from the Desk menu. Nor can you edit the contents of the Desk Menu, except for the first line.

For Alert Trees:  
=====

You cannot have more than one warning icon in an alert tree. To replace the default NOTE icon with another kind of icon, drag the other icon anywhere within the Alert Tree. (If you do not want an icon at all, drag it to the trash.) You are limited to three or fewer buttons and five or fewer message lines.

=====

Moving and Sizing Objects in the View Window

=====

To move an object around in the View Window, drag it. To discard an object, drag it to the trashcan. To cut it, drag it to the clipboard. To make a duplicate of it on the clipboard, <SHIFT>-drag it to the clipboard. To duplicate it within the View Window, use <SHIFT>-drag.

You can size any object by dragging the lower-right corner away from or toward the center of the object. Objects without visual extents around them, such as strings, formatted text, editable text fields, bit images, and icons, size automatically when you change the data they contain. Objects with visual extents require manual sizing.

For Menu Trees:  
=====

The default first entry in the Desk menu, "Your message here," is obligatory. You can (and should) change the text; you cannot, however, move it or throw the entry away and have no text there at all. Nor can you move, edit, or throw away the remaining default objects, the Separator Line and Desk Accessories 1-6. Think of the Desk Accessory as placeholders. They fill in appropriately at runtime if you have desk accessories (such as a clock or calculator). Otherwise, the unused slots disappear at runtime. The prototype File menu provides a first entry, Quit, for your convenience. To add



other entries, size the drop-down menu by <CTRL>-dragging the lower rightcorner.

For Alert Trees:

=====

You cannot size objects. They automatically expand or contract to contain your text. You can drag Alert Tree contents to the trash or clipboard only.

=====

Opening an Object

=====

To open an object, double-click on it or select it and choose Open from the File Menu. The object expands into an Attribute Dialog, in which you set its characteristics.

For Menu Trees:

=====

In the Desk menu, only the first entry opens.

For Alert Trees:

=====

The icon objects do not open.

=====

Attribute Dialogs

=====

Certain objects are compatible, sharing like attributes and identical Attribute Dialogs. There are five classes among the objects.

1. Unformatted text (Buttons, Strings, and Titles) constitutes the first class
2. Boxes (Boxes, Hollow Boxes, and Boxchars) constitute the second
3. Formatted text (Text, Boxtext, Editable Text, and Editable Text-in-a-Box) constitutes the third.
4. and 5. Icons and bit images are compatible with nothing and form their own one-member classes.

### Class 1 Attribute Dialogs:

=====

For Buttons and Strings, the Attribute Dialog consists of a set of object flags (described in your GEM Programmer's Guide AES under Object Library) and a field for text entry. The flags are toggles. Select them to turn them on or off. At the bottom of the Attribute Dialog, the current text, namely "STRING," or "BUTTON," occupies the text field. Backspace over it and type the text you want. (Instead of backspacing, you can use <ESC> to clear the whole string atonce.)

### Class 2 Attribute Dialogs:

=====

For Boxes, Hollow Boxes, and Boxchars, the Attribute Dialog contains the same object flags as in class 1. In addition, the Attribute Dialog contains options as follows.

- background dither patterns--eight available
- background color--offers 16 VDI color indices in hexadecimal
- border color--same as for background color
- border width--seven widths available
- character--enter any character
- character color--similar to background color. Offers same colors.
- masking rules--select replacement mode or transparent mode for characters. You can stamp characters on top of background or replace it. To choose a dither, color, or border width, click on one of the choices visible or click on the left or right scroll arrows to expose additional choices. The current selection appears in the "Selected" box.

### Class 3 Attribute Dialogs:

=====

For Text, Boxtext, Editable Text, and Editable Text-in-a-Box, the Attribute Dialog includes all the features of class 2 with additions pertaining to textsize and format:

- font--choice of large or small fonts for text
- justify--justifies text left, center, or right
- PTMPLT>EDIT:~~~~~--template for text entry

- PVALID>~~~~~XXXXXX--validation field for text entry.
- PTEXT>~~~~~--text entry field

The template, validation field, and text entry field (P fields) are explained in your GEM AES in the Object Library section, under "TEDINFO Structure." Note that in the RSC, PTMPLT fields use a tilde (~) in place of an underline ( \_ ) to avoid confusion between the field itself and the placeholders for editable characters. The following example, constructing an editable field in which the end user types in the date, illustrates the use of the template, validation field, and text entry fields.

```
PTMPLT>Today's Date:  __/__/__
PVALID>-----99~99~99_____
PTEXT>-----01~01~85_____
```

Explanation of template: To enter "Today's Date: " instead of the default, "EDIT:," backspace over EDIT and type in the new string. The tildes in the template are placeholders for characters that will later change. They are the only part of the template that the end user will be able to edit. The slashes, as non-tildes, are permanent features of the template.

Explanation of validation field: The validation field determines what kind of characters the end user can enter: it determines what constitutes valid input. The tildes under "Today's Date: " and the slashes means that those characters in the template are literal strings and will appear as they are; they require no validation. The nines under the tildes in the template mean that the enduser must type in digits where the nines occur.

Explanation of text entry field: The digits you type under the nines in the validation field will appear as the default entry. The end user must type over them to change the date.

#### Icon Attribute Dialogs:

=====

Icon Attribute Dialogs contain object flags as for other classes and special attributes for coloring icons and mixing text and icons.

- foreground color--determines color of lines in icon
- background color--determines color between lines in icon
- text--to insert text within, above, or below icon. Useful for labeling.
- character--to insert a single character within icon.

Example: floppy disk icons. The Icon Attribute Dialog contains two Locate Boxes: a Text Locate Box under the Text field, and a Character Locate Box under the Character field. For positioning Text within your icon, click on any of the three levels in the Text Locate Box. To position a Character within your icon, click on any square within the grid.

The note at the bottom of the menu indicates that you must load a data or maskfile to change an icon in the RCS. Select an icon object from the View Window, go to the Option Menu, and pick Load to load a mask or data file.

#### Bit Image Attribute Dialogs

=====

Bit Image Attribute Dialogs contain the standard object flags and an Image Color attribute. Like the color selectors in other attribute dialogs, this offers sixteen VDI-index colors. Note that you must load new data through the Load entry in the Options Menu.

#### Exiting an Attribute Dialog

=====

Hit <RETURN> or select OK to assign the attributes and close the object.

### 5.2.7: Tips And Concepts

#### 5.2.7.1: Visual Hierarchy: Relationship between Trees and Objects

The highest level of a tree is the surface of the View Window when you open the tree. Any solitary object you drag to the surface is a direct child of the View Window surface. An object within another object--a button within a box, say--is the child of the containing object. Because this hierarchy is apparent on sight, it is called "visual hierarchy."

"Note that according to the rules of visual hierarchy, a parent object must completely contain its progeny; objects that jut out from an object that otherwise contains them are not related to that object. Therefore, sizing objects up or down can affect their relationships to other members of the tree. With objects that are supposed to be nested at the same level, such as radio buttons, you must take care that the visual hierarchy of the tree accurately reflects your intentions.

Remember that a mouse operations modified by <CTRL> takes you up a step in the hierarchy and that Flatten, a choice

in the Options Menu, removes a level in the hierarchy. Recall too that any delete, move, or copy affecting an object also affects its progeny.

