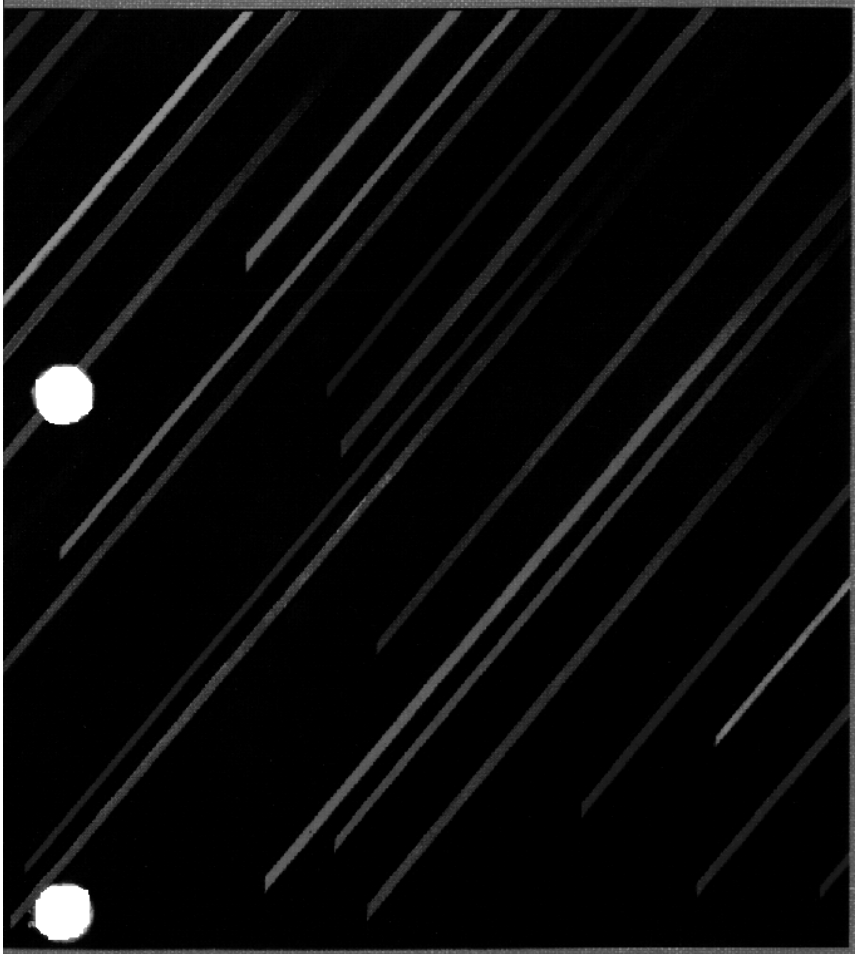
Multi-Vue



TANDY®

TERMS AND CONDITIONS OF SALE AND LICENSE OF TANDY COMPUTER EQUIPMENT AND SOFTWARE PURCHASED FROM RADIO SHACK COMPANY-OWNED COMPUTER CENTERS, RETAIL STORES AND RADIO SHACK FRANCHISEES OR DEALERS AT THEIR AUTHORIZED LOCATIONS

USA LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

A CUSTOMER assumes full responsibility that this computer hardware purchased (the "Equipment"), and any copies of software included with the Equipment or licensed separately (the "Software") meets the specifications capacity capabilities versatility, and other requirements of CUSTOMER

B CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation

II. LIMITED WARRANTIES AND CONDITIONS OF SALE

A For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects **This warranty is only applicable to purchases of Tandy Equipment by the original customer from Radio Shack company-owned computer centers, retail stores, and Radio Shack franchisees and dealers at their authorized locations** The warranty is void if the Equipment or Software has been subjected to improper or abnormal use If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center a Radio Shack retail store, a participating Radio Shack franchisee or a participating Radio Shack dealer for repair, along with a copy of the sales document or lease agreement The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair replacement or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items

B RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software except as provided in this paragraph Software is licensed on an "AS IS" basis, without warranty The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store a participating Radio Shack franchisee or Radio Shack dealer along with the sales document

C Except as provided herein no employee, agent franchisee dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK

D **EXCEPT AS PROVIDED HEREIN, RADIO SHACK MAKES NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.**

E Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER

III. LIMITATION OF LIABILITY

A. **EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE." IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE." NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.**

B RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software

C No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software whichever first occurs

D Some states do not allow the limitation or exclusion of incidental or consequential damages so the above limitation(s) or exclusion(s) may not apply to CUSTOMER

IV. SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the TANDY Software on **one** computer, subject to the following provisions:

A Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software

B Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER but not title to the Software.

C CUSTOMER may use Software on a multiuser or network system only if either, the Software is expressly labeled to be for use on a multiuser or network system, or one copy of this software is purchased for each node or terminal on which Software is to be used simultaneously

D. CUSTOMER shall not use make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software

E CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use

F. CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER

G All copyright notices shall be retained on all copies of the Software

V. APPLICABILITY OF WARRANTY

A The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby Radio Shack sells or conveys such Equipment to a third party for lease to CUSTOMER

B. The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK the author, owner and or licensor of the Software and any manufacturer of the Equipment sold by Radio Shack

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state

Multi-Vue

For

OS-9 Level Two

10 9 8 7 6 5 4 3 2 1

Contents

Part 1
The Multi-Vue System

The Control Panel/ 3

The Calendar/ 4

The Clock/ 5

Part 2
Programming Information

Part 1
The Multi-Vue System

Getting Started

Chapter 1

Multi-Vue is a *working environment* that uses the Color Computer 3 and OS-9 Level Two. With the keyboard, a joystick, or a mouse, you can make selections through the environment's *pull down* menus. Your selections include special Multi-Vue functions, selected OS-9 commands and utilities, and application programs.

Multi-Vue Windowing

As you pull down menus and make selections, Multi-Vue opens *windows* that it uses to execute the programs you choose. By opening several windows, several programs can share your computer's facilities and even share the same screen at the same time. With the press of a key, you can cycle through programs and you won't lose your place or terminate an operation.

Getting Started

To use Multi-Vue, you need:

- A Color Computer 3
- OS-9 Level Two Operating System
- One or more disk drives
- A TV or monitor

Also, you must prepare your software before you can use it. The rest of this section will carefully lead you through the process of making bootable copies of Multi-Vue. By carefully reading this chapter first, you can make this process easier. You might also find your OS-9 reference materials to be very helpful.

Make a Backup of Multi-Vue

Before you begin using Multi-Vue, make at least one set of backup copies of the Multi-Vue diskette. After making the backups, use them as your working disks. Don't use the original diskette unless your working copies are damaged. Then, use the original only to make other backups. The cost and time involved in making backups is small compared to the inconvenience of accidentally destroying your program.

To make copies of the Multi-Vue diskette, select two new diskettes, or ones that contain data you no longer want to keep. Format the diskettes you selected:

1. Turn on your computer and insert OS-9 Level Two Disk 1 into Drive 0.

2. Type **DOS** [ENTER]. If you don't have Disk Extended Color BASIC Version 2.1 or higher, see Appendix A.

3. At the OS-9 prompt, enter the time and date in the format shown or push [ENTER].

4. At the OS-9 prompt, type **format /d0** [ENTER]. Remove the OS-9 diskette and put in a blank diskette. Follow the prompts.

5. Remove the formatted diskette from Drive 0, reinsert the OS-9 Level Two Diskette and repeat the formatting process.

Now you can copy the contents of the Multi-Vue diskette onto the newly-formatted diskettes. To do this:

1. Reinsert the OS-9 diskette back into Drive 0.

2. Type **backup** [ENTER], and insert Multi-Vue Side A into Drive 0. Follow the prompts. (You will be using Multi-Vue's Side A to make Diskette 1).

3. Remove the copy of Side A from Drive 0, and reinsert OS-9 Level Two.

4. Repeat the backup process using Side B. *Be sure to flip the Multi-Vue diskette when you copy side B.*

5. When the process is finished, you will have two working-copies for Multi-Vue. Label the copy of Side A as Diskette 1, and the copy of Side B as Diskette 2.

For more detailed directions, Chapter 3 of *Getting Started With OS-9* describes how to format diskettes and how to make copies using either one or two disk drives.

If You Have 128K

If your computer has 128 kilobytes of memory, you can keep only one window open at a time. When you finish running an application, GShell clears its main screen, loads the new program, and draws a new window. GShell will need to clear the screen for every application you run.

Because of the limited amount of memory space, you must also use a low resolution display (16K). There is not enough free RAM to maintain a high resolution screen. See the OS-9 windowing manual for screen types and RAM recquirements.

If You Have 512K

If your computer has 512 kilobytes of memory, GShell creates concurrent windows if you want to run more than one application. Since 512K allows for a true multi-tasking environment, it dosen't need to reload the application and redraw the screen everytime you quit one application and select another. Once a screen is created, it is kept, and you can cycle from window to window and from application to application by pressing [CLEAR].

With 512K, you can use the high resolution mode for the screen.

The limit to the number of windows and processes you can have is the number of windows and the amount of physical memory that exists in your operating system. (Each GShell will allow the creation of 8 to 13 windows, depending on the operations being performed.)

GShell needs at least one background window for each full screen of active windows. If you run an application that automatically requires a full screen, such as the calendar, only one window will be required.

However, if you run an application that allows you to size a window, such as the calculator, there will be two windows: one for the calculator's image and the other for the background seen behind it. Additional applications that require sizing can be placed on that same background window until the window becomes full or memory is exceeded.

Do This If You Have 512K:

Your system will automatically default to 128K, even if you have a 512K system. The RAM size is set for 128K in the environment file of the Multi-Vue software, so you must adjust the parameter by entering the following sequence:

1. Boot your OS-9 Level 2 system diskette
2. Type **load edit** [ENTER]
3. Insert Multi-Vue diskette 1 in drive 0.
4. Type **edit /do/sys/env.file** [ENTER]
5. Type **+3** [ENTER]
6. Type **c/R/*R/** [ENTER]
7. Press the [ENTER] key
8. Type **c/*R/R/** [ENTER]
9. Type **Q** [ENTER]
10. Type **unlink edit** [ENTER]

Your Multi-Vue diskettes cannot be booted until you make a bootable version on the *working copy* of Diskette 2. When you make Diskette 2 bootable, it retrieves the operating system modules that it needs from the computer's memory. After you do this the first time, you won't have to do it again. You will be able to start your computer without the OS-9 Level 2 system diskette.

First, be sure that you're using the original system diskette, not one that's been customized.

1. Boot your computer with your OS-9 system diskette, and wait for the OS-9 prompt.

2. Put Multi-Vue Disk Two in Drive 0.

3. Type: **chd /d0** [ENTER]

4. Type: **chx d0/cmds** [ENTER]

5. At the "OS9:" prompt, type:
   BuildMV [ENTER]

6. Wait for the process to complete itself. You will be prompted to change diskettes.

You now have a bootable version of Multi-Vue. This diskette will boot your system, and it will load part of the Multi-Vue environment. The remainder of Multi-Vue will be loaded when you insert Diskette 1 in place of Diskette 2.

Place Multi-Vue's Diskette 2 in Drive 0. (If you don't have Disk Extended BASIC Version 2.1 or higher, see Appendix A.) Type:

DOS [ENTER]

After a short pause, the screen displays the OS-9 copyright and sign-on message:

```
OS-9 LEVEL TWO VR. 02.00.01
     COPYRIGHT 1986 BY
MICROWARE SYSTEMS CORP.
LICENSED TO TANDY CORP.
   All Rights Reserved
* Welcome to Multi-Vue     *
*  for OS-9 Level Two      *
* on the Color Computer 3 *

         yy/mm/dd  hh:mm:ss

 Time?
```

Setting the Date and Time

A prompt asks you for the date and time in the format:

yy/mm/dd hh:mm:ss

Type the current date and time. For example, to enter May 28, 1987 and 2:30 pm:

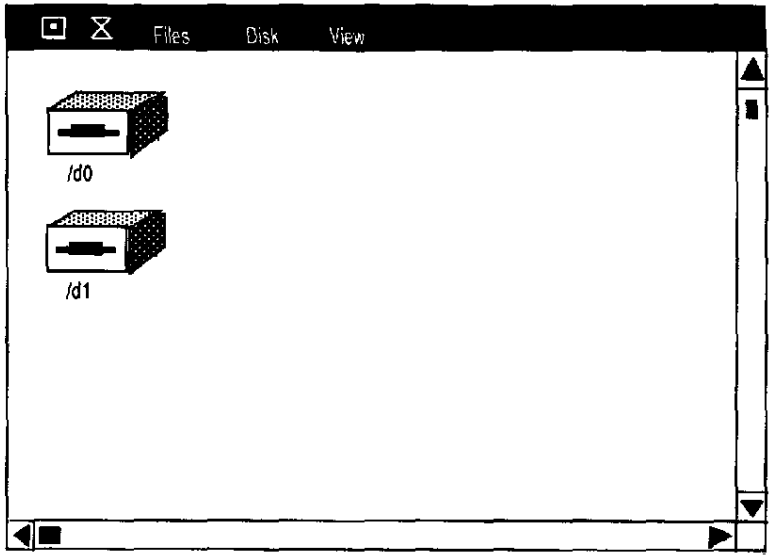
87/05/28 14:30:00

You can use spaces, commas, or other punctuation marks in place of the slashes and colons.

To completely skip this step, press **[ENTER]** when you see the prompt. By doing this, you choose to indicate a date and time if you need to later. However, if you don't enter the date and time, OS-9 can not accurately date files it creates, nor can you use all of the Multi-Vue Calendar, Time, and Alarm functions.

A few moments after you press **[ENTER]**, you see the copyright screen. Follow the prompt by inserting Diskette 1 (the one that isn't bootable) into Drive 0.

Next, press any key. In a few moments you will enter GShell:



This is the main screen. In the next chapter you will learn how to access all of the GShell functions.

GShell

The Graphics Shell

OS-9 uses an interpreter, OS-9 Shell, that converts your word commands into codes that your computer can understand. But what about Multi-Vue? It uses icons as its commands. What does the OS-9 Shell do then?

Multi-Vue centers its operations around a graphics version of that OS-9 Shell. This other shell is called a graphics shell or more commonly, GShell. GShell provides a simple and friendly environment in which you can run applications, manipulate files and peripherals, and use the built-in desktop functions such as Calendar and Clock. GShell responds to mouse, joystick, or special keyboard operations.

Because GShell uses symbols for OS-9 actions and objects, you can learn to use its advanced windowing capabilities quickly. GShell's functions are easy to use, and in a short time you will be able to access those functions without referring to your manual.

Graphics Controllers: Using the Keyboard, Joystick, or Mouse

Multi-Vue centers its operations around icons instead of word commands, so the controller you use determines how easily you can manipulate Multi-Vue's graphics. A mouse is the easiest to use, requiring only one hand, but a joystick can be just as fast with a little practice.