#### 5.2.7.2: Notes on Making Icons and Bit Images

Icons and bit images are drawn with IconEdit and are saved in .ICN files. An icon consists of data and a mask. A bit image consists of data only. You must load .ICN files into the RCS to turn them into resources. As bit images are but simpler versions of icons, in that they contain data only, the notes that follow focus mainly on icons, with bit-image exceptions indicated.

To load a .ICN file, drag an icon object from the Free or Dialog Partbox Window and drop it in the View Window. Select the icon object. Go up to the Options Menu and select Load. The Icon Load Dialog appears, and you can choose to load the data or mask, or both. For bit images, no Load Dialog appears.

When you load the data or mask, the Item Selector Dialog, appears so that you can click on the .ICN file you want. If you choose both mask and data, the Item Selector appears twice, once for the data and once for the mask. Exit the Item Selector Dialog. The data or mask you selected (or both) now appear in the View Window. If you loaded just the data, click on the icon again and choose Load to load its mask. When both data and mask are loaded, the files merge. Open the icon by double clicking and set its attributes. When you save the object tree containing your icon, it becomes a resource just like any other.

Remember that if you want to redraw the icon image itself, you have to go back to IconEdit.

#### 5.2.7.3: Memory Limits

You can have a maximum of sixty trees (on certain machines you can see only fifty-five of these) occupying a total of 30K bytes. These trees can contain cumulative maximum of 1,250 objects, of which 500 can have names. As names are stored in a separate index, they do not count against your 30K bytes. When in doubt, drop down the Options Menu and chose Info. You will be warned if you reach the end of your memory buffer or name index.

#### .7.4: Name Conflicts with Merge

Check for name conflicts among trees and objects when you merge files. When you duplicate a tree or file, the new copy loses all name information. If you merge a file

containing names already defined in the current file, the RCS makes up new names for the duplicates.

#### 5.2.8: Getting Started, a Sample Session with the RCS

This walk-through directs you in the making of a simple Dialog Tree. The steps you take here apply generally to all the trees in the RCS.

##### Starting Up =====

To begin work, select the RCS.EXE icon from the desktop. Because the RCS is a programmer's tool, the RCS.EXE icon depicts a hammer.

##### Choosing the Dialog Tree Option =====

Inside the RCS, you notice five icons in the Partbox Window at the top of the screen. Click on the fourth, which contains the word "Enter." This is the Dialog Tree icon.

Holding the mouse button down, drag the Dialog Icon to the center of the window labeled "Resource Construction Set." This is the View Window. Release the mouse button.

##### Naming the Dialog Tree =====

When you release the mouse button, a notice appears in the center of the screen, informing you that you can change the name and type of your Dialog Tree.

Backspace over the default name, TREE1, and type "FOO."

Next click on the OK button. The dialog tree labeled FOO appears in the upper-left corner of the editing window.

##### Opening the Dialog Tree =====

Click once in the view window to select it.

Double-click on FOO to open it. When FOO is open, its name appears in the title bar of the View Window. The View Window now shows a blank dialog tree, FOO.

##### Assembling the Objects

=====

Above the View Window, the Dialog Partbox window contains the objects you use to make your dialog tree.

Click once in the Dialog Partbox Window to select it, then drag the object labeled BUTTON to the FOO tree. Drop it in the middle of the View Window. Go back for another button and drop it below the first.

Now get some text by dragging down a STRING object in the same way you dragged the buttons. Drop it to the left of the buttons.

#### Customizing the Objects

=====

Click within the View Window to top it. Then double-click on the first button to open it. An Attribute Dialog appears, consisting of a list of object characteristics. Note that SELECTABLE is already chosen by default. Click on EXIT.

At the bottom of the list, backspace over "BUTTON" and type in "YES." When you are finished, select OK.

Open the second button and make it a selectable exit button like the first. This will be the default button. Click on DEFAULT. Over "BUTTON" type "CANCEL." When you are finished, select OK.

Now open the STRING object by double-clicking on it. The same Attribute Menu appears. Backspace over the word "STRING" and type "Do you want to continue?" When you are through, click on OK.

#### Naming an Object

=====

Next, give the button a name so that you can refer to it in your code. Click on the button to select it. Then go up to the Options Menu and selectName.... When the Object Name Dialog comes up, type in FOOBUTN. Click on OK when you're ready. The string is now associated with the name FOOSTRNG.

#### Rearranging Objects

=====

You might find upon closing the string object that the string overlaps the buttons or that the layout of string and buttons looks untidy. To move the string, click on it and drag. You can also drag the buttons by clicking anywhere within them. Move the three objects around till they look like the example

below:

```

-----
|  yes  |
-----

```

Do you want to continue?

```

-----
| cancel |
-----

```

Closing the Dialog Tree  
=====

Now you are ready to close FOO. Click on the Close Box, which appears at the far left of FOO's title bar.

Saving the Dialog Tree in a File  
=====

Trees are saved in .RSC files. Go to the File Menu and choose Save As.... The Item Selector Menu appears. After "Selection:" on the right side of the menu, type "FOOFILE.RSC." When you are finished, click on the OK button. FOO is now saved in file FOOFILE.RSC. In the process of saving your file, a .DAT file, containing naming and typing information on your objects and trees, was automatically created.

Close and Exit  
=====

To close "FOOFILE.RSC", click on the CLOSE box.

To exit the RCS, select QUIT from the File Menu. You are now back to the Desktop.

### 5.3: Using GEM SID

This is an addendum to SID-86 for operation with the Digital Research Graphics Environment Manager (GEM). This upgraded version of SID-86 is named GEM-SID. With the exceptions described below, GEM-SID operates in the same manner described in the SID-86 Programmer's Guide for PS-DOS manual.

```

=====
LOADING GEM-SID

```



=====

To load GEM-SID, enter the GEM Programmer Tools desktop and select the GEM-SID icon.

If GEM is not present when GEM-SID is executed, the following error message will appear in the GEM-SID banner:

!!!! Warning: GEM VDI not present !!!!

Without GEM present in memory, GEM-SID will operate in the same manner as SID-86.

5.3.1: Additions to the E Command

GEM-SID features large symbols that have both a segment and an offset. In order to debug GEM applications with large symbols, the large symbol segment values may have to be corrected.

To ensure the correct large symbol addresses for the GEM application, two new forms of the E command have been added. These forms are:

E filename -symfilename  
E filename +symfilename

If, after loading the GEM application and symbol file into GEM-SID, the symbol addresses are missing the segment value, or are otherwise in error, reenter the E command using the E filename -symfilename form. If the symbol addresses are still wrong, use the E filename +symfilename form. The effects of the minus and plus signs in front of the symbol table filename are explained below.

=====  
The Minus Sign  
=====

A minus sign (-) in front of the symbol table filename makes the symbols relative to the GEM application's PSP (Program Segment Prefix) plus 10h paragraphs.

When the minus sign is specified in front of the symbol table filename, the SYM file's segment values are constructed by adding 10h paragraphs to the PSP segment value (which is generally the beginning of the code segment). This sum is then added to the code and data segment values provided in the SYM file to generate the correct segment values for the large symbols.

The equations used to construct the code and data segment values for the SYM file can be illustrated as follows:

*Use minus sign if  
symbol table  
is relative to  
PSP. Use plus  
if it is absolute.*

```

PSP Segment
+ 10h Paragraphs
+ SYM file CS
-----
= Large Symbol CS

```

```

PSP Segment
+ 10h Paragraphs
+ SYM file DS
-----
= Large Symbol DS

```

```

=====
The Plus Sign
=====

```

When a plus (+) sign is placed in front of the symbol table filename, the start of the SYM file is offset by the value of the user code segment.

The equations used to construct the code and data segment values for the SYM file can be illustrated as follows:

```

Current User CS
+ Current SYM file CS
-----
= New SYM file CS

```

```

Current User CS
+ Current SYM file DS
-----
= New SYM file DS

```

### 5.3.2: The Y Command (output to 1 or 2 screens)

The Y command controls the graphics to text conversions under GEM-SID. The forms are as follows:

```

YGE
YGD
YG
YME
YMD
Y

```

The YGE command enables the graphics image buffer. This command causes the graphics image appearing on the screen to be saved in a buffer before GEM-SID switches the screen to text. The YG command is used to recall the graphics image buffer to the screen.

The YGD command disables the graphics image buffer. After entering the YGD command, the graphics image is not saved in the graphics image buffer.

The YG command restores the graphics image to the screen. The YG command only functions if the YGE command has enabled the graphics image buffer. Press any character key to restore the GEM-SID text to the screen.

The YME command enables the multi-screen mode. If you have two screens connected to your system, the YME command routes GEM-SID to one screen and graphics to the other. This is the default if GEM is not in memory at the time GEM-SID is loaded.

The YMD command enables the multi-screen mode. Both graphics and text are displayed on a single screen. This is the default if GEM is in memory at the time GEM-SID is loaded.

The Y command displays the status of the graphics image save/restore buffer and the current screen mode.

### 5.3.3: The N Command

The N command executes the G (Go) command to transfer control to the program being tested directly before or after the next call or callf. The forms are as follows:

```
-N
N
```

The -N form executes a G command directly before the next call or callf.

The N form executes a G command directly after returning from the procedure called by the next call or callf.

### 3.4: The Q Command (Quit)

The Q command terminates SID-86 if no process is being debugged. If a debug process loaded by the E command is running, the Q command stops the process, but does not terminate SID-86. The form is as follows:

```
Q
```

### 5.3.5: The SR Command (Search)

The SR command searches for a string within memory. The forms are as follows:

```
SRs,f,"string"
SRs,f,value
```

where s is the starting address to begin searching and f is the finishing address to end searching.

The SRs,f,"string" form searches for a string of ASCII characters. The "string" parameter specifies the string of one or more printable ASCII characters you want to search for. Note that you may use either single (') or double (") quotes.

The SRs,f,value form searches for a string of numerical characters. The value parameter specifies the string of nonprintable ASCII characters, numbers, and hexadecimal values you want to search for.

### 5.3.6: The ? Command (Help)

The ? command prints a list of available SID-86 commands. The form is as follows:

?

### 5.3.7: The ?? Command (Help)

The ?? command prints a detailed command list that, in addition to the SID-86 commands, includes the available command options. The form is as follows:

??

### 5.3.8: Using GEMSID with MAP files

In order to use GEMSID with compilers and assemblers that produce Microsoft format MAP files you must first convert those MAP files to SYM files using MAP2SYM. The correct syntax for using that utility follows:

```
MAP2SYM <filename.map> filename.sym
```

## 5.4: The Sample GEM Application and GEM Desk Accessory

```
=====  
DOODLE, a sample GEM application  
=====
```

The sample application, DOODLE, provided with the GEM Programmer's Toolkit serves as an example of way in which you program to the GEM environment. Functionality demonstrated includes:

- opening and closing of windows
- moving of windows
- use of Alert boxes and Dialog boxes
- pop-down menus
- error handling

There are several different BAT files for you to use depending upon which compiler you are using. We have supplied the compile and link options in these BAT files to simplify your understanding of these samples. For more information as to which BATCH file you might want to use see the name and description of

the files that follows this section.

```
=====
MAP2SYM utility
=====
```

If you are using a compiler that outputs Microsoft format MAP files and want to use GEMSID for debugging you must convert those files to SYM files using the MAP2SYM file converting utility provided with this package. The proper syntax for this utility is:

```
MAP2SYM <DOODLEM.MAP> DOODLEM.SYM
```

This conversion is automatically executed in the BATCH files that we provide.

NOTE: YOU WILL FIND A COPY OF MAP2SYM.EXE ON THE TOOLS DISKETTE PROVIDED WITH THE TOOLKIT. IN THE BATCH FILES PROVIDED WITH THE SAMPLE APPLICATION AND ACCESSORY THIS UTILITY IS REFERRED TO AS SYMS. YOU MUST MAKE A COPY OF MAP2SYM CALLED SYMS AS DEMONSTRATED IN THE FOLLOWING COMMAND FOR THOSE BATCH FILES TO WORK CORRECTLY:

```
COPY MAP2SYM.EXE SYMS.EXE
```

```
=====
HELLO, a sample GEM Desk accessory
=====
```

We have also provided a sample accessory called HELLO. There is a compiled copy of HELLO called DESK2.ACC. Placing this file in your GEMSYS sub-directory will cause it to be automatically loaded when you load the DESKTOP.

Since you cannot execute an accessory you must compile it as an application for debugging purposes. In the sample accessory we show you a technique to use for this purpose. On the sample programs disk there are two files, DESKACC0.H and DESKACC1.H, that you use for this purpose. COPY DESKAPP0.H to DESKACC.H and compile to create an application. When you are finished with the development and debugging of your accessory COPY DESKAPPL.H to DESKACC.H and recompile creating an accessory.

There are several BATCH files provided on the disk for compiling and linking of the sample accessory. Carefull examination of these files will demonstrate the correct technique for developemnt of a Desk Accessory.

Note: There are two files on the disk used by the accessory to demonstrate a technique for reducing the amount of code in your accessory, TGEMBIND.C and TVDIBIND.C. These are used in the HELLOMAC.BAT. What has been done in these files is to strip out