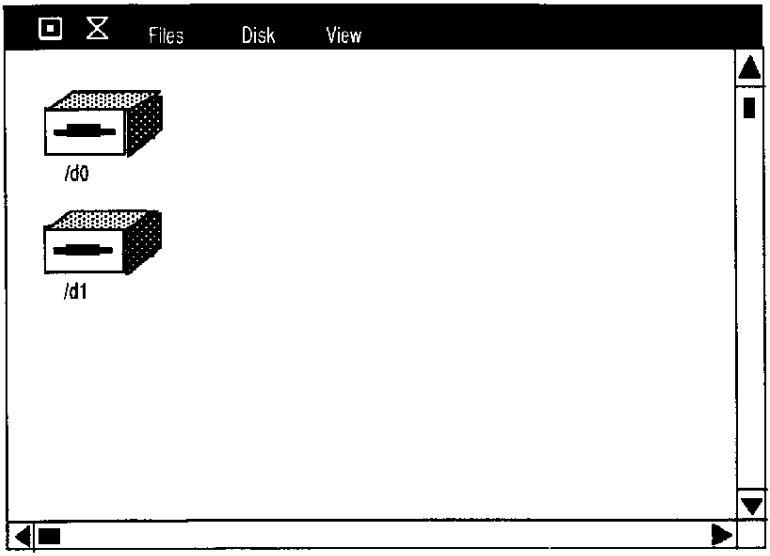
To use the keyboard as your controller, press **[CTRL][CLEAR]**. This allows you to use the four arrow keys to move the pointer. Pressing **[SHIFT]** while using the arrow keys moves the pointer in smaller increments. In place of the mouse or joystick button, use **[F1]**. Press **[CTRL] [CLEAR]** again to return to mouse or joystick operation.

While you are using the keyboard as your controller, the left arrow is not available as a backspace key. You can press **[CTRL] [CLEAR]** to turn off the keyboard mouse temporarily, or you can use **[CTRL] [H]** to backspace.

If you plan to use a mouse or joystick, use the *right* joystick port in the back of your computer. For more information on how to use the *left* port, see "Designating the Mouse/ Joystick Port."

Entering GShell

To enter GShell, boot your computer with Multi-Vue diskette 2, and change the diskettes as you are prompted. You will enter GShell:



This is the main screen. Through only a few icons, you can access all of the GShell functions.

Selecting GShell Options

By moving a small arrow-shaped pointer, you point to what you want to do, and GShell understands. If you are using a joystick or mouse, connect it to the right joystick port on the back of your computer. If you're not using a mouse or joystick, press [CTRL][CLEAR]. Then, you can move the pointer using the arrow keys on your keyboard.

To select functions, move the screen pointer to one of the icons, and click the button.

Icon	Function
	Terminates the current program or process and returns to the previous level.
	Accesses Multi-Vue desktop functions: Calc Clock Calendar Control Printer Port Help Shell Clipboard

Files Accesses OS-9 file manipulation commands:

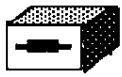
- Open
- List
- Copy
- Stat
- Print
- Rename
- Delete
- Quit

Disk Accesses OS-9 disk management commands:

- Free
- New Folder
- Format
- Backup
- Set Execute
- Set Devices

View Establishes screen resolution:

- Lo Res
- High Res



Selects the specified disk drive

There are also three screen pointers:

Screen Pointers	Function
	Make all of your selections with this pointer. Simply position it over icons or menu selections, and click the joystick or mouse button (or press [F1]).
	When your pointer changes to this, it's telling you that it's not in a valid location.
	If your pointer changes to this (an hour glass icon), the system is accessing the disk drives and it wants you to wait.

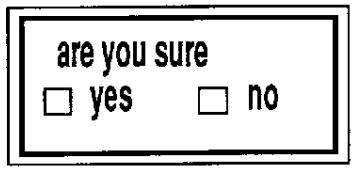
Scroll Markers

The Multi-Vue main menu also uses a simple system of position indicators that become especially helpful when only part of a file's contents can be displayed. These *GShell scroll bars* appear along the right side and at the bottom of the main menu.

As you move through the file, the scroll markers indicate your relative position. For example, if you are halfway through a written document, you'll notice that the scroll marker on the right side of the page is halfway down the scroll bar. The same is true for the bottom scroll bar, except it indicates right/left overflow.

Exiting GShell

To exit GShell, point and click on the closed window icon in the upper left corner of the GShell screen. A dialogue box will appear:



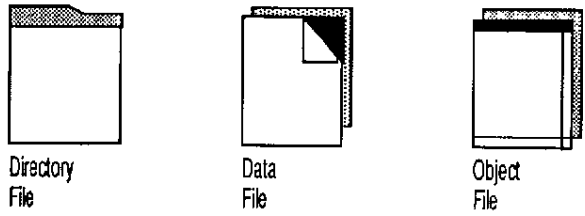
Point and click on the YES or NO boxes.

You may also exit by simply using the keyboard. Press Q to quit and then Y or N to answer yes or no.

You can access other keyboard controls also. By pressing $ you will select shell from the Tandy Menu. If you press [=], GShell will reread the current folder and redraw the screen

GShell File Icons

GShell recognizes three types of files: directory files (folders), data files (formatted or unformatted text files), object files (applications and utilities). GShell symbolizes these items with the following icons:



Directory files have icons shaped like file folders. In essence, a directory is a file folder that contains information about other files.

Data files have icons shaped like sheets of paper. (Note the bent upper-right corner.) These files contain data you will use in your application programs. This data can be unformatted text, specially formatted text from a word processing application or a data management system, graphic images, or other forms of data.

Object files have icons shaped like small Multi-Vue windows. Since object files contain executable programs, the open window icon represents the new window that needs to be opened to run the program. This type of object file is called a general object file.

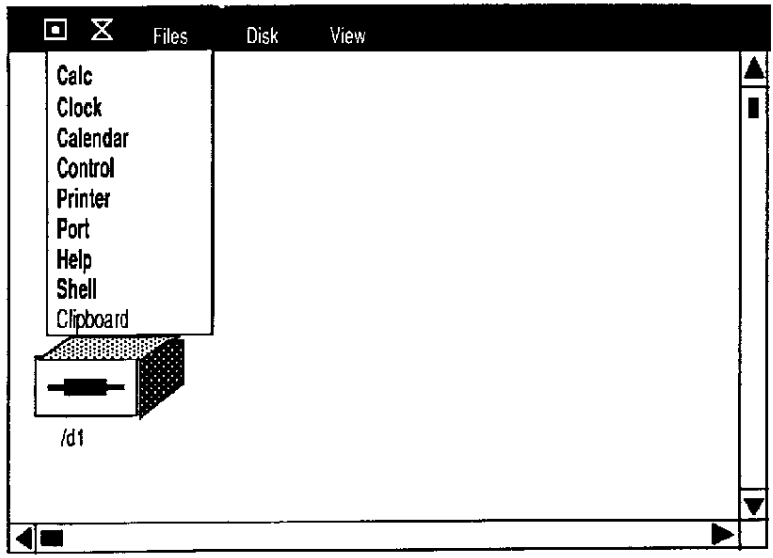
Using Menus

The GShell screen contains five menus. You can access these menus through their names or icons, which you'll find at the top of the window, in the *menu bar*.

If you are using a mouse or a joystick, position the screen pointer on the menu name, and click the button on your controller.

If you decide to use the keyboard, press **[CTRL][CLEAR]**. Now you can use the arrow keys to control direction and **[F1]** as the click button. You can toggle back to your mouse or joystick controller by pressing **[CTRL][CLEAR]** again.

In either case, a menu window opens and displays the options.



- The first icon is a small box with a dot in the middle. Think of it as a closed window, because that's what it will do. It lets you exit GShell and return to the OS-9 system or exit any program.

- The next icon is shaped like an hourglass. This represents the Tandy Menu, and it lets you select desktop functions.

- The **Files** menu contains options for file manipulation .

- The **Disk** menu lets you select commands for disk management.

- The **View** menu lets you select high or low resolution screens.

Multi-Vue lets you highlight only those items that are appropriate for selection. They appear in the menu in boldface type. If an option can't be selected, then it will not appear in boldface or highlight.

When Multi-Vue displays a list of options, make your choice by moving the screen pointer through the menu. When the option you want becomes highlighted (reverse colors), click the mouse or joystick button once, or press [F1] if you are using the keyboard.

Double Clicking

Some GShell functions require that you *double click* (press the button twice). For example, when you double click on a folder icon, GShell displays the files in the folder and gives you access to them.

If you enter a folder and then you double click on an application data file icon, the application chooses that data as its input.

However, entering a general folder and double clicking on a standard data file has no effect. The same is true for double clicking on an object file.

Executing Programs

There are three ways to execute an application. The first can be done by simply moving the screen pointer to the application icon and clicking. That application runs without any stored input.

You can also double click on an application data file icon. The application uses the selected data file as input.

A third way is to single click on a data file instead of double clicking. Then choose the **Files** menu, and select the **Open** option. Like the second method, this one supplies data to the application.

Executing Object Files

You can execute a general object file (small window icon), but you won't be doing it with two clicks. It's still simple, though. All you need to do is:

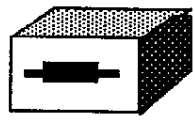
1. Select the file by clicking once on its icon.
2. Go to the Files menu and select Open.

An overlay menu asks you for any parameters that exist. For example, if you selected Copy, the old file name and the new file name would be the parameters.

If there are no parameters, press **[ENTER]**. GShell executes the application in the overlay window that drops down over the top of GShell's screen.

Changing Drives

If you have two or more disk drives, you'll need to access the drive that has the files you need. To access the contents of the disk drive you need, click on its icon. You'll find the drive icons on the left side of the GShell screen:



/d1

If you have drives other than those shown on the screen, you can add icons for them, too. From the GShell screen, select the Disk menu and the Set Devices option. An overlay window appears with a list of current devices. Press **[ENTER]** until the cursor is positioned at the end of the list, and type in the new device, for instance, **/d3**.

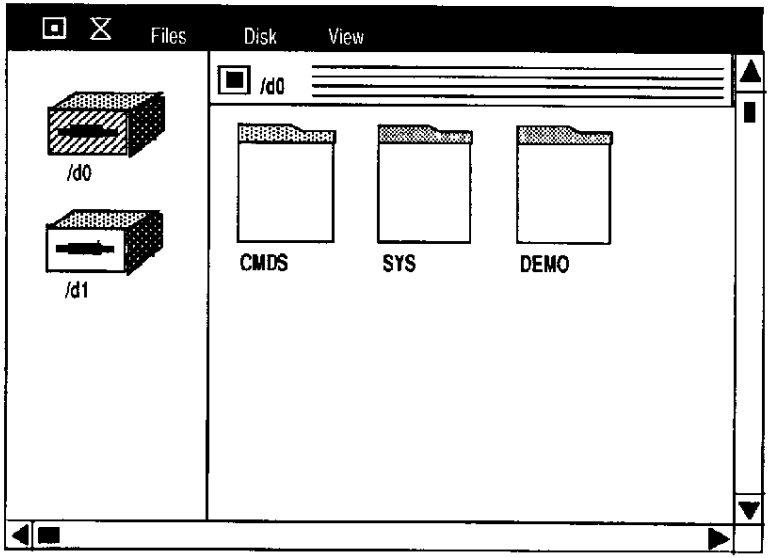
If you add a non-existent drive, its icon appears on the screen, however OS-9 won't let you access the device when it's not there.

You can indicate a maximum of five devices. If you already have five devices set, you can change a device by re-entering the SET DEVICES command. (Set Devices always starts with the device list in the environment file and then lets you add up to a total of five devices.

Changing Folders

When you begin looking for folders, you'll need to remember that the GShell structures its contents into various levels. The first level is the root level. By simply clicking on a disk drive icon,

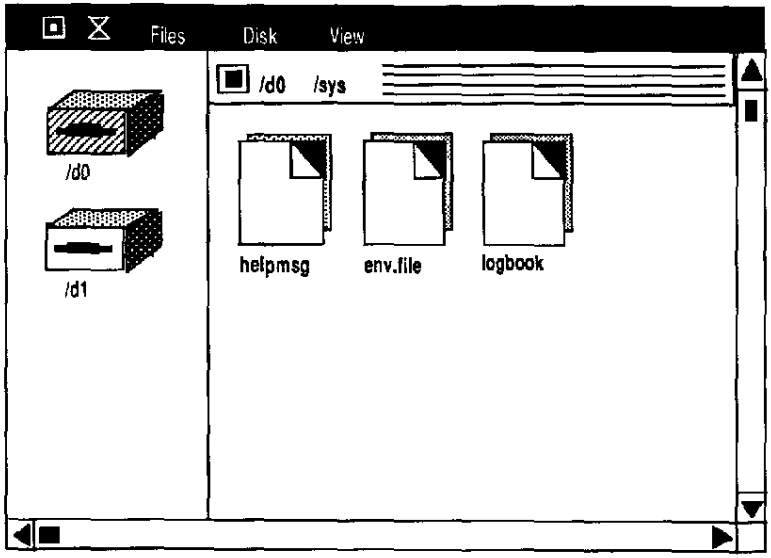
you can display the root level contents of a diskette. For example, if you have the Multi-Vue System Disk in Drive 0, and you click on the Drive 0 icon, then its root level contents are displayed with folder icons like this:



The next level occurs when you read what's in one of those folders. To do that, double click on one of the folders in that window. Now you'll see a new set of folders appear, with the name of their root folder on the title bar at the top of the window. The title bar names each level that you choose, like this:

disk drive / first folder / second folder / etc.

If you double click on the SYS folder, your screen looks like this:



If you use an application program which creates new files, GShell will not automatically update itself. The current folder can be updated by clicking on the striped part of the title bar, or by pressing [SHIFT] [=]. GShell will reread the folder and display the current information.

To trace your steps backward through the chain, you need to close each window. You can do this by clicking on the small closed window icon. Notice that there are two of these now. The one that you want to use is on the shorter bar with the stripes, not on the higher one that's solid.

If you choose the closed window icon in the upper left corner of the screen, an overlay window appears. It asks if you're sure you want to exit. If you answer "yes," you will make a complete exit of GShell to OS-9. "No" maintains the current screen.

Opening Multiple Windows

If you have a 512K system, you can have two or more applications running on the same screen at one time:

1. Select and execute the first application.
2. Press **[CLEAR]** to return to the main screen.
3. Select and execute the second application. GShell executes it on the same screen as the original application, if there is enough space.

You can repeat this method to install applications on the screen until you run out of room. Use **[CLEAR]** to cycle through applications. If one application is still running when you exit the screen, it continues after you exit.

If a screen is full or you wish to start an application on a new screen, press the spacebar. GShell then creates a new screen and its background window.

Sizing Windows

If an application you choose requires the full screen, it begins execution immediately. However, if it dosen't require a full screen, one of two symbols appears on the screen when you call the application.

If a circle with a slash appears, move the symbol toward the middle of the screen. This symbol means that the pointer is positioned too close to a window boundary to allow for minimum window size.

If a rectangle appears, move the rectangle to the screen position where you want the application to run. To establish a position for the upper left corner of the window, click the button or press [F1]. Then drag the opposite corner of the box to make it larger or smaller. You can drag the window by moving the mouse, the joystick, or the arrow keys. (The minimum size of each box is set by the AIF file of each application.)

When the window is the size you want, push the button a second time. GShell draws the application window and begins execution.

Desktop Applications

The Tandy Menu contains desktop applications that you can access through the hourglass icon:

- A *calculator*, complete with memory functions and decimal/hexadecimal conversions
- A *clock*, complete with an alarm
- A *calendar* (and memo book) that lets you select dates and write reminders for upcoming events
- A *control* panel from which you can adjust screen and keyboard settings
- A *printer* panel from which you can adjust printing formats
- A *port* panel from which you can control communication settings with other devices
- A *help* function to give you information on selected subjects. (Press [ENTER] for a subject listing, when the "What Subject(s)?" prompt appears in the Help overlay window.)
- A *shell* facility for starting a standard OS-9 shell from within Multi-Vue.
- A *clipboard* for moving data between files

Help Function

The help function accesses a file of Multi-Vue subjects. By selecting Help from the Tandy Menu (hourglass-shaped icon), you will see a window open and a prompt that asks "What Subject(s)?"

If you answer by pressing [ENTER], you will get a listing of the eight subjects available. If you want to broaden the list, you can merge other help files in with this one. You'll find help files with OS-9 Level Two software or in applications packages. See MERGE in the *Commands* section of *OS-9 Level Two Manual* for more details on how to expand the file.