the code for the calls that the accessory is not using to reduce overhead. You may also accomplish the same effect by creating a library file of the bindings using a librarian of your choice such as LIB-86.

```
=====
List of files on GEM TOOLKIT MS SOURCE DISK
=====
```

```

ACCSTART.A86      sets up DS and local stack
ACCSTART.ASM      sets up DS and local stack
DESK2.ACC         HELLO.C compiled by HELLOMAC.BAT
DESK2.MAP         Microsoft format symbols of DESK2
DESK2.SYM         converted MAP file using MAP2SYM
DESKACC.H         include used to specify whether app or acc
DESKACCØ.H        include used to specify application
DESKACC1.H        include used to specify accessory
DOODLE.BAT        compiles and links all modules (w/RASM)
DOODLE.C          sample application source code
DOODLE.DEF        usedbyRCS to find names of objects for H file
DOODLE.EXE        executable sample application
DOODLE.H          include file for DOODLE.C generated by the RCS
DOODLE.INP        link input for DOODLE for use with LINK-86
DOODLE.MAP        Microsoft format symbols of DOODLE
DOODLE.RSC        resource file for DOODLE
DOODLE.SYM        converted MAP file, used in debugging
DOODLEL.EXE       executable sample application
DOODLEL.SYM       symbol file for DOODLEL
DOODLEM.BAT       compiles & links DOODLEL & DOODLEM (uses tinyC lib)
DOODNRSC.C        source code for resource file load error message
DOODNRSC.DEF     usedbyRCS to find names of objects for H file
DOODNRSC.H        include file for objects used in load error RSC
DOODNRSC.RSC     compiled resource file containing error message
DOSASM.A86        assembly code portion of DOSBIND
DOSASM.ASM        assembly code portion of DOSBIND
DOSBIND.C         all DOS 2.0 requests go through this module
DOSBIND.H         include module for DOSBIND.C
GEMASM.A86        assembly portion of GEMBIND
GEMASM.ASM        assembly portion of GEMBIND
GEMBIND.C         bindings to GEM AES
GEMBIND.H         include file for GEMBIND
HELLO.BAT         compiles & links HELLO with RASM as an application
HELLO.C           source code for sample accessory
HELLO.EXE         executable HELLO as an application (RASM)
HELLO.INP         input line for linking HELLO (application,w/RASM)
HELLO.SYM         symbol file from HELLO for debugging
HELLOACC.BAT     compiles & links HELLO as an accessory (w/RASM)
HELLOACC.INP     input line for linking HELLO (accessory,w/RASM)
HELLOM.BAT       compiles & links HELLO with MASM as an application
HELLOM.EXE       executable HELLO as an application (MASM)
HELLOM.MAP       MS symbol file
HELLOM.SYM       converted MAP file for debugging
HELLOMAC.BAT     compiles & links HELLO as an accessory (w/MASM)
LOGASM.A86       routines to address data not in your segment

```

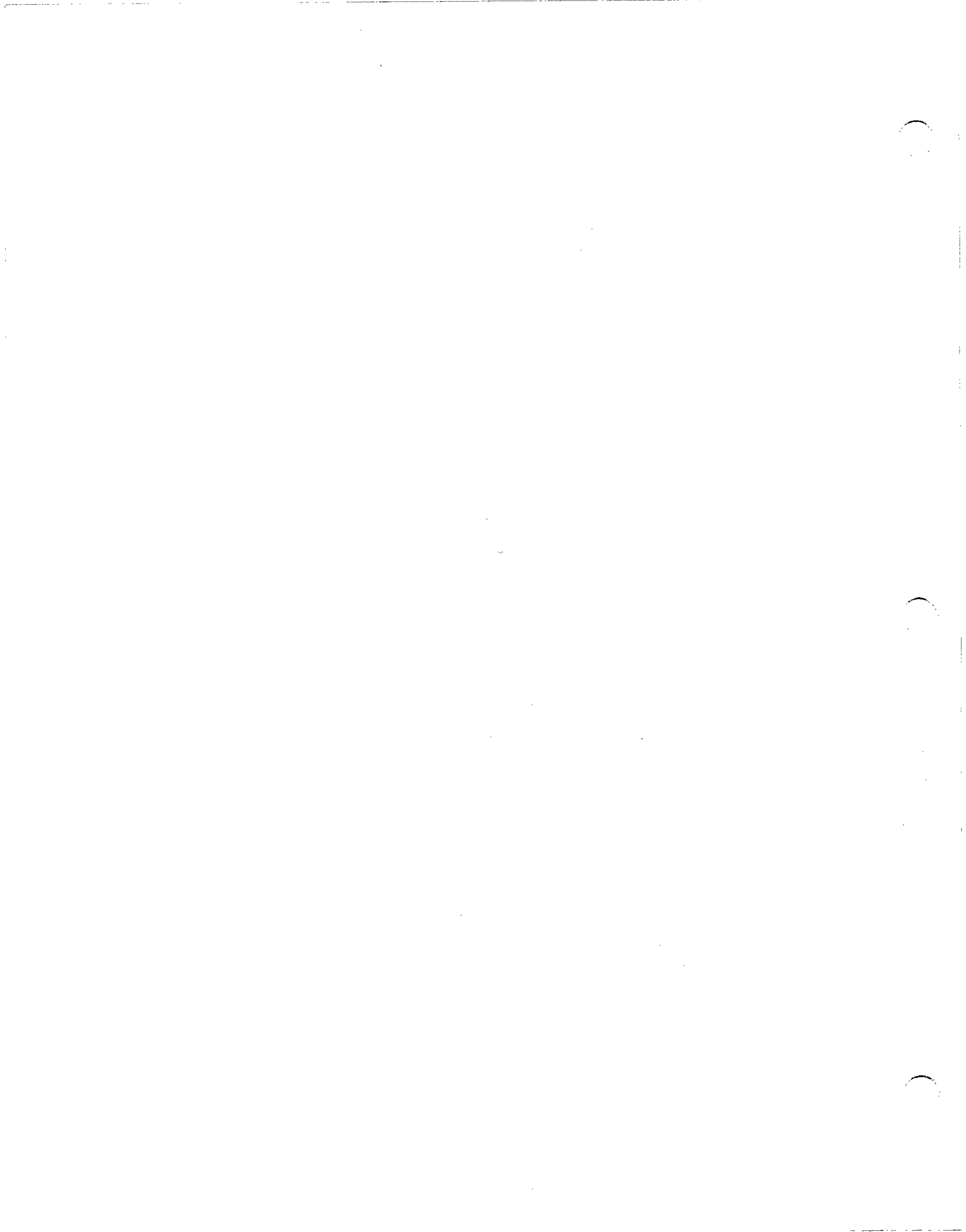
LONGASM.ASM	routines to address data not in your segment
MACHINE.H	processorspecific routines
OBDEFS.H	include file of data structure for objects
PORTAB.H	processor specific definitions
PROEND.A86	marks end of code for PROSTART
PROEND.ASM	marks end of code for PROSTART
PROSTART.A86	shrinks memory, see additional documentation
PROSTART.ASM	shrinks memory, see additional documentation
TCRTL.C	tiny C run time library (STRLEN, STRCAT, STRCPY)
TCRTLASM.A86	assembly portion of TCRTL
TCRTLASM.ASM	assembly portion of TCRTL
TCS.ASM	tiny C start, replaces CS in LATTICE
TGEMBIND.C	tiny GEM BIND
TREEADDR.H	includeforaccessof objecttreesinother segments
TVDIBIND.C	tiny VDI BIND
VDIASM.A86	assembly portion of GEM VDI bindings
VDIASM.ASM	assembly portion of GEM VDI bindings
VDIBIND.C	bindings to GEM VDI
VDIBIND.H	include files for VDIBIND.C

### 5.5: Porting To Other Environments

A program written to GEM System Software is portable to any machine with the same processor that is running GEM without any modification. It is also possible to move an application to an environment with a different processor running GEM with minor modifications.

With our sample application we have included all of the machine and operating system specific calls into several include files. Most of the work in moving an application to another processor requires merely replacing these files with the ones required for your new environment followed by recompiling and linking. An exception to this is that the icons must be remade for each different resolution system.