Creating a Standard OS-9 Shell

By choosing the Shell option from the desktop menu, you can create a standard OS-9 shell that you can size to fill the screen or to use only a small amount of space. You can even place several OS-9 shells on the same screen.

The Control Panel

Chapter 3

The climate of the Multi-Vue environment can be easily adjusted. Through the control panel, you adjust everything from the type of monitor you're using to the colors and shades you prefer on your screen.

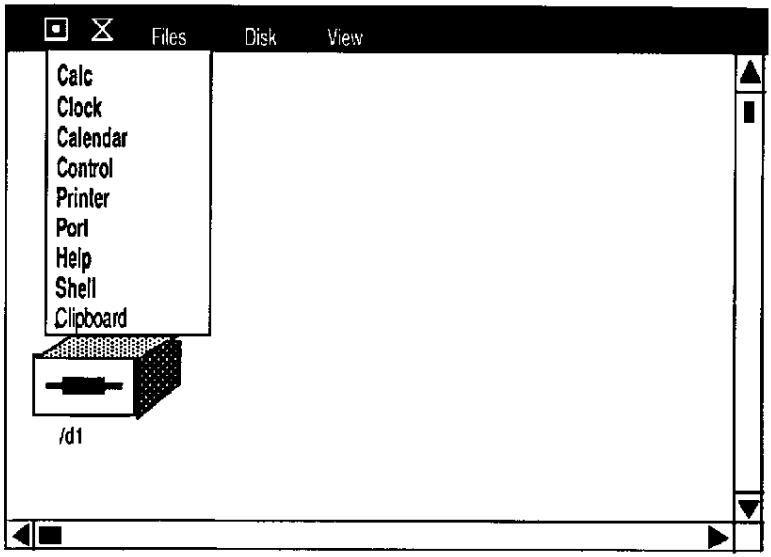
Something that you might want to reset right away is your monitor. The environment is originally set for an RGB monitor. You'll notice this when you access the control panel. You might even notice it earlier if you are using a composite monitor (TV type) because the colors will look odd. By using the control panel utility, you can change this along with other environment parameters. Even if you have an RGB monitor, you might still want to use the control panel to select your own screen colors.

Besides selecting monitor type and screen colors, you can also adjust keyboard response values and joystick resolution. The joystick resolution setting lets you use the high resolution joystick interface.

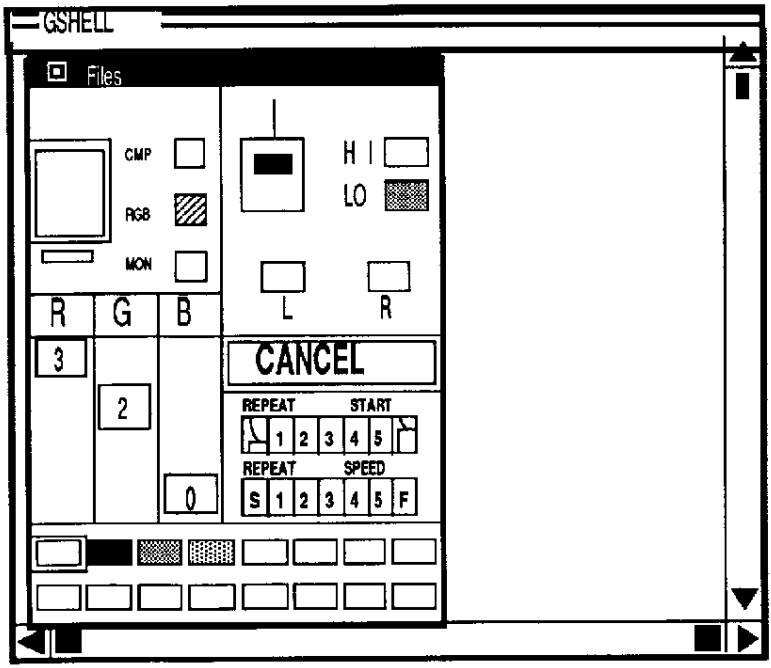
Accessing the control panel

Since the control panel is one of the desktop functions, you'll need to begin from the Multi-Vue main menu. Use your mouse, joystick, or keyboard to position the screen pointer on the Tandy Menu icon (hourglass shape) in the menu bar, then click the

button once. (If you are using the keyboard to position the pointer, press [F1].) A pull-down menu appears and displays the following options:

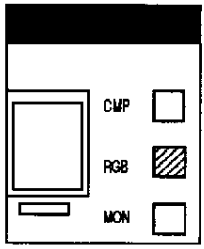


Move the pointer down until it highlights the word Control. Select the control panel by clicking the button once, or by pressing **[F1]** if you're using the keyboard. The control panel will be displayed as a window in the left half of the GShell screen:



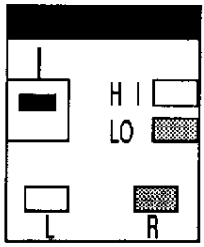
Setting the Monitor Type

The upper left corner of the control panel contains a monitor icon and three small boxes that let you indicate the type of monitor connected to your computer:



The three small boxes, labeled CMP, RGB, and MON, refer to Composite, RGB, and Monochrome. If you have something other than an RGB monitor connected to your computer, you'll need to move the pointer to the appropriate box, and click the button once. A highlight fills the box you selected.

Controller Resolution



The control panel offers high (HI) and low (LO) resolution for your mouse or joystick. In this option, resolution refers to the size of the steps that the pointer takes as it moves across the

screen. The low resolution pointer takes visible steps, but the high resolution pointer uses much smaller steps that allow it to move more smoothly.

To access this function, move to the window in the upper-right corner of the control panel. The icon in this window is a mouse. The HI and LO boxes to the right let you make your resolution choice.

You should choose HI only if you have a high resolution adapter for your joystick. Move the screen pointer to the small box labeled HI, and click the button once. The HI box becomes highlighted.

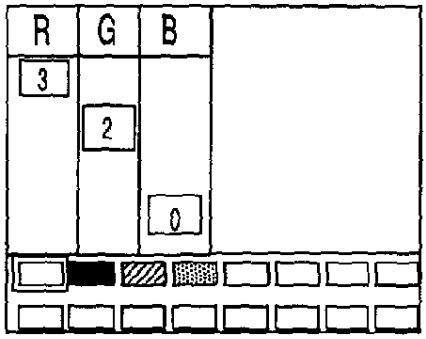
When you choose to exit the control menu (through the small closed window icon), an overlay window appears asking if you accept your changes. After you accept your choice by exiting the screen, connect your high resolution adapter into the cassette port and the designated port.

Designating the Mouse/Joystick Port

Below the mouse icon are two boxes. They are labeled L and R for the Left and Right ports. Since GShell requires the input from only one pointer, you only need to use one port. You can switch to the other port by positioning the screen pointer on the box and clicking. That box becomes highlighted. When you exit the control menu and accept the change, connect your mouse or joystick into the newly designated joystick port.

Setting Screen Color

The lower left part of the control panel has everything you need to adjust the colors in the Multi-Vue environment. If you are using a color monitor, you can customize the colors of your display. If you have a monochrome monitor, you can adjust your display for the sharpest possible picture.



The lower left corner of the control panel contains two rows of boxes. The two rows represent this color mixing palette register configuration:

Row 1:	0	1	2	3	4	5	6	7
Row 2:	8	9	10	11	12	13	14	15

Although the registers may have different assignments for each application, in general, the first four palettes have the following assignment:

Palette 0: **Background Color** (the color of the screen)
Palette 1: **Foreground Color** (the color of the graphics)
Palette 2: **Icon Color** (the color of the icons)
Palette 3: **Highlight Color** (the color of highlighted icons)

The color palette is separated into two parts. The vertical columns are for blending the colors, and the horizontal rows of boxes (color registers) are for selecting and displaying them. First, you need to choose the color register that you want to remix, then you can mix the right color.

Selecting The Register

To select the color register, simply click on the one that you want to change. The color register can't be highlighted because the highlight would block out the actual color, so it is outlined by a second box instead.

Mixing Colors

The vertical columns represent the Red, Green, and Blue mixing colors. You can mix these three colors in proportion with each other to blend 64 different colors. You set the proportions by positioning the pointer in one of the color columns, and clicking. The box moves up or down to the pointer's location. The important part is the number that appears. It tells how much you're mixing.

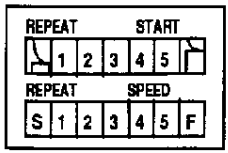
For example, if you set your R,G, and B columns to three, zero, and zero, you get a pure red color. That's because the proportions have been set to three parts red, zero parts green, and zero parts blue.

Notice that the mixed color depends on the proportionate amounts of all three colors. When you've finished mixing, you see the color change in the outlined color palette below.

Since GShell operates on a four-color screen, only four of the 16 color palettes can be displayed at one time. That's why you'll only see the first four palettes, even though you've set all of them.

However, an example will be given later which shows how to display all 16 palettes at the same time.

Keyboard Repeat



The area to the right of the vertical color columns lets you change keyboard repeat parameters. The two scales control the keys. Larger values in the top scale increase the length of time between when you press a key and when it begins to repeat. By selecting a repeat start value of 1, you can stop the keys from repeating.

Selecting larger values in the bottom scale makes the keys repeat faster.

Canceling Entries

CANCEL

You can use this function if you decide not to make the changes you selected. To access CANCEL, move the screen pointer into the CANCEL box and click the button. The CANCEL function restores all values to the default conditions stored in the environment file.

You will also delete all of your changes if you forget to *save* the new environment.

Saving the New Environment

If you are satisfied with your new environment, you can save the changes on the diskette. To save your changes, position the pointer over the word Files on the menu bar, and click the button once. An overlay menu appears:

Move the pointer until you highlight Save Env. Click the button once. Your disk drive turns on, and in a moment the changes are stored in the environment file.

If you choose Read Env. off of the menu, the screen is restored to the parameters stored on the environment file.

Selecting Abandon adjusts the screen to the parameters currently stored in the environment file and exits the control panel. Quit will also exit the control panel, but still keeps the changes until you end your session.

If you decide to exit the control panel by clicking on the small window icon, all of your changes remain intact until you save them, end your session, or run Control again.

Saving them allows you to store your changes on disk so that they can be recalled after a reboot or in subsequent uses of the control program.

The Calendar

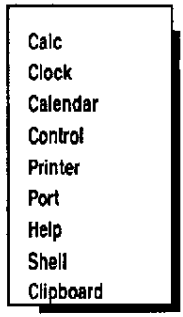
Chapter 4

Multi-Vue's Calendar provides a way for you to keep track of past and upcoming events and dates. You can look back to see what day of the week particular events occurred, or you can plan ahead and make notes for upcoming events. With the Calendar function, you can work with monthly calendars from January, 1901 until January, 2100.

Accessing the Calendar

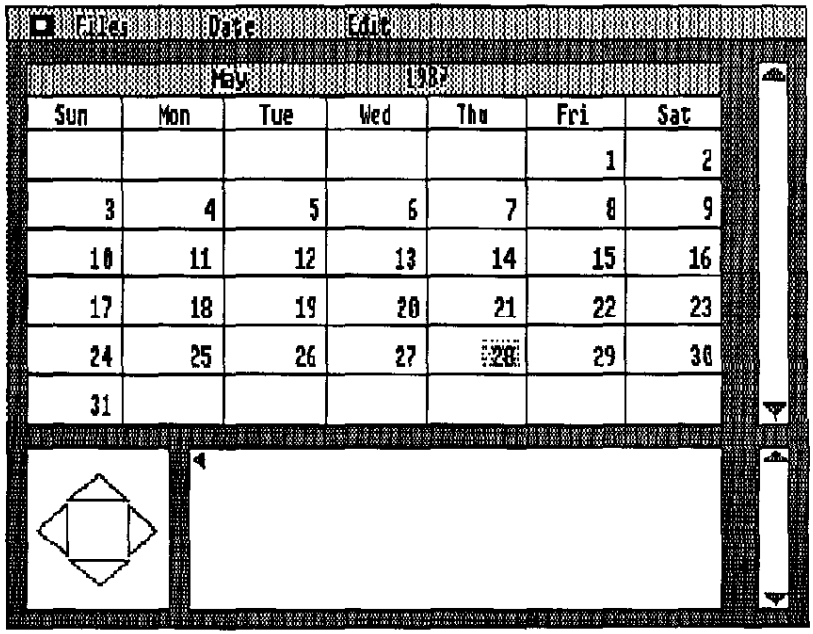
If you are at a place where you can access the Tandy menu (hourglass-shaped icon), then you can access the Calendar . Use your mouse or joystick to position the screen pointer on the hourglass icon on the menu bar. Click the button. If you are using the keyboard to position the pointer, press [F1].

A pull down menu appears and displays the following options from which you can select:



Move the screen pointer until it highlights Calendar, then click the button or press [↵]. Your disk drive becomes active and then the calendar appears.

If you entered the date when you booted OS-9, a calendar for the current month and year appears, similar to the following:



Setting the Date

If you did not enter the date when booting OS-9, the calendar is blank, and a window opens and asks you to type a date:

Type the date in the format indicated and press **[ENTER]**. The overlay window disappears and the calendar of the month you specified is displayed. The specified month and year appears at the top of the calendar, and the specified day is highlighted.

The date you specify is for the calendar display, and it doesn't change the system date.

Selecting Other Dates

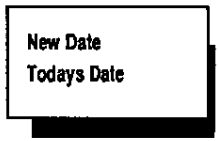
You can flip through the months (backward or forward) by:

- Using the month and year scroll bars on the right side of the screen.

- Selecting the Date option from the menu bar and typing in another date.

To use the scroll bars, simply put the pointer in the month or year bar, depending on which you want to change. Then, place the pointer over one of the small solid triangles, and click. To move into the past, use the triangles at the bottom of the bars. To move into the future, use the triangles at the top.

The other way to change the calendar is to use the pointer to select the Date option from the menu bar and click the button or press [F1]. A pull down file appears:



Move the screen pointer down until it highlights New Date, and then click the button. The date selection window opens in the middle of the screen:

Type the date using the format shown, and press [ENTER]. Multi-Vue displays the new calendar.

If you select Todays Date, then the calendar uses the system date.

Making Notes

Multi-Vue's Calendar also lets you write notes that will appear everytime you select a particular date.

Notes for the Day

The first thing you need to do is select the right date. For instance, suppose you want to remind yourself of several things on May 28, 1987. You can select the correct date by using the side bars, or by typing a new date.

If you are already in the right year and month, but the day is wrong, there are three ways to change days within the month. The first is to move the pointer to the right day and click the button.

The second way is to use the four large triangles in the lower left corner. By placing the pointer on one of the triangles and clicking, you can move the highlighting through the month until it rests on the right date. Clicking on a triangle moves the highlighting in the direction that the triangle points.

By using the arrow keys, you can control the calendar functions. Use the arrow keys to highlight any day of the month. By pressing SHIFT while you use the up and down arrow keys, you can change the months. By pressing CONTROL while you use the up and down arrow keys, you can change the years.

Writing and Editing Notes

When you have the correct date, you can begin writing your notes. To begin writing, select Edit from the menu bar, and click the button. You can also enter the edit mode by moving the pointer into the window along the bottom of the screen and clicking the button, or pressing ENTER.

In both cases the note cursor highlights the upper left corner of the window at the bottom of the screen. When that happens, you can write your notes.

The following control keys will be helpful as you edit your calendar notes:

Edit Function Keys

Key(s)	Function
[↑]	Moves the note cursor up one line
[↓]	Moves the note cursor down one line
[←]	Moves the note cursor left one character
[→]	Moves the note cursor right one character
[ENTER]	Terminates the current line and moves the cursor to the beginning of the next line
[ALT] [ENTER]	Exits the edit mode
[SHIFT] [↑]	Moves to the top of the edit box
[SHIFT] [↓]	Moves to the bottom of the edit box
[SHIFT] [←]	Moves to the beginning of the current line
[SHIFT] [→]	Moves to the end of the current line

Edit Function Keys

Key(s)	Function
ALT + ↑	Deletes the previous line if one exists
ALT + ↓	Deletes the next line if one exists
ALT + ←	Deletes the character to the left of the cursor
ALT + →	Deletes the line to the right of the cursor
CTRL + ↑	Moves to the top of the day's buffer
CTRL + ↓	Moves to the bottom of the day's buffer

The note cursor begins from the upper left corner of the Edit box. As you type, the cursor moves ahead of each character. If you make a mistake, use the arrow keys to position the cursor over the incorrect letter and type over the characters you don't want.

To move quickly through the type, hold down both the shift key and an arrow key.

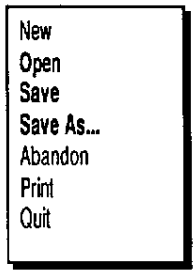
When you finish typing a line, or if you reach the right margin of the box, you must press **[ENTER]** to continue your notes. Pressing **[ENTER]** moves the cursor down one line and back to the left side of the box, just like the carriage return on a typewriter.

When you finish typing, exit by clicking the pointer anywhere outside of the Edit box at the bottom of the screen or by pressing [ALT] [ENTER].

Notice that when you exit the Edit mode, the current day is highlighted on the Calendar. This tells you that there are notes for that day. Calendar also highlights all other dates for which you enter notes.

Saving Notes

If you want the Calendar to retain the notes, you need to save the data you enter. You do this by positioning the screen pointer over the word Files in the Calendar menu bar and clicking the button or pressing [F1]. A pull down menu appears:



(The menu options shown in boldface type are valid options for you to choose. You cannot execute an option that is not shown in boldface type).

To save your notes, move the screen pointer down until it highlights Save As..., and then click the button.

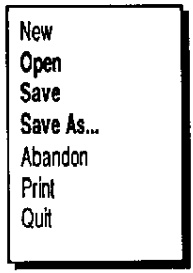
Selecting Save As... causes a new overlay window to open in the center of the Calendar box. The new window asks you to type a filename in which Calendar can save your data.

Type a filename, such as *Reminders*, and press **[ENTER]**. This tells Calendar to save your notes in a file named *Reminders*. If you want to store the *Reminders* file in a directory other than the current directory, specify a full pathname, such as:

/d0/dates/Reminders [ENTER]

Recalling Notes

Once you have saved your notes into a file, you can recall them from the Calendar at any time. Next time you enter the Calendar, call the Files menu by moving the screen pointer over the word Files and click the button or press **[F1]**. The Files menu again appears:



This time, move the screen pointer down until it highlights Open, then click the button or press [F1]. Again, an overlay window appears on the screen. Type the name of the file you previously saved and press [ENTER]. If you select (highlight) the same year, month, and day (May 28, 1987), the notes you typed appear in the box at the bottom of the screen.

You'll have an easy time using the note feature if you use the same filename each time. For example, if you use *Reminders* the first time, it will be easier to continue using it.

Adding Notes to an Existing File

You can add additional information to one of your note files any time you are operating in Calendar and have a note file open. (A note file is opened whenever you step through the process of selecting Open.)

After you type new notes, add them to the current file by highlighting Save and clicking the button. Calendar saves the file using the current file name.

Leaving the Calendar

You can exit Calendar in two ways:

- Position the screen pointer over the closed window icon in the corner of the Calendar menu bar. Click the button or press [F1].

- Select Quit from the Files menu.

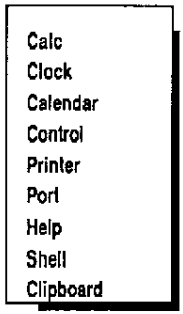
The Clock

Chapter 5

Multi-Vue's Clock is an easy way for you to set and view your system's time. As an added feature, the Clock has an alarm that beeps at any predetermined time, using your monitor or television's speaker.

Selecting the Clock

You can access the Clock using the hourglass icon (Tandy menu). Then, position your pointer on that icon, and click the button. If you are using the keyboard to position the pointer, press [F1]. A pull down menu appears and displays the following options from which you can select:

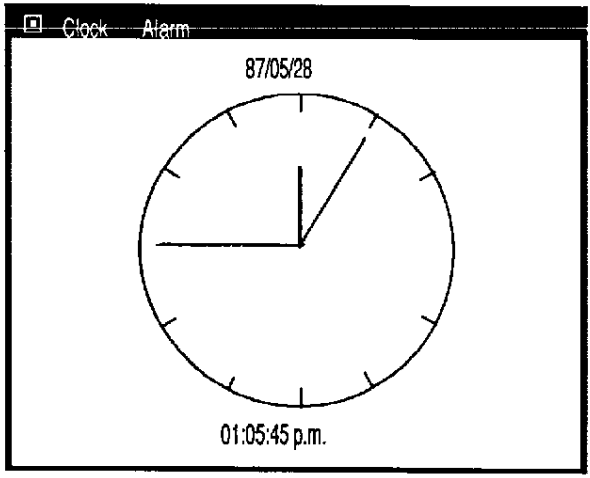


Move the screen pointer until it highlights the word Clock, and then click the button or press [F1]. Your disk drive begins to run, and in a few moments, a blank screen appears displaying either a rectangle or the circle/slash symbol.

If the circle/slash symbol appears, there's not enough room for GShell to draw the clock. Use the mouse, joystick, or keyboard to move the symbol until a rectangle appears. Position the upper left corner of the rectangle at the screen location where you want the clock, and click the button or press [F1].

If you want a larger window, use your mouse, joystick, or arrow keys to expand the window. The rectangle you see becomes the Clock's window. When it is the right size, push the button or [F1] a second time. (If a rectangle never appears, then the current screen dosen't have enough room. Simply press the space bar, [F2] or the second joystick button, and GShell allocates a new screen).

The clock screen will look like this:

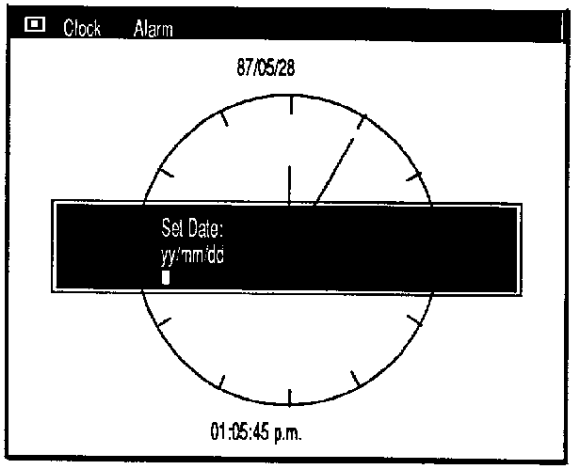


This screen provides an analog timepiece (represented by the moving hands on a circular clock face), a digital display, and the date.

If you did not type in the correct date and time when you booted OS-9, you can do it now by moving the screen pointer over the word Clock in the Clock display menu bar and clicking the button or pressing [F1]. A menu appears:

Setting the Date

To set a new date, move the pointer until it highlights Set Date. Click the button or press [F1]. An overlay window opens in the center of the Clock window and prompts you for a new date:



Type the date in the format indicated. For example:

87/05/28 [ENTER]

When you press **[ENTER]**, your system date changes to match what you typed (May 28,1987).

Setting the Time

To set a new time, move the screen pointer until it highlights Set Time, and click the button or press **[F1]**. An overlay window opens in the center of the Clock window and prompts you to enter the time.

Type the time in the format indicated. For example:

1/45/15 p.m [ENTER]

When you press **[ENTER]**, your system time changes to match what you typed. Both the analog and the digital clocks are reset to the time you entered.

Setting the Alarm

To set the alarm, move the screen pointer to the word Alarm in the Clock display menu bar and click the button or press [F1]. A menu displays:

To set your alarm, you must set both a date and a time. First, move the screen pointer to highlight Set Date, and click the button or press [F1]. When you do, an overlay window opens and prompts you for the Alarm date:

Type the date you want the alarm to sound. For instance:

87/05/26 [ENTER]

If you have not previously set an alarm date, Clock assumes that the alarm date is the current date.

Again, select the Alarm menu,and this time choose the Set Time function. When you do, an overlay window opens, and Multi-Vue prompts you for the Alarm Time:.

Type the time you want the alarm to sound. For instance:

5/30/00 pm [ENTER]

When you press [ENTER] the alarm time is updated to the time you set.

If your computer is on at the time you set, the alarm sounds even if you have exited the Clock and Multi-Vue. (Be sure that the volume on your monitor or television is turned up.) Remember, the alarm is cleared when the computer is turned off.

Checking and Clearing the Alarm

The Alarm menu has two other functions, Clear and Query. You can clear the Alarm setting by highlighting Clear in the Alarm menu and clicking the button or pressing [F1].

You can display the alarm setting by selecting Query in the Alarm menu and clicking the button or pressing [F1].

Leaving the Clock

To exit the Clock function, move the screen pointer to the closed window icon on the far left of the Clock menu bar,and click the button or press [F1]. You can also exit by selecting Quit on the Clock menu.

The Calculator

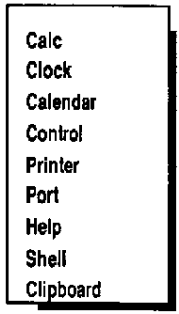
Chapter 6

Multi-Vue's Calculator lets you make quick calculations while using the OS-9 system or OS-9 application programs. The Calculator provides:

- Memory save , recall, clear, add and subtract functions
- Decimal add, subtract, multiply, and divide functions
- Hexadecimal add, subtract, multiply and divide functions
- Hexadecimal memory functions
- Base conversion between decimal and hexadecimal values
- Hexadecimal numeric range from 80000000 to 7FFFFFFF
- Decimal numeric range of $\pm$ 1.2 x 10^{38}
- Positive and negative signing of decimal numeric values

Selecting the Calculator

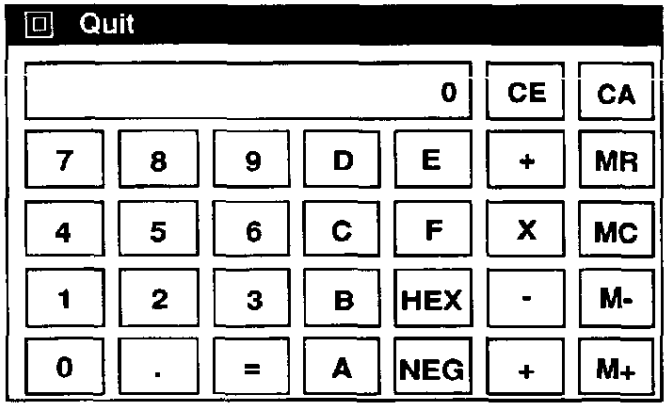
Position the screen pointer on the hourglass icon in the menu bar. Click the button. If you are using the keyboard, press [F1]. A pull-down menu appears:



Move the pointer until it highlights Calc. Click the button or press [F1].

If the circle/slash symbol appears when the new window opens, the box is too close to a window margin or there is not enough space for the calculator to fit on the screen.

Move the circle/slash symbol away from the margins until a rectangle appears. Position the upper left corner of the rectangle where you want the Calculator to appear and click the button or press [F1]. If the rectangle is the size you want it to be, click the button or press [F1] again. Otherwise, use the mouse, joystick, or arrow keys to enlarge the box to the the size you want, and then click the button or press [F1].



Using the Calculator

The Calculator looks like any other calculator: buttons and a display window.

You can *press* the keys by moving the screen pointer onto a Calculator key and clicking the joystick or mouse button, or by pressing [F1]. Or, you can activate the calculator keys by pressing an appropriate keyboard key.

The following table defines the calculator keys, their functions, and the keyboard keys that activate the calculator keys.

Key	Function	Keyboard Key
0	Produces the value of 0	0
1	Produces the value of 1	1
2	Produces the value of 2	2
3	Produces the value of 3	3
4	Produces the value of 4	4
5	Produces the value of 5	5
6	Produces the value of 6	6
7	Produces the value of 7	7
8	Produces the value of 8	8
9	Produces the value of 9	9
A	Produces the value of 10 (hex mode only)	A or a
B	Produces the value of 11 (hex mode only)	B or b
C	Produces the value of 12 (hex mode only)	C or c
D	Produces the value of 13 (hex mode only)	D or d
E	Produces the value of 14 (hex mode only)	E or e
F	Produces the value of 15 (hex mode only)	F or f
+	Performs addition	+
-	Performs subtraction	-
X	Performs multiplication	*
÷	Performs division	/
.	Produces a decimal point	.
=	Performs the equals evaluation	=
MC	Clears memory storage	T or t
MR	Recalls memory storage	R or r
M-	Subtracts the current display from the contents of memor	M or m
M+	Adds the current display to the contents of memory	P or p

DEC/ HEX	Toggles between decimal and hexidecimal operations modes	**ALT-D**
CE	Clears all entries since the last operator function	**ALT-E**
CA	Clears all current Calculator entries	**ALT-A**
NEG	Changes the sign of the value	**N or n**

To try the Calculator, from your keyboard, type:

99*44=

The answer, 4356, appears in the display.
Clear the Calculator's current operation by pressing [ALT] [A]. Then type:

1/.9=

The answer, 1.1111111111, appears.

Type:

/.9=

The number, 1.2345679012 appears.

You can also perform all the calculator functions by moving the screen pointer to a calculator key and clicking the button or pressing **[F1]**.

Leaving the Calculator

To exit the calculator, position the screen pointer on the window icon on the Calculator menu bar. Click the button or press [F1]. In a few moments, the Calculator screen is replaced with the Multi-Vue screen.

You can also push the Q key to quit.

Port and Printer Setups

Chapter 7

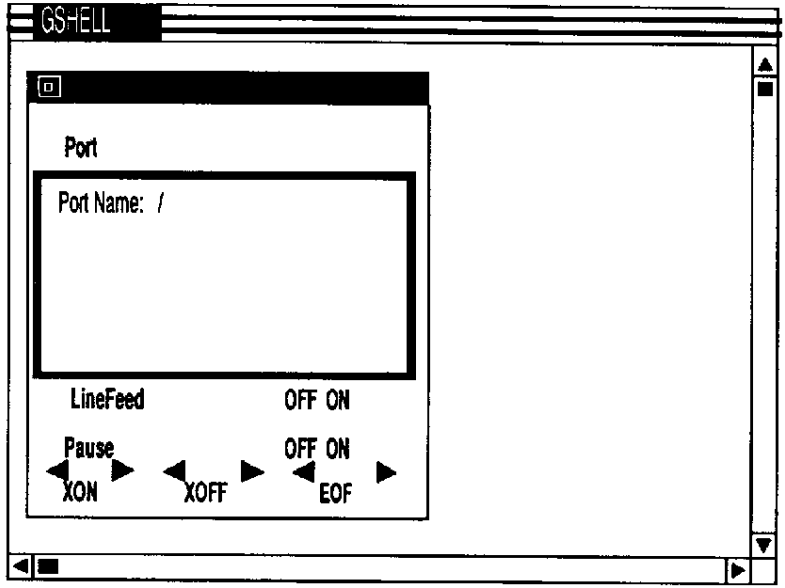
Multi-Vue has two functions for setting up RS-232 ports and controlling the operation of your printer. You can access two functions, Printer and Port, from the Tandy Menu (hourglass-shaped icon).

RS-232 Port Configuration

Move the screen pointer to the hourglass icon in the menu bar, and click the mouse or joystick button or press [F1]. This menu appears:

Move the screen pointer until it highlights the Port option. Then, click the button or press [F1].