As GEM becomes available on other systems Digital Research will make available the necessary files for making the changeover. We strongly advise you to write machine independent code or to include all machine dependent code into modules that are simply identified and replaced. We have already moved some of our applications to a new environment with a minimum of effort and expect that you will be able to do likewise.





OUTLINE FOR ISV SEMINAR:  
GEM RESOURCES AND THE RESOURCE CONSTRUCTION SET

1. Demonstration: Creation of Resource File for Demo App (60 minutes)

- Info Dialog
- Pen/Eraser Select Dialog
- Save As Dialog
- Duplicate File Alert
- Out of Windows Alert
- Menu

I. Review of Resource and Object Format

- Comparison to Macintosh
  - A GEM Resource is a file
  - A GEM Resource contains object trees, text strings, and bit images such as mouse forms
- Hierarchy of structures within a resource
  - Tree index
  - Objects
  - Tedinfos, Iconblks, Bitblks
  - Strings
  - Bit image data
  - The "free" indices
    - Free strings
    - Free bitblks
- GEM Objects (GEM AES 6.3)
  - Contain linkage, rectangle def'n, flags, and type specific information
  - Linkage Information
    - next
    - head
    - tail
  - How Objects are organized into trees
    - how siblings are linked
    - how offspring are linked
  - The Visual Hierarchy Rule
    - Each object owns a screen rectangle determined by its x, y, w, h fields
    - All of its offspring must be within that rectangle
    - The offspring DO NOT need to be disjoint
    - Object x,y fields are positive displacements from the parent's x,y coordinates
    - The rule is applied recursively
    - In the case of non-disjoint siblings, the rightmost (last added) will be the last one drawn, and will be one located by objc\_find
  - Object flags
    - Selectable: may be selected by clicking at run time
    - Default: will be selected when return is entered (On a button, causes a triple weight border.)
    - Exit: selection causes the dialog to complete (On a button, causes a double weight border.)
    - Editable: contains editable text
    - Rbutton: is a member of a set of radio buttons
      - Only one of a set of radio buttons is on at a given time
      - All members of a set of radio buttons must be siblings of a common parent object
    - Lastob: set for the last object in a tree (Generated automatically by RCS)
    - Hidetree: makes an object and its progeny invisible. The object library will neither draw nor find

- the object(s).
- Touchexit: the dialog will be exited immediately when the mouse button goes down over this object. (Note: this is different from EXIT, which requires a press+release to activate.)
- Indirect: the ob\_spec field (see below) is actually a long pointer to the real ob\_spec data. (Note: the standard rs\_load and RCS do not support this flag. You must set up the indirect at run-time.)
- Object states
  - Selected: The body of the object is drawn in reverse video. May not be set in the RCS. Used at run time only.
  - Crossed: An X is drawn through the box in system background color. Only useful with boxed objects
  - Checked: A check mark is drawn inside the left margin of the object.
  - Disabled: The object is drawn at half-intensity (gray).
  - Outlined: An outline appears a boxed object. (Note: conflicts with shadowing and outside box borders.)
  - Shadowed: A drop shadow is drawn on the object which is usually a box. Conflicts with outlining.
- Object types and special information (ob\_spec field)
  - Box type objects
    - G\_IBOX - a hollow box
    - G\_BOX - an opaque box
    - G\_BOXCHAR - an opaque box containing a single character
    - The obspec contains packed bit fields defining:
      - Background color
      - Background dither pattern
      - Border color
      - Border thickness
      - Character
      - Character writing mode
      - Character color
  - Raw text objects
    - Are always written in mono-spaced full-size system font, using system foreground and background colors in replace mode.
    - G\_STRING - text string
    - G\_BUTTON - text string enclosed by a box
    - G\_TITLE - text string (used in menu bar only)
    - The ob\_spec is a long pointer to a zero terminated ASCII string.
  - Formatted text types
    - used for editable text
    - also used for text which should not appear in the default system font.
    - G\_TEXT - formatted text
    - G\_BOXTEXT - formatted text with a box
    - G\_FTEXT - editable text
    - G\_FBOXTEXT - editable text with a box
    - The ob\_spec field is a long pointer to a TEDINFO structure.
    - The TEDINFO contains long pointers to text, template and validation strings.

(Note: the latter two are only used in editable fields.)

- TEDINFO also includes the same info as in the box-types, as well as font and justification designators.
- Bit image type (G\_IMAGE)
  - A non-masked (replace mode) rectangular bit image.
  - The ob\_spec is a long pointer to a BITBLK structure.
  - The BITBLK contains a long pointer to the actual data, plus horizontal and vertical size fields, and the writing color.
- Icon type (G\_ICON)
  - A data+mask bit image, with optional associated text string and character.
  - The ob\_spec is a long pointer to an ICONBLK structure.
  - The ICONBLK contains long pointers to the data and mask bit images, and to the text string.
  - It also contains a X and Y sizes, foreground and background colors, the (optional) character, and relative positions of the text and character.
- User defined type (G\_PROGDEF)
  - A type which is handled at run-time by application specific code.  
Not supported by RCS or rs\_load:  
set up at run-time
  - See GEM AES 6.3.5 - 6.3.6

## II. Outline of the resource editing system.

- The Resource Construction Set
  - .RSC files - input/output - binary resource
  - .DEF files - input/output - RCS use only, contain tree/object names and types.
- Binding to languages/applications
  - Associate user supplied tree/object names with actual locations in the resource.
  - .H file is produced for 'C'
  - .I file is (optionally) produced for PASCAL
- Loading bit images
  - Bit images are defined/edited in the ICON EDITOR
  - .ICN files contain an ASCII representation of the binary image.
  - .ICN files are loaded into the RCS using the LOAD menu option applied to a G\_IMAGE or G\_ICON
- The secondary update cycle
  - Use to create specialized objects or include "free" strings or images
  - Create optional .C file from RCS - contains a C language source of the resource.
  - Hand edit this file
  - #include the file in RSCREATE.C and compile, link, and execute RSCREATE
  - A regular .RSC will be produced: resume the normal editing cycle

## . Overview of RCS operation

- Program states
  - Nofile state: initial state: no file associations

- view and parts window contain tree icons
- File state: file under edit; view and parts window contain tree icons
- Dialog state: editing a dialog type tree. View and parts windows contain GEM drawing objects
- Free state: editing a "free", unformatted. View and parts windows contain drawing objects.
- Menu state: editing a menu tree. View and parts contain menu objects. Special formatting rules apply.
- Alert state: editing an alert tree. View and parts contain alert objects. Special formatting rules apply.  
(Note: an alert is stored as a tree only for the duration of the edit - it is stored in the resource as a free string.)
- Box, String, Formatted text, image, and icon editing states. Reached by opening an object of the appropriate type. These are not "full" states, but rather editing dialogs.
- The Screen Layout
  - The Parts Box Window: contains prototype trees or objects (an infinite supply)
  - The View Window: contains a picture of the resource or tree under edit.
  - The Trashcan: is the destination for deleting parts from the view.
  - The Clipboard: a temporary save area for transferring objects between trees. (Note: the clipboard is cleared when files are saved/opened; it is local to the RCS, and is one deep.)
  - The Files Menu: contains file related operations. OPEN and CLOSE may also be used with selected trees and objects.
  - The Options Menu: contains advanced editing commands which may be used after selecting a tree or object.
  - The Global Menu: for selections which affect the overall operation of the program.
- Mouse Operations
  - Dragging == Move
    - Part to View: place new part
    - View to View: move part
    - View to Clipboard: "cut" part to clipboard
    - View to Trashcan: delete part
    - Clipboard to View: "paste" part into view, clearing clipboard.
  - Shift-drag == Copy
    - Part to View: place new part
    - View to View: duplicate part
    - View to Clipboard: copy part to clipboard
    - Clipboard to View: "paste" new part into view, clipboard is not cleared.
  - Click + menu == perform operation on selected part
  - Double-click == open part for edit
  - Operations peculiar to tree edit states:
    - Drag object size handle == resize object in view window.
    - Control key modifier == select the parent of the object pointed at, then perform the normal operation.
    - All copy/move operations apply to the selected object AND its progeny.

## Walkthrough of the RCS

- Desktop operations
  - Moving the windows. You may move the partbox and view window by grabbing their title bars and dragging.
  - Sizing the windows. Size either window by dragging the sizebox at the lower right. The part box contents will be rearranged to fit the new size. Tree icons in a "whole-resource" view will also be rearranged.
  - Moving desktop objects. You may move the trashcan and clipboard by dragging them to a new location on an EXPOSED piece of the desktop.
- Global options
  - Safety selection
    - Locked: Object editing and sizing only. Intended for "post-release" changes, such as internationalization. This mode preserves the values associated with tree and object names, so that the application need not be recompiled.
    - Normal: The default. All operations are legal. Warnings are given for hierarchy changes and workspace clears.
    - Expert: Anything goes. No warnings.
  - Output options
    - .H: generates a #define file to be included in the compile of the application. It contains the numeric values of the names given to trees and objects.
    - .I: the same thing, but formatted for use with PASCAL
    - .C: generates a C source of the resource which may be hand-modified and regenerated using RSCREATE.
- Operations in NOFILE and FILE states (the "whole resource" views)
  - File operations
    - OPEN: brings in an existing resource, replacing whatever is in the workspace. If the workspace is not empty, you are warned. FILE state is entered.
    - NEW: clears the workspace, releases any file association, returning to NOFILE state. You are given a chance to cancel.
    - MERGE: adds an existing resource file to the current workspace. If name conflicts are detected, RCS invents a unique name.
    - CLOSE: saves the current file to disk, and clears the workspace, returning to NOFILE state.
    - SAVE: writes the current file to disk and continues the edit. (A memory compress is done during the write.)
    - SAVE AS: writes the workspace to a new file. If no file name currently exists, the SAVE AS file becomes the default, and FILE state is entered.
    - ABANDON: reload the last saved version of the current file. Useful if you if you make a mistake.
    - QUIT: exit the RCS immediately, without saving anything. You are given a chance to cancel.
    - INFO: gives the total number of trees, objects, and other structures in the workspace. Also gives a byte total and bytes remaining in the workspace.

operations on trees (represented by icons)

- Add a tree: Drag the icon for the appropriate tree type from the partbox and drop it in the view window. A NAME operation is automatically executed.
- Reorder trees: Drag the tree you want to move to its new position and drop it. To move to end of the list, you may drop it on any open area below the array of icons.
- Delete a tree: Drag the tree to the trashcan.
- Duplicate a tree: Shift-drag the tree to the place you want the new one inserted. A NAME operation is automatically executed. (NOTE: All naming information is lost during a duplicate.)
- "Cut" a tree to the clipboard. Drag the tree to the clipboard. Anything which is cut or copied to the clipboard loses its name information. (Note: since the clipboard does not survive file load/saves, its main use with trees is holding a prototype tree to be copied many times.)
- Copy a tree into the clipboard. Shift-drag the tree to the clipboard.
- Paste clipboard contents into window, clearing clipboard. Drag from clipboard to view, dropping at the insertion point. (A NAME operation is automatically executed.)
- Paste from clipboard without clearing clipboard. Shift-drag from clipboard to insertion point. A NAME operation is executed.
- INFO: Click on the tree, and select INFO. Gives statistics for the tree only, plus bytes remaining in workspace.
- NAME: Click on tree and select NAME. You are allowed to change the tree name and/or the type of the tree. Be extremely careful when changing a tree from a less to more restricted type, e.g., FREE to MENU.
- OPEN: Double-click on the tree, or click on the tree and select OPEN. The RCS enters the appropriate editing state depending on the type of tree opened.
- Operations in the tree editing states: FREE, DIALOG, MENU, and ALERT
  - The tree editing states differ in the rules applied as the tree is edited, and in the operations which may be performed. An appropriate partbox is provided for each state.
  - FREE state is the most general, the only rule enforced is the visual hierarchy.
  - In DIALOG state a character snap grid is enforced. This makes the resulting trees portable amongst machines.
  - A MENU tree is constrained to have a parallel structure of title bar entries and pulldowns, as well as a critical sizing of the bar objects. The RCS enforces these constraints.
  - An ALERT tree is an analogue of a free string which will be interpreted by form\_alert at application runtime. The rules enforced are those used by form\_alert. An alert will contain at most one image out of a set of three, 5 message lines of 40 characters max, and three buttons. The size of the alert and the object position are automatically readjusted as editing proceeds.

- another TITLE will cause the pull-down to snap up.
- When you place a new TITLE, a blank pull-down is created.
  - If you delete an entire pull-down, a new blank one is created. Deleting the TITLE will also remove the pull-down.
  - The DESK menu is required. The only operation you may perform on it is to edit the first line, which is traditionally the program information entry.
  - Remember that the control key is useful for selecting pull-downs which are completely covered with entries.
  - The disabled dashed line is used as a separator between logical groups of menu entries.
  - The box object is provided as a place-holder for user-defined objects. For an example, see GEM DRAW.
- Alert editing rules
- Alerts are actually emitted as strings. The editing rules applied reflect those in the form\_alert run-time.
  - You may place an object in an alert by dropping it anywhere within the alert. It will snap to the appropriate position.
  - Only one icon may be present at a time. It must be one of those from the partbox. A new icon replaces an old one. You may not edit or load the icon. You may delete the icon totally by dragging it to the trash.
  - No more than five strings or three buttons are allowed. The size of the alert automatically adjusts as strings/buttons are added, deleted, or edited.
- Editing objects
- Object editing is accomplished within dialog boxes. There are five different dialog which appear depending on the type of object opened: Raw text type (string, button, or menu title), box type (box, hollow box, and boxchar), formatted text (text, boxtext, ftext, and fboxtext), bit image, or icon.
  - Autosizing. The rectangle size of certain objects will be altered to reflect the length of their text or the size of their bit images. In general, the types affected are those lacking a box as part of their definition: strings, menu titles, text and ftext, and icons and images. The new size of a text type is determined by the length of the PTEXT field, that of an ftext by the length of its PTMPLT. New bit object sizes are generated at LOAD time. You can always change the automatic size by dragging the size handle.
  - Raw text edit dialog. The options presented are the flags and states as discussed in the summary of object format, as well as the ability to enter/edit a character string up to 40 characters long.
  - Box edit dialog. The standard flags and states are presented. You may also select the background color and dither pattern, the border thickness and color, and the character color, masking rule, and value. Most of these attributes are selected with scrolling selectors. To make a choice, click on one of the visible options and it will appear in the Selected box. To see more choices, scroll the selector by clicking on the arrow boxes. Some of these box

of this type: background information is not relevant for hollow boxes, and character data is only useful for boxchars. You can go ahead and enter this data, but it will only be used if you change the object type. Note that choosing an inside border thickness on a BOXCHAR which is one character high will prevent the character from being drawn.