The Port window overlays the current screen:



You'll notice another overlay window inside the port overlay window. Its prompt asks you for the name of the port you need to use for the new device. The names that you can use are:

/p (printer)

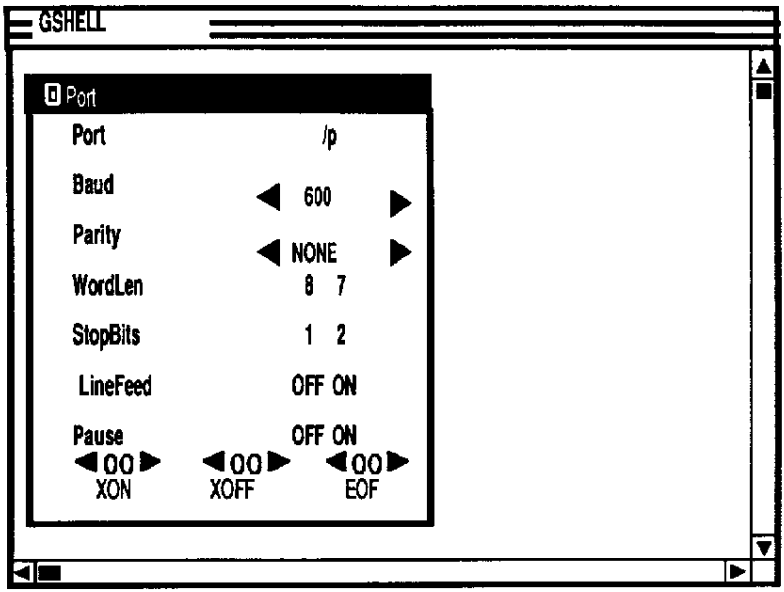
/t1 (built-in Color Computer 3 serial port)

You can use these devices also, after you install them in your boot file. (See your OS-9 Level II manual for more information on the OS9gen command.):

/t2 and /t3 (optional Deluxe RS-232 Paks)

/m1 and /m2 (optional Modem Paks)

Type the name of the port you want to change and press **[ENTER]**. The second window closes and reveals the full configuration menu:



The values shown (or highlighted) for each of these options are the current setting of that port.

By using your mouse, joystick or keyboard, you can move the screen pointer among these options and change the configuration of the specified port.

Changing the Baud Rate

Baud rate is the speed at which data is transmitted (in bits per second).

Since the equipment you work with will have pre-assigned baud rates, you need to select a baud rate that matches the equipment with which you want to exchange data. For example, if your printer accepts data at a baud rate of 2400, but your computer's printer port (/P) is configured for only 600 baud, then your computer will only transmit at 600 baud. To quadruple the transmission speed, set both your printer and the port to 2400 baud. If your printer is capable of a higher baud rate, refer to its owner's manual to determine how to select that option.

To change the baud setting, move the screen pointer to the triangle immediately after Baud on the Multi-Vue Port menu. Click the button or press [F1] to increase the baud rate to the new setting.

If you increase the baud rate too far, use the other triangle to lower the setting. Continue clicking until you cycle back to the baud rate you want.

Changing Parity

Parity is a method used by your equipment to detect transmission errors. Choose the parity that matches the equipment with which you are exchanging data (EVEN, ODD, MARK, SPACE, or NONE). Since some devices do not support all types of parity, see your OS-9 manuals for the supported values of your device. *You do not need to set parity for sending data to a printer.*

Position the screen pointer at one of the small triangles near the Parity indicator. Click your button or press [F1] to scroll through the options, one at a time.

Selecting Word Length

Word length describes how many bits form a byte. Most equipment you connect to your computer expects eight bits per byte. However, some equipment will expect only seven bits per byte.

Set the word length to match the equipment with which you are exchanging data by positioning the screen pointer on either the 7 or the 8 near WordLen. Click your button or press [F1].

Selecting Stop Bits

Transmitting equipment sends either one or two bits following a unit of data to indicate that the transmission of each unit is complete. You need to match the stop bit setting of your computer with the stop bit setting of the equipment to which it is connected. To do this, move the screen pointer to either the 1 or 2 following the Stop Bits option, and click the button or press [F1] to highlight your choice.

Selecting Line Feed

By turning LineFeed OFF or ON, you can match your computer's output to the requirements of the device to which it is connected. Some devices add a linefeed character (Character $0A) whenever your computer sends them a carriage return character (Character $0D), while others expect a linefeed to follow a carriage return.

If you send text to your printer and it prints all of the lines on top of each other, you need to select LineFeed ON. To do this, move the screen pointer to the ON selection opposite the LineFeed option, and click the button or press **[F1]**.

Selecting Pause

The pause option lets you select a pause in transmission at the end of each page of information you send to a device such as a printer or terminal. A page of information is determined by the lines-per-page setting of the device to which you are transmitting data.

If Pause is ON, your computer stops transmission at the end of a page and waits for you to press a key. If you want your printer to pause, so that you can insert the paper yourself (you might want to use letterhead). See the printer configuration section in this chapter.

To activate Pause ON, move the screen pointer to the ON selection opposite Pause, and click the button or press [F1].

If Pause is OFF, your computer sends data uninterrupted until transmission is complete. To activate Pause OFF, move the screen pointer to the OFF selection opposite Pause, and click the button or press [F1].

Selecting XON, XOFF and EOF

Transmission on, transmission off, and *end of file* are protocol communication characters that you can activate easily from the bottom of the port menu. When their values are set at 00, they are disabled.

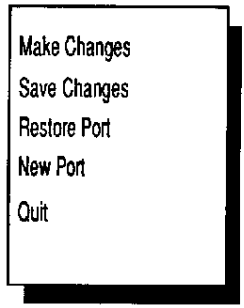
To activate them at their standard values, move the pointer onto XON, XOFF, or EOF. Click the button or push [F1]. The standard value is displayed. Click again to turn them off (00). (You can toggle back and forth). Their standard values are:

XON	$11
XOFF	$13
EOF	$1B

If you choose to use a different value, you can change the numbers by using the small triangles to the sides of the values. Cycle through the numbers by clicking the button or using the [F1] key.

Completing Your Selections

When you are satisfied with the changes, move the screen pointer to the word Port in the Port menu bar and click your button or press [F1]. A pull-down menu appears:



Make Changes is the equivalent of the standard OS-9 TMODE utility.

With it you can make changes to your terminal's input paths. However, Make Changes only makes temporary changes that will be cancelled when you exit the current process.

Save Changes is the equivailent of the standard OS-9 XMODE utility.

With it you can make changes to your terminal's input paths that will last until you reset or turn off your computer.

If you wish to make your changes permanent:

1. Make changes using PORT
2. Generate a new OS9 Boot onto your disk using the cobbler Command that comes wih OS9.

The changes will be in place every time you start. See *OS-9 Commands Reference* for more information.

Restore Port lets you reset the parameters to the initial values (or last values that were previously saved) before any changes were made to the port.

New Port lets you select another port for reconfiguration. When you select this option, an overlay window appears in which you type the name of a new port to configure. You can make changes to the new port by using Make Changes and Save Changes.

Quit closes the port configuration screen and returns to the Multi-Vue main screen. ***Quit does not save new changes.*** Select either Make Changes or Save Changes before selecting Quit if you want the changes you made to take effect.

Exiting Port Configuration

To leave the port menu and return to the Multi-Vue main screen, click on the Quit option in the Port menu, or click on the closed window icon on the Port menu bar. The port menu disappears, and the previous screen is restored.

Printer Configuration

The Printer option provides a way for you to set up the page configuration of graphic printouts and screen dumps. By selecting Printer, you can establish page margins, line length, and page length.

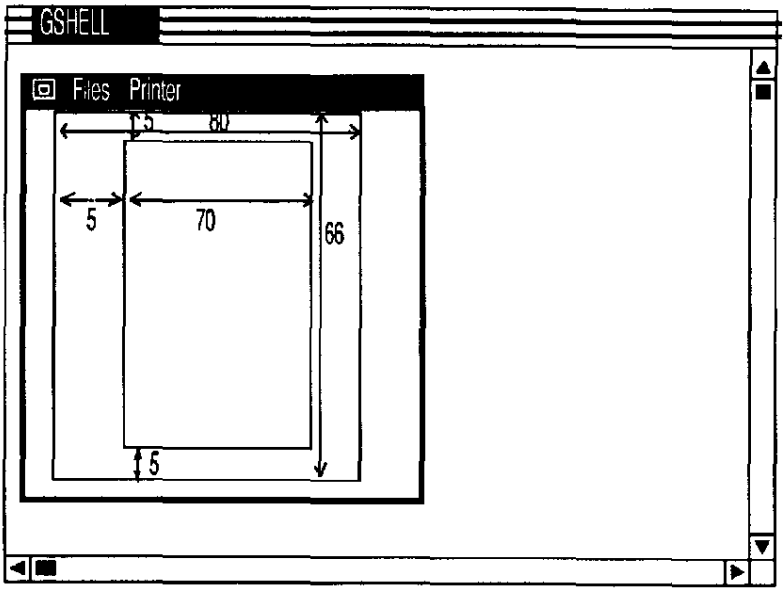
Printer does not directly affect your printer's operation. Rather, it sets parameters in Multi-Vue's environment file. Application programs can then use these parameters for graphic printing operations. For Printer to be effective, you must be running an application program that expects to find its printer format information in the Multi-Vue environment file.

When you choose Print from the Files menu, Print will fork COCOPR, which looks at the values in the environment file when formatting the text on the page.

Setting the Printer Format

From the Main menu select Printer from the Tandy menu by positioning the screen pointer on the hourglass icon at the top of

the screen and clicking the button or pressing [F1]. Then click a second time on the Printer option.Your disk drive turns on and in a moment the Printer screen appears:



This display shows a printer page with marked margins. Change the size of these margins by moving the screen pointer to the value you wish to change. Click the button on the value. An overlay screen appears, and asks for the setting value.

Type a new value and press [ENTER]. The values on the screen will change to match the values you type.

Save the values you select by moving the screen pointer to the word Files in the Printer menu bar and clicking the button or pressing [F1]. A pull-down menu appears with several options.

Move the screen pointer to highlight Save Format and click the button or press [F1]. After you have entered a new value, it will appear in the space for that margin.

If you don't like your changes, and you decide to restore the original format, choose the Restore Format option.

The values are set as follows:

Left Margin	the distance between the left edge of your paper and the the printed image
Top Margin	the distance from the top of the paper to the printed image
Bottom Margin	the distance from the bottom of the paper to the printed image
Page Length	the length of the sheet of paper, usually 66 inches
Line Length	the number of characters printed per line
Page Width	the width of the page, measured in characters

Other options, such as Printer Name, Printer Port, and Page Pause can be accessed through the Printer option on the menu bar.

Printer Name enables graphics dump applications to send appropriate commands to your printer. The default printer and name is "DMP 100." To change it, select the Printer Name option. An overlay will prompt you for the new name.

Printer Port is the /p descriptor that receives printer data. If you have a special printer device descriptor that you need to change, you should change it through this function.

Page Pause is the last printer option. It asks how many pages need to be printed until output is paused. The default value, zero, turns pause off. A value of one will cause the printer to pause after each page.

Exiting from Printer Formatting

You can exit Printer by clicking on the small closed window icon in the Printer menu bar, or by selecting Quit from of the files menu. Be sure to save your new formats before exiting.

The Files, Disk, and View Menus

Chapter 8

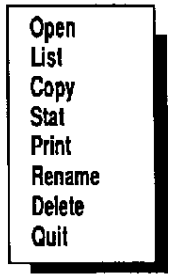
GShell uses three other menus that allow you to easily execute tasks that would otherwise require difficult commands. By simply clicking on the appropriate menu name in the menu bar, you can access functions that control data filing, disk storage, and screen resolution.

When you use these menus, you'll need to remember to highlight the appropriate drive and file icons *(current drive, current file)*.

Also, remember that Diskette 1 should be kept in Drive 0. Diskette 1 contains the information GShell needs to manipulate the files.

Files Menu

When you choose Files from the menu bar, a menu will appear:



These commands offer the following functions for complete file manipulation:

Open executes an object file. Since you can't double-click on an object file, the Open command substitutes for the double-clicking process.

List displays the contents of the current datafile to an overlay window in the GShell screen.

Copy duplicates data or object files to any of your disk devices. It requires a new name for the new file.

Enter the new name in one the following formats:

1. DIRECTORY / NAME
2. /DRIVE / NAME
3. /DRIVE / DIRECTORY / NAME
4. NAME (to be saved in the current directory of the current device)

Stat gives extended statistics about a file or folder. Since Stat gets its information by calling the FSTAT command, you can access this this same information inside or outside of GShell. Use FSTAT outside of GShell (on a comand line), and Stat inside GShell.

Print lists a data file to a standard printer path. The data file must be a text file. Print may be accessed outside of GShell, also, using COCOPR. Any printing jobs will use the parameters from "/dd/sys/env.file."

Rename assigns a new name to a file or folder. An overlay window willask if you're sure. You can respond by using the Y or N keys. A second window will ask you for the new name. (Remember, if you want to exit any of these sequences, press the BREAK key.)

Delete deletes a file or folder. An overlay window will ask if you're sure.

Quit exits GShell. This function is the same as choosing the closed window icon in the upper left corner.

DISK Menu

The DISK option has six commands that give you control of disk operations. By clicking on DISK in the menu bar, the following menu will appear:

Free gives the total free disk space available on the current drive.

New Folder creates a new file folder. You will need to specify a new folder name.

Format allows you to format a diskette without exiting GShell. You need to choose a format device. By choosing /d0, you'll need to exchange the Multi-Vue diskette with the disk being formatted.

Backup makes a backup of an entire disk device. A prompt will ask you for your source device and your destination device.

Set Execute allows you to reassign your execution directory.

Set Devices enters new devices into the environment file. The new device will be displayed as an icon in the icon column on the left side of the screen.

View Menu

You can change the resolution (scale) of the display by accessing the VIEW menu:

By choosing Hi Res, the icons and all other screen characters will become condensed. Although they will be smaller, twice as many icons will fit on the screen. This is especially helpful if you have many files on one diskette; they can all be displayed at the same time. This option is only available on 512K machines.

Lo Res is the default screen. Its icons are twice as large and easy to read, but there are fewer per window.

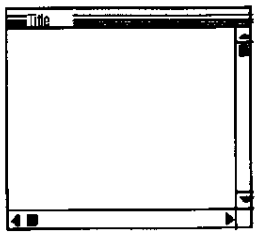
Programmer's Notes

Chapter 9

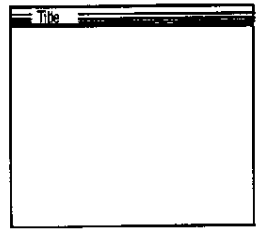
Multi-Vue's enhanced windowing system provides programmers with the tools to establish and manipulate windows with ease. Not only do you have all of the standard OS-9 Shell's windowing support, but Multi-Vue adds bordered windows and provides the SetStat and GetStat calls to manage them. It also adds calls for monitoring mouse and joystick ports.

Window Types

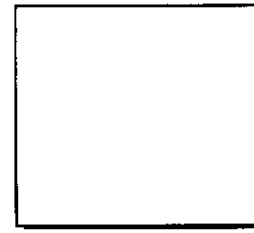
The types of bordered windows are: framed window, framed window with scroll bars, shadowed box, double box, and plain box. The system also supports a borderless window called *No Box*.