- Bit image edit dialog. This dialog includes the standard flags and states, as well as a color selector for the bit image. Note: the image data is loaded via the LOAD menu option.
- Icon edit dialog. In addition to the standard flags and states, you may select the foreground and background colors, specify an associated text string and/or character, and select their position. To select position, click the box in the location where you would like the text or character to appear. Remember that the icon's data and mask images are read in with the LOAD menu option.
- Formatted text dialog.  
This is the most complicated dialog. It includes the standard states and flags, and all of the selectors which appear in the box dialog. (The latter are used with G\_BOXTEXT and G\_FBOXTEXT. Again, note that choice of an inside border with a single character high box will prevent the characters from being drawn.) In addition, you can select the font and justification of the text, and enter the PTEXT, PTMPLT, and PVALID fields. If the object type is G\_TEXT or G\_BOXTEXT, then only the PTEXT field is used. If the type is G\_FTEXT or G\_FBOXTEXT, then all three must be specified. The PTMPLT (template) entry establishes the fixed and variable portions of the field. Use a tilde to represent an editable character position, anything else is regarded as fixed. (The tilde is used to avoid visual confusion with the underscores in the dialog - tildes are replaced with underscores in the actual TEDINFO.) Under every tilde you must enter a validation character in PVALID. These characters are defined in the GEM AES 6.3.2. If you want a default character to appear enter it in the corresponding position in PTEXT. If you don't want a default, enter blanks or a null string. CAUTION: if you use a null string, you MUST allocate and link a scratch PTEXT before activating the dialog at run-time. Failure to do this will crash the resource.



## VI. RCS Caveats, Tradecraft, and Folklore

- Capacity limits on the RCS
  - 60 trees (you can't see more than 55 on an IBM PC, but they are there, just out of your viewing area)
  - 1250 objects
  - 30 Kbytes (decimal) total resource size
  - 500 names for objects and trees (these DO NOT count against the size limit - they are stored in a separate index.)
- Things the RCS will not do
  - Handle free strings other than alerts
  - Handle free bit images
  - Handle user-defined objects or indirects
  - Let you scroll the tree icons or partbox
- Known bugs in Beta 1.0
  - Icons and bit images are currently in IBM system dependent format. Save your .ICN files for reloading when the final RCS version is released.
  - RCS will not allow you to open an object with a non-zero upper byte in the type field.
  - There is no ctrl-Z at the end of output text files.
  - The system performs extra redraws.
  - Entering an empty string or cancelling an image/icon load will leave the object with a zero width/height.
  - Dropping a menu entry on the dividing line between the bar and the pull-down may cause a hang.
- Respect for tree typing
  - Be sure to mark non-standard trees with the ? type. A tree is non-standard if it contains user defined objects, unresolved indirects, or non-standard linkage.
  - You may freely retype trees from a restricted type such as alert or menu, to a less restricted type such as free or dialog. DO NOT retype in the other direction unless you clearly understand the required format. You can crash the program this way!
  - If you have loaded an existing resource which lacks a .DEF file, try typing each of the trees as a FREE or DIALOG and examining them.
  - Use caution when pasting objects from other trees into menus or alerts. Most incompatible objects will be rejected by the RCS, but you may be the lucky one to find an exception!
- Memory, and the lack thereof
  - RCS uses memory from its workspace as you edit. There is no garbage collector, you recover space by doing a SAVE which writes out and reloads the resource. (This is a good practice anyway.)
  - It is easy to create a large resource without noticing. Use INFO to check on what you are doing.
  - Remember that the resource uses a separate segment, but counts against your 128K total if you are creating a minimum memory application.
  - The byte total given by INFO is the working total and will be somewhat different from the actual output. Alerts will take up less space when emitted as strings. Also, the RCS does not duplicate strings and images from shift-dragged objects until required. These are resolved

at save time and may add to the size of the resource.

- If you must create a larger resource, generate it in two parts, and combine them by hand using RSCREATE. There is an ABSOLUTE limit of 64K in a resource. You may "chain" resources but only after clearing any references to the old one in the AES, e.g., menu and desktop definitions.
- Using RSCREATE: the secondary update process
  - You may hand-edit a .C file produced by RCS and regenerate the resource by #including it in RSCREATE.C and compiling, linking, and executing RSCREATE.
  - The .C file contains some mnemonics for object types, etc, as well as base numbers for each object tree. When adding any new object, TEDINFO, etc. insert them at the END of the current entries. Update the tree base definitions if you insert new objects into existing trees.
  - You MUST use the secondary update process to insert free strings (other than alerts) and free bitblks. Do this early in the resource creation process to keep down the amount of editing involved.
  - If you know you will alter the object structure by hand, do not enter object names until you are done. Changes in object numbers caused by RSCREATE will NOT be adjusted for by the RCS, and your .DEF file may have to be re-entered.
  - RCS always emits the objects in a tree in pre-order: root first, followed by its children left to right, with the rule applied recursively. Any hand-built trees which are passed through the RCS will have this done: do not write code which is dependent on other ordering.
  - Be aware that some C compilers (such as Lattice) will fold duplicate strings together when you compile RSCREATE. This may be hazardous at run-time if the strings are to be altered by editing. Run the resource through the RCS to resolve the duplicate strings.
- Saving time
  - When you are laying out a dialog, put related text items into a box. You can then move and center them as a unit. When you are satisfied, FLATTEN the box to save memory.
  - Whenever possible, drag one of each part type you will need into the view and then use shift-drag to peel off copies. This saves the aggravation of topping the parts window each time you need a new object.
  - If you have several similar objects (such as radio buttons) within a tree, define one of them and make copies, editing only what differs.
  - Often entire subtrees are nearly duplicated within a tree or among a set of trees. Create a prototype and copy it using shift-drag and/or the clipboard.
  - For commonly used icons, subtrees and the like, create a "library" resource file (or files). Use MERGE to copy a library into your working resource, copy the parts you want using the clipboard, then delete the unneeded trees. You can use