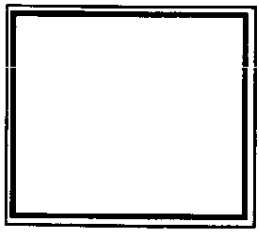
WT.FSWin
Framed Window
with Scroll Bars



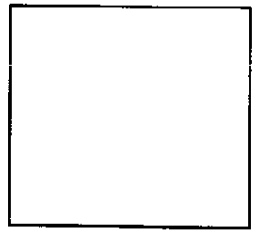
WT.FWin
Framed Window



WT.SBox
Shadowed Box



WT.DBox
Double Box



WT.PBox
Plain Box

Window Regions

Multi-Vue defines windows in terms of regions. All types have a *user region* and a *content region*, in addition to a *border region*. The user region is the area in which a user can select icons and menus to perform actions. The content region is the area that an application program can directly control and in which it can display its data.

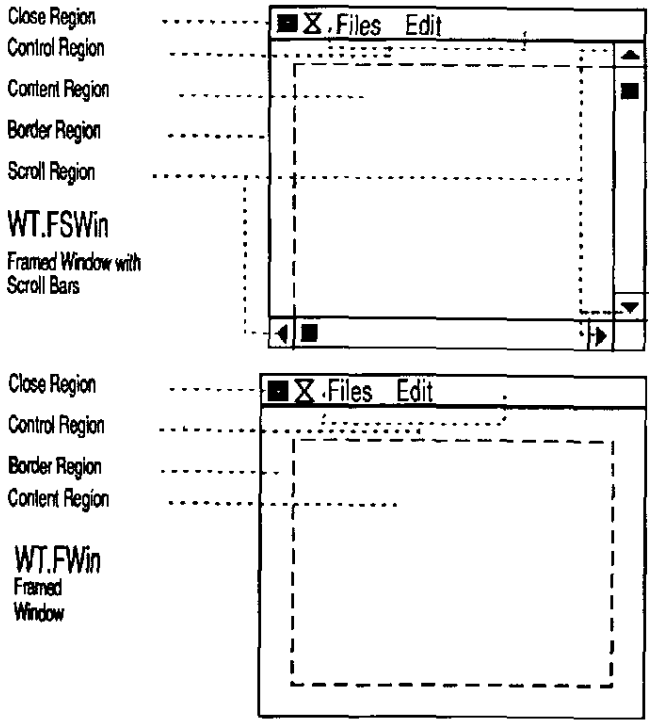
Some bordered windows also have special *control regions* reserved for system use.

Framed Windows

Application programs run in either *framed* windows or *framed windows with scroll bars*. Within the user region, framed windows have:

- The *Control Region* -- This region contains the *menu bar*, so if the user clicks the screen pointer in this region, the application performs the appropriate action.

- The *Close Region* --This region allows the user to quit the application and close the window by clicking on the the closed window icon.
- The *Scroll Region* -- When applicable, the user can click in these regions to scroll the display (only on a framed window with scroll bar).



Both types of window have a 1-pixel border around the sides and bottom and either a title bar or a menu bar at the top. As shown, one type of framed window has areas for scroll bars while the other does not.

You can write programs that expand the content region to include all of a framed window (except for the title bar). To do this, use the "Change Working Area" system call. However, your program must ensure that the border lines of the window are not destroyed.

Because OS-9 is a multi-tasking operating system, you can have more than one application running at a time on the same screen. Although many processes can concurrently transmit output to their windows, only one process can receive input from the keyboard at a time. If you have several processes running on a screen, it is important to know which process *owns* the keyboard input (which window is *active*).

The windowing interface helps you keep track of the active window. When a framed window is *passive* (not receiving keyboard input) the system displays the title bar at the top. When a framed window is active, the system displays the menu bar in place of the title bar.

Use **[CLEAR]** to select a new active window and make the current window passive. To select a new active window, but in the reverse direction of the rotation, press **[SHIFT]** **[CLEAR]**.

Clicking on a scroll bar arrow causes the menu select call to return a value specifying which scroll arrow you selected. Your application must scroll the display and call the system to manage

the scroll marker located within the scroll bar. Use the SetStat, SS.SBar, as outlined in the following example, to handle the scroll marker.

SETSTAT SS.SBar -- Update Scroll Bar

Entry Conditions:

A = *path*
B = *function code* (SS.SBar)
X = *x coordinate of horizontal scroll marker*
Y = *y coordinate of vertical scroll marker*

Exit Conditions:

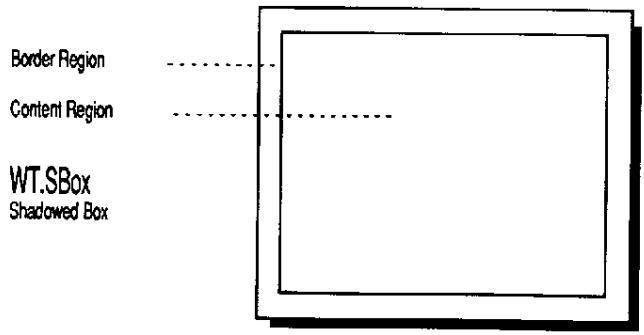
Scroll marker updated if no IO error occurs
Error code returned if out of bounds

Your application must calculate the coordinates to use for the scroll markers. It can use the SS.ScSize Getstat call to provide the information for the computation (the horizontal and vertical size of the content region of the window). The scroll bar size is one character cell less than the vertical and horizontal sizes returned from the SS.SCSiz GetStat call.

When the window is first created, the content area is the size returned by the SS.ScSize Getstat call. The content region is smaller than the window and its borders by two character widths horizontally, and three character heights vertically.

Shadowed Box

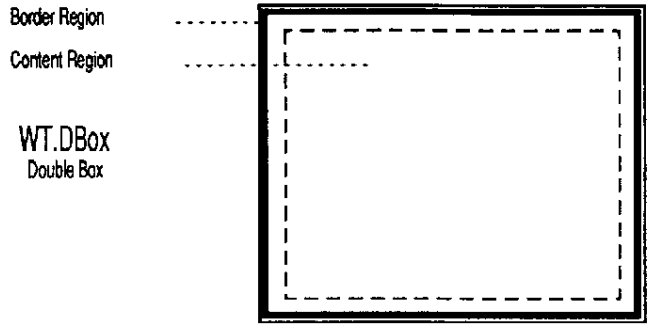
Multi-Vue applications commonly use shadowed box windows with pull-down menus. The shadowed box is outlined with a 1-pixel border with a 2-pixel shadow on the right and bottom borders. The window interface automatically creates a shadowed box whenever you initialize a pull-down menu.



Double Box

Programs usually use double box windows for *alert* or *dialog* boxes. An alert box usually indicates an error situation. It can also provide the information needed to remedy the error. A dialog box usually asks for specific information needed to complete an action, such as the name of a file for a Load operation.

The double box has a 1-pixel border on all sides, a 1-pixel space, then a 2-pixel inside border.



Plain Box

The *plain box* is a utility window that you can use for a variety of purposes, such as tool box utilities and notes. It has a 1-pixel border around the window.

No Box

A *no box* is a window without a border. Normally, you use this type for text screen windows in pull-down menus, dialog boxes, and alert boxes. You can distinguish this window from other areas by changing the background color when you create it. The content area consists of the entire window.

Window Conventions

The Multi-Vue environment does not enforce the previously described uses of window types. However, adopting these conventions provides consistency and makes it easier to learn how to use the Multi-Vue system.

Creating Windows

When an application begins execution, GShell automatically opens a window as the standard output path for that application. However, before the application can use that window, it must specify the type of window (framed, plain box, double box, shadowed box, no box) for the system. If the application uses menu bars and pull-down menus, the window must be one of the framed types.

If the application needs overlay windows for alert or dialog boxes, it must use the OWSet command to set the overlay windows.

If it uses another device window, it must open a path to the selected /w device. Then, it must use DWSet to set the device window's size, location, and type.

Use the OS-9 SetStat command as follows to specify the window type:

SETSTAT SS.WnSet - Window Setup

Entry Conditions:

A = *path*
B = *function code* (SS.WnSet)
Y = *window type definition*
WT.FWin = Framed window
WT.FSWin = Framed window with scroll bars
WT.SBox = Shadowed box
WT.DBox = Double box
WT.PBox = Plain box
WT.NBox = No box
X = *pointer to window/menu data for framed windows* (unused for other window types)

Example call to set up framed window:

```
lda    #1              use stdout
ldb    #SS.WnSet       get function code
ldy    #WT.FWin        get the type code
leax    WinDat,u       point to the window and
                       menu data
 os9   I$SETSTT        call the setstat
```

Example call to set up a double box:

```
lda #1                 use stdout
ldb #SS.WnSet          get the function code
ldy #WT.DBox           get the type code
os9 I$SETSTT           call the setstat
```

When you establish a framed window, Register X must point to a data structure that defines the window title, the menu bar, and any menus for the pull-down windows. This data structure is defined in full in the following section and is illustrated in the assembly language and C language definitions.

Menu Bar and Pull-Down Menu Support

The framed window is the only type of window that supports a menu bar. Therefore, any application that uses a menu bar and pull-down menus must run in a framed device window.

The menu bar is located at the top of the window as long as the window is active. When a window is passive, the title bar appears in that location.

If you properly set the menu bars and pull-down menus, the system handles most of their operations. Your application must provide the system with a pointer to the required data structure at the time a command to set a framed window is executed. This data structure must hold the title of the window, the title of all menus, and the items for each pull-down menu. Once the setup is complete, the system creates the menu bar, the pull-down menus, and the entire selection process.

Using the Mouse

To access the menu-handling capabilities of the system, the application must be able to make use of the mouse and the system-wide graphics pointer. Under the enhanced windowing environment, you can link the mouse to the graphics cursor, with the cursor acting as a pointer for the mouse. You can then direct

this screen pointer to the appropriate screen location, such as a menu title, by moving the mouse. The application monitors the mouse status and takes action based on the mouse position and the click information.

The simplest method of using the mouse in an application is to use the SS.Mouse GetStat call to constantly monitor the mouse status. This call returns information on the mouse location, the number of clicks made, and the state of the button (up or down). Use this method for handling the mouse during command processing--for example, to draw graphic images on the window.

Almost all mouse-driven events use the initial mouse click to tell an application that the user is ready to take action. Therefore, an application's initial code can look for a signal from the SS.MsSig SetStat call after setting-up the appropriate signal handler, using the F$Icpt service request. When you click the mouse button (change the state from up to down) the application receives a signal indicating the change. The application can then read the mouse status using the SS.Mouse GetStat call, and determine the action to take.

The application can check the *valid flag* Pt.Valid to determine whether it has *ownership* of the keyboard and mouse. If it has lost ownership, it must take the appropriate action. This action might be nothing more than repeatedly checking the mouse or sending a series of sleep calls followed by a mouse check. If the application does own the mouse, it can examine the *location flag* Pt.Stat to determine whether the pointer is in the content region or in the control region of the window.

If the mouse is in the content region, the application can respond with the appropriate action based on the coordinates of the

pointer. If the mouse is in the control region, the application can request a menu selection via the SS.MnSel GetStat function. (See "Setting Up the Menus.")

The following is an example of the basic structure for an application using the mouse:

```
/* application initialization routine */

intercept(intfun);          /* set up signal intercept routine */
_ss_msig(0,msig);           /* set up for mouse click signal */
    .
    .
/* main processing loop of program */

if (moussig) {
    _gs_mous(0,mspacket);       /* read the mouse */
    if (mspacket->pt_valid == 0){

        /* not owned any longer, take action */

    } else {
        /* still own mouse, check location */
        if (mspacket->pt_stat == CONTREG) [

            /* take action for click on window region */

        } else {
            /* mouse on control region now */
            /* request menu selection */
            menid = _gs_mnsel(wpath,&item);
            switch (menid){
                /* base action on menu id and item */
            }
        }
    }
```

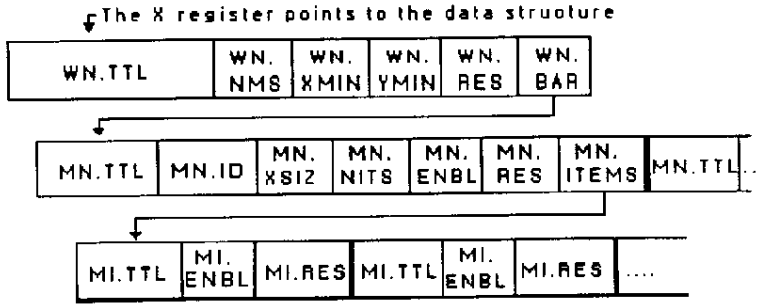
Setting Up The Menus

To set up the menu bar and pull-down menus, an application must create a data structure in its address space that contains all the information needed by the system to handle the menus.

If the structure is not to be modified, it can be in the code area. Use the fcs, fcc, fdb, and fcb assembler pseudo-opts to create the code. If you install the data structure in the code area, your application cannot change the menus while operating. (It is not possible to enable or disable menu items or entire menus interactively because OS-9 does not allow for self-modifying code.)

If you want your application to be able to modify menus while operating, keep the data structure in the process's data area.

The following illustration represents a data structure layout:



The following assembly code defines the data structure code that is required:

```
*
* descriptor for window title/menu bar line
* one required for each process running in a framed
* window
          org  0
WN.TTL    rmb  20    title for window title bar
WN.NMNS   rmb  1     number of menus in menu bar
WN.XMIN   rmb  1     min. horizontal window size
WN.YMIN   rmb  1     min. vertical window size
WN.SYNC   rmb  2     sync bytes $C0C0
WN.RES    rmb  7     reserved bytes
WN.BAR    rmb  2     pointer to an array of menu
*                    descriptors
WN.SIZ    equ  .     size for window data structure

* menu descriptors:
*   One is required for each menu in the menu bar
*   The Menu array descriptor is pointed to by a pointer
*    in the previous structure in the MB.MPTR area.

          org  0
MN.TTL    rmb  15    menu title (15 chars max)
MN.ID     rmb  1     menu id number (for selection
*                    1-255)
MN.XSIZ   rmb  1     horizontal size of desired
*                    pull-down
MN.NITS   rmb  1     number of items in the menu
MN.ENBL   rmb  1     enable flag
MN.RES    rmb  2     reserved bytes
MN.ITEMS  rmb  2     pointer to menu item
*                    descriptor's array
MN.SIZ    equ  .     size of menu descriptor
```

```
*
* menu item descriptor:
*   One is required for each menu item in a pull-down
*   menu
*   The ITEMS menu array is pointed to by the ITEMS
*   pointer in the previous data struture
            org  0
MI.TTL      rmb 15          item name (15 chars max)
MI.ENBL     rmb 1           enable flag
MI.RES      rmb 5           reserved bytes
MI.SIZ      equ             size of menu item descriptor's
                            array
```

Note: All strings are terminated with a NULL ($00) byte. The data structure definitions are also in the C language and assembly language Defs.

When you make a SS.WnSET SetStat call to set up a framed window, set Register X to point to the WN.TTL item of the data structure. The WindInt interface uses that pointer to locate the data structure in the process's address space. It then sets up the title bar or the menu bar using the information in the data structure.

You can have a maximum of ten menus for any menu bar, and each menu can have a maximum of 20 items. Keep the number of menus and items as small as possible so that users can make selections easily and quickly. One of the major benefits of a window environment is its simplicity.

The MN.ID field in the menu descriptor selections is the menu ID number, in the range 1-255. The system uses this ID as a return value in the menu selection call to identify the menu from

which a selection is made. Microware uses values in the range 1-127 to identify standard menus (discussed in the following section). If you want to add items to the menu ID list, use values in the range 128-255.

Microware has reserved WN.RES, MB.RES, and MI.RES for future updates and applications. Do not use these definitions. If you do, your application might not run under future releases of the Multi-Vue package.

Updating Menus

Your application must have a way to update the menu bar information in order to change the menu bar or menu item status (to enable and disable items). You can provide this ability using the SS.Umbar SetStat call. This call forces the system to re-read the menu bar data and update the menu bar accordingly:

SETSTAT SS.UMBar - Update Menu Bar

Entry Conditions:

A = *path*
B = *function call* (SS.UMBar)

Exit Conditions:

None

Menu Selection

The WindInt interface module handles most menu selections. However, your application must monitor the pointer position using the mouse data GETSTAT call. If the mouse is in the control region, the program can call the menu selection routine.

The call for menu selection is outlined as follows:

GETSTAT SS.MnSel - Menu Selection

Entry Conditions:

A = *path*
B = *function call* (SS.MnSel)

Exit Conditions:

A = 0 (if no selection was made)
= *menu id* (if selection was made)
B = *item number* (if selection was valid)

Example call:

```
lda     #path           get the path
ldb     #SS.MnSel       get the code
os9     $GETSTT         make the call
```

When this call returns without error, Register A has the menu information and Register B has the item information. Your application can test these values to determine the action to take:

```
IF (A) = 0 THEN no selection
ELSE take action for ID=A and item=B
ENDIF
```

If the menu does not have pull-down items, the item number returned is 0. You can use this feature to set up special menus for selecting graphic items. For example, you might create a menu, *Tools*, that has no items in it. When you select *Tools*, the call returns the menu ID, and B contains a value of 0. The application can now open a menu with the tools drawn in it and let the user make a selection among these tools.

Standard Menus

Standard menus in Multi-Vue provide consistency. They make it easier for a user to move from one application to another, because of similar menu organization. The system does not enforce these standards.

As previously noted, the standard menu ID values range from 1-127. The currently defined menus are:

ID Number	Definition
2	**Close Box.** The close application box is selected. Your application must follow the necessary procedures to exit.
4	**Scroll Up.** The scroll up arrow is selected. The application must scroll the screen up and make the appropriate call to move the vertical scroll marker to the new location. This function applies only to a WT.FSWin window.
5	**Scroll Down.** The scroll down arrow is selected. The application must scroll the screen down and make the call to the system to move the vertical scroll marker accordingly. This function applies only to a WT.FSWin window.
6	**Scroll Right.** The right scroll arrow is selected. The application must scroll the screen horizontally to the right and make the call to move the horizontal scroll marker. This function applies only to a FT.FSWin window.

7 **Scroll Left.** The left scroll arrow is selected. The application must scroll the screen horizontally to the left and make the call to move the horizontal scroll marker. This function applies only to a FT.FSWin window.

8 **Character Pressed.** A keyboard character is pressed. The application needs to read one character to identify the key. You can use this function to let your application associate keys with menu items.

20 **Tandy Menu.** The Tandy menu hourglass icon is selected. The Tandy menu displays the list of desktop utilities available with Multi-Vue. When one is chosen, fork to the specified utility, and wait for its execution to complete before returning control. If your application adds new desktop utilities, be sure it adds them at the end of the list. The format of the menu is:

Item #1 = Calculator
#2 = Clock
#3 = Calendar
#4 = Control
#5 = Printer
#6 = Port
#7 = Help
#8 = Shell
#9 = Clipboard

21 **Files Menu.** The File menu is selected. The format of this menu is:

Item #1 = New
#2 = Open ...
#3 = Save
#4 = Save As...
#5 = Abandon
#6 = Print
#7 = Quit

The application must perform any actions associated with the menu selections. The system does nothing but return the menu ID and item numbers.

The Open and Save As items are followed by three dots. These dots indicate that a dialog box asking for a filename is to follow when the user selects one of these options:

New starts a new, unnamed file. Use Save As to name the file.

Open reads an existing file into the application.

Save saves a named file.

Save As gives a newly created file a name and saves it or saves an existing file under a new name.

Abandon erases the file in memory and does not save it.

Print sends the current file to the printer device. First, configure the printer with the Port utility.

Quit ends the application and closes the window.

ID Number	Definition

22 **Edit Menu.** The Edit menu is selected. The layout for this menu is:

Item #1 = Undo
#2 = Cut
#3 = Copy
#4 = Paste
#5 = Clear
#6 = Show

Undo reverses the last operation performed on the file to let the user correct an error.

Cut removes and saves a marked portion of the display. It moves the data into a GET/PUT buffer for later use by Paste.

Copy saves a marked portion of the display without removing it. It moves the data into a GET/PUT buffer for later use by paste.

Paste inserts previously Cut or Copied data into the screen display and the file.

Clear erases the marked area of the file.

Show displays the contents of the buffer, such as the last text cut or copied.

With the exception of Undo, all edit options should remain disabled until an area of the file is marked for editing by the application. If you want to add new edit options, put them at the end of the file.

Using Application Information Files (AIF)

When GShell reads a disk, it classifies each file into a file-type. The files are listed in the main screen, and each file-type is indicated through an associated icon.

The algorithm for this process is structured like this:

1. If it's a folder, display a folder icon.

2. If it's executable, display an object icon.

3. If it's neither a folder nor executable, then it must be a data file. Check if:

   A. File is named "aif.xxx"(xxx represents three legal filename characters).

   B. File is named "yyyyy.xxx" and ".xxx" matches the extension of an AIF encountered previously on this device.

When neither A nor B is true, then a data icon is displayed. But when either A or B is true, GShell tries to load an icon file that is designated in the third line of the AIF. That file should be a 24x24 pixel 4 color bitmap that will be loaded as a Get/Put buffer. If GShell is unable to load the icon file into a Get/Put buffer, it will display the data file icon.

The path to the icon file will be assumed to begin in the current execution folder unless a full path is given. For example, if the current execution folder is /D0/CMDS and the AIF contains "ICONS/icon.scred," then GShell attempts to load "/D0/CMDS/ICONS/icon.scred." (The file must have its execution attribute set for it to be recognized.) The 24x24 pixel bitmap will then be displayed as the icon for the file.

A folder can contain multiple AIF's and corresponding data files. Once an AIF has been recognized by GShell, all corresponding data files subsequently processed will be identified with it. They need to be on the same device, yet the folder that they're in dosen't matter.

An Application Information File provides the following information to GShell:

1. Name of application program
2. Parameters
3. Name (or path) of icon file
4. Memory modifier (0=use default) specified in pages
5. Screen type
6. Defined window width (in decimal)
7. Defined window length (in decimal)
8. Background palette register
9. Foreground palette register

For example, an AIF for Scred can be set up as follows:

Scred	Application's name (executable program)
	Parameters (blank line if no parameters)
ICONS/icon.scred	Name of icon file for GShell to display
0	Use Default Memory
7	Screen type
80	Screen width
24	Screen Length
3	Background Palette Register
2	Foreground

These parameters set up an AIF with Scred as the application. GShell will use an 80x24 graphics screen for Scred with black characters on a green background (assuming standard palettes are used).

The following is a dump of "/DD/SYS/ICON/icon.scred":

Addr	0 1	2 3	4 5	6 7	8 9	A B	C D	E F
0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0555	0000
0020	0000	1004	0000	0000	4010	0000	0001	0040
0030	0000	0004	0100	0000	0010	0400	0000	0040
0040	1000	0000	0100	4000	0000	0401	0000	0000
0050	1004	0000	0000	4010	0000	0001	0040	0000
0060	0004	0100	0000	0010	0400	0000	0055	5000
0070	0000	0055	0000	0000	0050	0000	0000	0040
0080	0000	0000	0000	0000	0000	0000	0000	0000

```
0   2   4   6   8   A   C   E
. . . . . . . . . . . . . . .
. . . . . . . . . . . . U . .
. . . . . . . @ . . . . . . @
. . . . . . . . . . . . . . @
. . . . . @ . . . . . . . . .
. . . . . @ . . . . . @ . . .
. . . . . . . . . . . U P . .
. . . U . . . . P . . . . . @
. . . . . . . . . . . . . . .
```

If the user double-clicks on a directory, then GShell will open that folder and any AIF files will be displayed with the icon and the name of the application. Clicking on the AIF causes GShell to fork that application with no data file as an parameter. Double-clicking on a data file that has the same suffix as an AIF file, or clicking once on a data file and then selecting OPEN in the files menu will cause GShell to fork the application with the name of that data file as an arguement. The data file does not need to be in the same directory as the AIF, but GShell must process the AIF before it processes the data file, if the two are to be associatedwith each other.

Setting 16 Colors

From the Control Panel, you can adjust the four color palettes that you use with Multi-Vue. If you would like to set all 16 color palettes, then you'll need to do this by calling Control as a stand-alone program on a 16-color screen.

The following procedure will open a 16-color graphics window by calling control:

1. Reboot the system using the OS-9 Level II disk.

2. OS9: **merge sys/stdfonts >/w**

3. OS9: **iniz w2**

4. OS9: **display 1b 20 8 0 0 28 18 1 0 0 >/w2**

5. OS9: **shell i=w2&**

6. Press [CLEAR] and you should see a white screen with "OS9:" in blue letters. While that screen is displayed, place Multi-Vue Diskette #1 in Drive 0.

7. OS9: **chx /d0/cmds**

8. OS9: **chd /d0**

9. OS9: **control**

The control panel will appear with the 16 blocks at the bottom displaying the 16 colors currently in the color palette.

An alternate approach can be taken if you have 512 K. Create a file called AIF.CON that looks like this:

```
CONTROL
(blank line)
ICONS/icon.demo
0
8
40
24
0
1
```

This AIF will start the control program on a 16-color, 40x24 screen with blue foreground on a white background. For simplicity it uses the icon for the Demo program. You can design your own icon instead.

This application will not have any data files associated with it. Clicking on it will start control in a full 16-color screen.

The Clipboard Function

When you worked with the desktop functions, you might have noticed that an extra function, clipboard, was inaccessable. This function is intended to be used to transfer data between files or applications. It will be implemented in future applications that require this type of transfer. This is presented now to promote uniformity and compatibility

The menu that will appear when clipboard is selected includes:

Undo
Cut
Copy
Paste
Clear
Show

The Environment File Layout

System-wide parameters are used by the current OS-9 device drivers to form the environment. Here are some important values:

Variable	Values	Function	Set By
MONTYPE	0=Composite 1=RGB 2= Monochrome	Determines monitor type	Control panel
RAM	128 512	Determines RAM Size	Text Editor
REPSPD	1= slow to 5= fast	Key Repeat Start	Control Panel
REPSTR	1= no key repeat 2= long delay 3= short delay	Key Repeat Start	Control Panel
PTRRES	0=lo res 1=hi res	Mouse Resolution	Control Panel

Variable	Values	Function	Set By
PTRSID	0=left side 1=right side	Determines port for mouse/ joystick	Control Panel
EXEC	<pathlist>	Default Execution	Text Editor
Palet0	0-3, 0-3, 0-3	Palette Colors	Control Panel
through			
Palet15	0-3, 0-3, 0-3		
RBFDEV	/d0, /d1, .	RBF devices supported	Text Editor
SCFDEV	/p, /t1, ...	SCF devices supported	Text Editor
PRNAME	name	Identify type of printer	Printer
PRPORT	/p	Device for printer	Printer
PGLEN	0-255	Printer page length	Printer
PGWDTH	0-255	Printer page width	Printer
LFTMRGN	0-255	Left margin	Printer
LNLEN	0-255	Printer Line Length	Printer
HDRSIZ	0-255	Header size	Printer
TRLSIZ	0-255	Trailer size	Printer
PGPAUS	0-255	Pages until pause	Printer

The Environment File

The following is a copy of the environment file with its original settings. Note that lines beginning with an asterisk are comments, and they are ignored by programs that read the env.file:

```
RBFDEV= /d0, /d1
SCFDEV= /p, /t1
MONTYPE=1
RAM=128
*RAM=512
EXEC= /d0/CMDS
*PROGRAM= Shell
*PARAM= i= /term
REPSTR= 3
REPSPD= 4
PTRRES= 0
PTRSID= 1
LFTMRGN= 5
LNLEN= 70
PGWDTH= 80
HDRSIZ= 5
TRLSIZ= 5
PGLEN= 66
TABSIZ= 4
PGPAUS= 0
PRPORT= /p
PRNAME= DMP100
PALET0= 3,3,3
PALET1= 1,0,2
PALET2= 0,0,0
PALET3= 0,3,0
PALET4= 3,0,2
PALET5= 0,0,0
```

```
PALET6= 3,2,3
PALET7= 0,2,0
PALET8= 3,2,1
PALET9= 0,0,3
PALET10= 0,1,0
PALET11= 2,3,0
PALET12= 3,1,1
PALET13= 3,3,0
PALET14= 2,0,3
PALET15= 0,3,3
```

C Language Graphics Library Support

Chapter 10

C language support of the windowing/graphics primitives and high-level commands for the Color Computer 3 is provided by a linkable library named CGFX. This library contains routines that you can access to perform the functions outlined in the *OS-9 Level Two Windowing* manual when programming in C. It also contains the extensions for the Multi-Vue environment.

In addition to CGFX, header files also exist for C. These files contain all the definitions for the high-level window environment and structure definitions for the required data structures.

To use CGFX functions, you must link the CGFX library--along with the other required libraries--to your C program. If you are going to use other high-level window commands, you must include the Wind.h header file in your program. If you are going to use the mouse, you must include Mouse.h. To access standard patterns, fonts, or pointers, include the Buffs.h and Stdmenu.h files. See the codes for the header files later in this chapter.

Compile the C program to a .r file using the -r option on your compiler command line. Then link the program. The lines that you use to create the object file are as follows:

```
OS9:   cc1 prog.c -r
OS9:   c.link /d1/lib/cstart.r prog.r -l=/d1/lib/cgfx.l
-l=/d1/lib/clib.l -l=/d1/lib/sys.l -o=prog
```

These lines link the program with the C graphics library, the standard C library, and the OS-9 C system library. The output file, Prog, is now fully executable and able to perform all the graphics/window functions.

The following sections describe the functions included in the CGFX library. The remainder of the chapter includes:

A. Standard Windowing Commands
B. Get/Put Commands
C. Configuration Comands
D. Font Handling Commands
E. Standard Text Commands
F. Drawing Commands
G. General System Status Functions
H. Mouse Handling Functions
I. Alarm Functions
J. Multi-Vue Windowing Functions

As well, the section "Writing Applications for Multi-Vue in C" provides specific information on producing programs to operate in the Multi-Vue environment.

All the calls explained in this chapter return a value of -1 on error. The system places the error number in the global parameter *errno*.

A. Standard Windowing Commands

The following commands define, alter, or deallocate device and overlay windows.

To create a device window, first open a path to it using OPEN, and then define it, using DWSet. The path to a device window must be an OS-9 path number, and not a C file pointer. Therefore, use open(), not fopen(), for all device window paths.

Overlay windows do not require opening. They are defined in full when you use the OWSet call.

DWSet
Device Window Set

Usage:

```
DWSet(path,sty,cpx,cpy,szx,szy,fprn,bprn,bdprn)
int path,sty,cpx,cpy,szx,szy,fprn,bprn,bdprn;

DWSet(path,sty=0,cpx,cpy,szx,szy,fprn,bprn)
int path,sty,cpx,cpy,szx,szy,fprn,bprn;
```

Description:

Creates a device in a window of type *sty*.

If *sty*=0, the system opens the window on the current screen. If *sty*=0, do not include a border palette (*bdprn*).

The window has its upper left corner located at *cpx,cpy*, and its size is set to *szx,szy*. Note that the coordinates and size values are in standard character cell (8x8) coordinates.

The window uses *fprn* as the foreground palette and *bprn* as the background palette. If you are using a new screen (*sty* is not equal to 0), the border palette is *bdprn*.