- appearance of your applications if you include prototype dialogs of your own design. (BTW, the merge/delete process uses up workspace, SAVE your work to get it back!)
- Take advantage of the quirks of the MENU and ALERT views. To make more menu bar entries, point at one which exists, hold down the shift key and do a "drag-in-place" by holding down the mouse button. Watch the copies zip down the bar! You can do the same thing with message strings and buttons in the alert view!
  - May we suggest? (Some hints on formatting)
    - In the resource file
      - Put your menu first
      - Group dialogs next, then any free trees
      - Put "non-standard" trees last, so they will be easy to edit manually.
      - Establish tree and object naming conventions. Design them to avoid conflicts. Occasionally print out and check your .H file for errors.
    - In menus
      - Make sure the entries totally cover each pull down
      - Use the "disabled dashes" entry to separate disparate functions, and to set off hazardous ones.
      - Use two leading blanks on menu entries, and follow them with dots if they lead to a dialog. Leave at least one trailing blank as a margin.
      - Sort each pull-down by Y when you are done. It looks rather funny when they are drawn in random order!
      - But a leading and trailing blank on each title bar entry.
      - Leave about 30% of the menu bar clear. You will need the space if you internationalize your program.
      - Avoid lengthy and cluttered menus and menu bars. Remember that the maximum pull-down size is one quarter of the total screen area. The beta version of RCS does not enforce this limit!
      - Remember to NAME the menu titles as well as the entries.
    - In dialogs
      - Use the standard frame - a two-in border plus an outline.
      - SORT your radio-buttons, and SORT the whole dialog by Y major, X minor when you are done. It will then draw in a smooth, natural looking order.
      - Place exit buttons near the right edge. Standardize their appearance and placement by creating a your own prototype dialog. Put it in a library resource.
      - Make selectable objects LARGE so they are easy to point at. Embed text IN the object in preference to placing it nearby. Avoid packing selectable objects together. Place dangerous options and exit buttons away from other objects.
      - Choose your defaults carefully. The same choice

pick the exit option that is the safest is your default.

- Alerts
  - Keep them simple. Avoid jargon. Avoid three button alerts.
  - Use them in single button form for error messages. But, see if you can avoid the need for some alerts by disabling or hiding inappropriate menu entries, dialog selections, and so on.
  - Adopt consistent standards for use of the three severity levels of alert.
- Cheating on visual hierarchy
  - is not recommended, but can be useful for visual effects or in creating sophisticated selector sub-trees.
  - to cheat, be sure that the object you want to lie "on top" was inserted after the "bottom" object.
  - size the "top" object so it cannot fit totally into the bottom object.
  - move the top object, placing its upper left corner where you want it inside the bottom object.
  - size the top object down to its desired size.
  - if you later want to adjust the position of the top object, you must first size it back up.
  - NEVER turn off the move alert when working on trees of this sort.
- Hints on extending form\_do
  - You may do this to create complex selector objects such as the scrolling selectors in the RCS.
  - Use TOUCHEXIT to return control from form\_do as soon as the user clicks on your special objects. Write a shell routine which processes the special object, does any necessary redrws and re-enters form\_do.
  - Use the user defined object type to draw non-standard objects.
  - Use the HIDE TREE flag to make variable parts of the dialog appear and disappear. (Use this sparingly - it encourages cumbersome dialogs and ill-structured driver code.)
  - Use the upper byte of the object type word to create your own extended object types. GEM AES ignores this byte; RCS ignores and preserves it. (You will have to do this with RSCREATE - once you have created such an object put it into your library so you can make copies easily! The brave may attempt this modification using a debugger directly on the resource - use a .C file to figure out the internal structure and make a backup before proceeding!)
  - Use the INDIRECT option to set up pointers to data areas.
  - Be careful to restore the resource to a known state at the end of your routine.
  - Leverage your work by putting your code and the most general form of the associated subtree into libraries.

END

- Object part box contents
  - BUTTON
  - STRING (unformatted)
  - FTEXT (editable, formatted, shown as EDIT: \_\_\_\_\_)
  - FBOXTTEXT (editable, formatted, shown as EDIT: \_\_\_\_\_ enclosed in a box)
  - IBOX (hollow box, shown schematically as a box with an outline)
  - BOX
  - TEXT (formatted)
  - BOXCHAR (a 'C' in a box)
  - BOXTTEXT (formatted)
  - ICON (labelled)
  - IMAGE (labelled DRI logo)
- Operations on the whole tree
  - INFO. If this option is used when no individual object is selected, statistics for the whole tree are given.
  - CLOSE. To finish editing the tree, select CLOSE or click the close box of the view window.
- Operations on objects
  - These operations are described in the most general case: FREE and DIALOG. Restrictions in the MENU and ALERT states are noted below.
  - Add a new object. Drag the prototype object from the partbox. It is placed at the appropriate visual hierarchy level.
  - Deleting an object. Drag it from the view to the trashcan. All progeny of the object are also deleted.
  - Moving an object. Place the mouse pointer near the center of the object and drag it to its new location. All of its progeny will also be moved. If the move will result in a change in visual heirarchy, you will be warned.
  - Sizing an object. Each object has an invisible sizing handle inside its lower right hand corner. Drag this handle to change the objects size. The drag operation is clamped to prevent you from violating hierarchy rules. (See also: auto-sizing under object editing below.)
  - Duplicating an object. Shift-drag from the object to be duplicated to the desired insertion point. All of its progeny will also be duplicated.
  - "Cutting" an object to the clipboard. Drag the object from the view to the clipboard. In all clipboard operations, progeny are also affected, and naming information is lost.
  - Copying an object to the clipboard. Shift-drag the object to the clipboard.
  - Pasting an object from the clipboard to the view. Drag from the clipboard to the view. The drag box will assume the size of the object being dragged to aid in positioning. The clipboard is cleared by this operation.
  - Paste an object from the clipboard, without clearing the clipboard. Shift-drag from the clipboard to the view.
  - Editing an object. Double-click on the object, or click on it, and select OPEN. You will

which will reflect the type of the object under edit.

- INFO: Click on an object and select INFO. You will be given the statistics for this object alone.
- NAME: Click the object and select NAME. You may assign the object a name and/or change its type amongst compatible types. To delete an object's name, enter an empty name. There are three sets of compatible object types: boxes (G\_BOX, G\_IBOX, G\_BOXCHAR), raw text (G\_STRING, G\_BUTTON), and formatted text (G\_TEXT, G\_FTEXT, G\_BOXTEXT, G\_FBOXTEXT). No other types are compatible with bit images or icons.
- HIDE: Click the object and select HIDE. This sets the HIDETREE bit for the object, causing it and its progeny to disappear.
- UNHIDE: Click an object and select UNHIDE. Any children which have been hidden will reappear. This option only appears if the object has offspring.
- SORT: Click an object and select SORT. This reorders the progeny of the object based on their location on the screen. A dialog will appear with options for X, Y, or combined sorting criteria. This option only appears if an object has offspring.
- FLATTEN: Click an object and select FLATTEN. This deletes the object and raises all of its offspring up one level, preserving their visual position. This option only appears if the selected object is a box type and has offspring.
- SNAP: Click the object and select SNAP. The objects X, Y, W, and H parameters are rounded to the next even character values. This option only appears in FREE state.
- LOAD: Click on an icon or image and select LOAD. This option is used to import bit images (.ICN files) from ICONEDIT. If the selected object was an icon, a dialog will appear allowing you the choice of loading the data, the mask, or both. If the new bit image(s) are of different dimension from the old, auto-sizing will be applied (see below).
- The control key modifier. You may cause any operation to refer to the parent of the object under the pointer by pressing the control key when the operation begins. For example, pointing at a radio button and doing a control-drag would move the entire set of radio buttons.

- Menu editing rules

- Only TITLE objects may be placed in the menu bar.
- Whenever a TITLE is moved, inserted, deleted or sized the menu bar and the corresponding pull-downs are rearranged.
- To see a pull-down click on its TITLE entry. You may now select entries within the pull-down.