Parameters are:

path = OS-9 path number for the window
sty = window type
cpx = horizontal character position for upper left corner of the window
cpy = vertical character position for upper left corner of the window
szx = size (width) of the window (in character positions)
szy = size (height) of the window (in character positions)
fprn = foreground palette number
bprn = background palette number
bdprn = border palette number

DWEnd
Device Window End

Usage:

```
DWEnd(path)
int path;
```

Description:

Deallocates the device window associated with the specified path. If the deallocated window is the last device window on the screen, DWEnd deallocates the screen.

The parameter is:

path = OS-9 path number for the window

OWSet
Overlay Window Set

Usage:

```
OWSet (path,svs,cpx,cpy,szx,szy,fprn,bprn)
int path,svs,cpx,cpy,szx,szy,fprn,bprn;
```

Description:

Creates an overlay window of size *szx,szy* at location *cpx,cpy* on the current device window.

If the save switch (*svs*) is 0, the system does not save the area under the overlay window. If *svs* is 1, the system saves the area under the window if possible and restores it when OWEnd is called.

The parameters are:

path	= OS-9 path number for the window
svs	= save switch value
cpx	= horizontal character position for the upper left corner of the window
cpy	= vertical character position for the upper right corner of the window
szx	= size (width) of the window (in character positions)
szy	= size (height) of the window (in character positions)
fprn	= foreground palette number
bprn	= background palette number

OWEnd
Overlay Window End

Usage:

```
OWEnd(path)   int path;
```

Description:

Deallocates the top overlay window. If you created the window with a save switch value of 1, the area under the screen is restored.

The parameter is:

path = OS-9 path number for the window

Select
Select

Usage:

```
Select(path)
int path;
```

Description:

Displays the window associated with the specified path.

The parameter is:

path = OS-9 path number for the window

CWArea
Change Working Area

Usage:

```
CWArea(path,cpx,cpy,szx,szy)
int path,cpx,cpy,szx,szy;
```

Description:

Changes the working area of a window to the new location given by *cpx, cpy,* which is an offset from the position of the original window area. The size is *sxz,szy.*

This call cannot make a window larger than originally defined; it can only change the working area inside the window.

The parameters are:

path	= OS-9 path number for the window
cpx	= horizontal character position of the working area's upper left corner
cpy	= vertical character position of the upper left corner for the working area
szx	= size (width) of the working area in character positions
szy	= size (height) of the working area in character positions

DWProtSw
Device Window Protect Switch

Usage:

```
DWProtSW(path,bsw)
int path,bsw;
```

Description:

Sets device window protection. Normally a window is protected (*bws*=1). Setting *bws* to 0 unprotects the window.

The parameters are:

path = OS-9 path number for the window
bws = protect switch value

B. Get/Put Commands

Get/Put buffers provide a means of moving and storing data. Use the Get/Put commands for the allocation and deallocation of these buffers and for transferring data.

DfnGPBuf
Define Get/Put Buffer

Usage:

```
DfnGPBuf(path,grpnum,bufnum,length)
int path,grpnum,bufnum,length;
```

Description:

Defines and allocates storage for a get/put buffer by specifying the buffer's group, number, and size.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number associated with the buffer
bufnum	= buffer number
length	= buffer size

KilBuf
Deallocate Get/Put Buffer

Usage:

```
KilBuf(path,grpnum,bufnum)
int path,grpnum,bufnum;
```

Description:

Deallocates the specified get/put buffer. To deallocate an entire group of buffers, set *bufnum* to 0.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number associated with the buffer
bufnum	= buffer number

GPLoad
Get/Put Buffer Pre-Load

Usage:

```
GPLoad(path,grpnum,bufnum,sty,sizex,sizey,length)
int path,grpnum,bufnum,sty,sizex,sizey,length;
```

Description:

Prepares to load a get/put buffer with data. After receiving a GPLoad() call, the system loads the next bytes written to that path into the specified get/put buffer.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number associated with the buffer
bufnum	= buffer number
sty	= screen type for the data
sizex	= horizontal size for the data
sizey	= vertical size for the data
length	= size of the buffer in bytes

GetBlk
Get Block

Usage:

```
GetBlk(path,grpnum,bufnum,x,y,sizex,sizey)
int path,grpnum,bufnum,x,y,sizex,sizey;
```

Description:

Copies a block of screen data from *x,y* to *x+sizex,y+sizey*. Stores the data in the buffer specified by *grpnum,bufnum*.

Parameters are:

path	= The OS-9 path number for the window
grpnum	= group number associated with the buffer
bufnum	= buffer number
x	= beginning x-coordinate of the screen portion to save
y	= beginning y-coordinate of the screen portion to save
sizex	= added to *x*, gives the ending x-coordinate of the screen portion to save
sizey	= added to *y*, gives the ending y-coordinate of the screen portion to save

PutBlk
Put Block

Usage:

```
PutBlk(path,grpnum,bufnum,x,y)
int path,grpnum,bufnum,x,y;
```

Description:

Puts a block of data from the buffer specified by *grpnum,bufnum* onto the screen at location *x,y*.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number associated with the buffer
bufnum	= buffer number
x	= x-coordinate location for gpbuffer on screen
y	= y-coordinate location for gpbuffer on screen

C. Configuration Commands

Use configuration commands to set up the patterns, logic and colors to be used by the drawing commands in a window.

PSet
Pattern Set

Usage:

```
PSet(path,grpnum,bufnum)
int path,grpnum,bufnum;
```

Description:

Specifies the group and buffer numbers that provide the buffer pattern to be used for fill and draw functions.

Parameters are:

path	= OS-9 path number for the window device
grpnum	= group number associated with the buffer
bufnum	= buffer number

LSet
Logic Set

Usage:

```
LSet(path,logiccode)
int path,logiccode;
```

Description:

Defines the logic to be used in all draw commands used on the specified window.

Parameters are:

path = OS-9 path number for the window
logiccode = one of the following values:
0 = no logic
1 = AND logic
2 = OR logic
3 = XOR logic

DefColr
Default Color Set

Usage:

```
DefColr(path)
int path;
```

Description:

Sets the colors associated with the specified window to the default colors.

The parameter is:

path = OS-9 path number to the window

Palette
Palette Set

Usage:

```
Palette(path,prn,colno)
int path,prn,colno;
```

Description:

Sets the palette register number defined by prn to contain the color defined by colno.

Parameters are:

path	= OS-9 path number to thewindow
prn	= palette register number
colno	= color number to install

FColor
Foreground Color Set

Usage:

```
FColor(path,prn)
int path,prn;
```

Description:

Specifies the palette register that contains the color to be used for foreground drawing.

Parameters are:

path = OS-9 path number for the window
prn = palette register number

BColor
Background Color Set

Usage:

```
BColor(path,prn)
int path,prn;
```

Description:

Specifies the palette register that contains the color to be used for background drawing.

Parameters are:

path = OS-9 path number for the window
prn = palette register number

Border
Border Color Set

Usage:

```
Border(path,prn)
int path,prn;
```

Description:

Specifies the palette register that contains the color to be used for the screen border.

Parameters are:

path = OS-9 path number for the window
prn = palette register number

ScaleSw
Scaling On/Off Switch

Usage:

```
ScaleSw(path,bsw)
int path,bsw;
```

Description:

Turns scaling on and off for a window.

Parameters are:

path = OS-9 path number for the window
bsw = 0 for off or 1 for on

SetGC
Set Graphics Cursor

Usage:

```
#include <buffs.h>

SetGc(path,grpnum,bufnum)
int path,grpnum,bufnum;
```

Description:

Select the Get/Put buffer to be used for the graphics cursor. Standard pointer shape group and buffer numbers are defined in the Buffs.h header file. The default pointer is an arrow, defined by GRP_PTR and PTR_ARR.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number of the Get/Put buffer holding the cursor image
bufnum	= buffer number

D. Font Handling Commands

The following functions set up the fonts and attributes for software generated text on graphics screens. In addition to the boldface and proportional spacing attributes, discussed here, you can also user reverse video and underlining, discussed in "Standard Text Commands."

Font
Font Selection

Usage:

```
#include <buffs.h>

Font(path,grpnum,bufnum)
int path,grpnum,bufnum;
```

Description:

Specifies the get/put buffer from which to get font data for generating graphics text. The OS-9 system comes with three standard fonts: an 8x8 font, a 6x8 font, and an 8x8 graphics font. The standard fonts are defined in the Buffs.h header file. The default is GRP_FNT,FNT_S8X8.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number of the buffer
bufnum	= buffer number

TCharSw
Transparent Character Switch

Usage:

```
TCharSw(path,bsw)
int path,bsw;
```

Description:

Turns the character transparency mode on and off.

Parameters are:

path = OS-9 path number for the window device
bsw = one of the following values:
0 = draw characters in an opaque character cell
1 = draw characters in a transparent character cell

BoldSw
BoldFace Attribute Switch

Usage:

```
BoldSw(path,bsw)
int path,bsw;
```

Description:

Turns boldface on and off for the characters.

Parameters are:

path = OS-9 path number for the window device
bsw = one of the following values:
0 = display characters in normal type face
1 = display characters in boldface type

PropSw
Proportional Spacing Attribute Switch

Usage:

```
PropSw(path,bsw)
int path,bsw;
```

Description:

Turns proportional spacing on and off.

path = OS-9 path number for the window device
bsw = one of the following values:
0 = turn off proportional spacing
1 = turn on proportional spacing

E. Standard Text Commands

The following commands let you manipulate the text cursor and text attributes in both standard hardware-generated text and graphics-generated text. The only functions not supported on graphics screens are the blinking text functions.

The functions in this section are self-explanatory and require only a path parameter.

CurHome
Home Cursor

Usage:

```
CurHome(path)
int path;
```

CurOn
Turn Cursor On

Usage:

```
CurOn(path)
int path;
```

CurOff
Turn Cursor Off

Usage:

```
CurOff(path)
int path;
```

CurLft
Move Cursor Left

Usage:

```
CurLft(path)
int path;
```

CurRgt
Move Cursor Right

Usage:

```
CurRgt(path)
int path;
```

CurUp
Move Cursor Up

Usage:

```
CurUp(path)
int path;
```

CurDwn
Move Cursor Down

Usage:

```
CurDwn(path)
int path;
```

CurXY
Move Cursor TO X,Y

Usage:

```
CurXY (path,x,y)
int path,x,y;
```

Erline
Erase Line

Usage:

```
Erline(path)
int path;
```

ErEOLine
Erase to End Of Line

Usage:

```
ErEOLine(path)
int path;
```

ErEOScrn
Erase to End Of Screen

Usage:

```
ErEOScrn(path)
int path;
```

Clear
Clear the Screen

Usage:

```
Clear(path)
int path;
```

Bell
Ring Bell

Usage:

```
Bell(path)
int path;
```

CrRtn
Output Carriage Return

Usage:

```
CrRtn(path)
int path;
```

ReVOn
Turn On Reverse Video

Usage:

```
ReVOn(path)
int path;
```

ReVOff
Turn Off Reverse Video

Usage:

```
ReVOff(path)
int path;
```

UndlnOn
Turn On Underline

Usage:

```
UndlnOn(path)
int path;
```

UndInOff
Turn Off Underline

Usage:

```
UndInOff(path)
    int path;
```

BlnkOn
Turn On Blink (hardware text only)

Usage:

```
    BlnkOn(path)
    int path;
```

BlnkOff
Turn Off Blink (hardware text only)

Usage:

```
    BlnkOff(path)
    int path;
```

InsLin
Insert Line

Usage:

```
    InsLin(path)
    int path;
```

DelLine
Delete Line

Usage:

```
DelLine(path)
int path;
```

F. Drawing Commands

These comands draw graphics primitives in the window specified by the parameter *path.*

By default, the system draws all graphics primitives in relation to the current position of the draw pointer, and you can set the draw pointer to any position in a window. Also by default, all drawing is scaled on a 640x192 coordinate grid.

The ScaleSw() function turns scaling off and on. The system uses the upper left corner of the window as its reference point.

The system clips all drawings at the edges of the window's working area. It produces all images in the foreground color set by FColor() and uses the pattern and logic set by PSet() and LSet(). The default setting for pattern and logic is off.

SetDPtr
Set the Draw Pointer Position

Usage:

```
SetDPtr(path,x,y)
int path,x,y;
```

Description:

Positions the draw pointer at position *x,y*, in relation to the upper left corner of the window.

Parameters are:

path	= OS-9 path number for the window
x	= x-coordinate position for the draw pointer
y	= y-coordinate position for the draw pointer

RSetDPtr
Set the Draw Pointer Relative to the Current Position

Usage:

```
RSetDPtr(path,xo,yo)
int path,xo,yo;
```

Description:

Positions the draw pointer at offset *xo,yo* from the draw pointer's current position.

Parameters are:

path = OS-9 path number for the window
xo = x-coordinate offset from the draw pointer's position
yo = y-coordinate offset from the draw pointer's position

Point
Draw a Point

Usage:

```
Point(path)
int path;
```

Description:

Draws a point at the draw pointer's current position.

The parameter is:

path = OS-9 path number for the window

RPoint
Draw a Point Relative to the Draw Pointer

Usage:

```
RPoint(path,xo,yo)
int path,xo,yo;
```

Description:

Draws a point at offset *xo,yo* from the draw pointer's current position.

Parameters are:

path	= OS-9 path number for the window
xo	= x-coordinate offset from the draw pointer's current position
yo	= y-coordinate offset from the draw pointer's current position

Line
Draw a Line

Usage:

```
Line(path,x,y)
int path,x,y;
```

Description:

Draws a line from the draw pointer to *x,y*. This function does not update the draw pointer.

Parameters are:

path	= OS-9 path number for the window
x	= x-coordinate for the end of the line
y	= y-coordinate for the end of the line

RLine
Draw a Line Relative to the Draw Pointer

Usage:

```
RLine(path,xo,yo)
int path,xo,yo;
```

Description:

Draws a line from the draw pointer to offset *xo,yo*. This function does not update the draw pointer.

Parameters are:

path	= OS-9 path number for the window
xo	= x-coordinate offset from the draw pointer's current position
yo	= y-coordinate offset from the draw pointer's current position

LineM
Draw a Line and Update the Draw Pointer

Usage:

```
LineM(path,x,y)
int path,x,y;
```

Description:

Draws a line from the draw pointer to *x,y* and updates the draw pointer to the new position.

Parameters are:

path	= OS-9 path number for the window
x	= x-coordinate for the end of the line
y	= y-coordinate for the end of the line

RLineM
Draw a Relative Line and Update the Draw Pointer

Usage:

```
RLineM(path,xo,yo)
int path,xo,yo;
```

Description:

Draws a line from the draw pointer to offset *xo,yo* and updates the draw pointer to the new location.

Parameters are:

path = OS-9 path number for the window
xo = x-coordinate offset from the draw pointer's current position
yo = y-coordinate offset from the draw pointer's current position

Box
Draw a Box

Usage:

```
Box(path,x,y)
int path,x,y;
```

Description:

Draws a rectangle with one corner at the draw pointer's current position and the opposite corner at *x,y*. This function does not update the draw pointer.

Parameters are:

path = OS-9 path number for the window
x = x-coordinate for the corner opposite the draw pointer's current position
y = y-coordinate for the corner opposite the draw pointer's current position

RBox
Draw a Box Relative to the Draw Pointer

Usage:

```
RBox(path,xo,yo)
int path,xo,yo;
```

Description:

Draws a box with one corner at the draw pointer's current position and the opposite corner at offset *xo,yo*. This function does not update the draw pointer.

Parameters are:

path	= OS-9 path number for the window
xo	= x-coordinate offset from the draw pointer's current position
yo	= y-coordinate offset from the draw pointer's current position

Bar
Draw a Bar

Usage:

```
Bar(path,x,y) int path,x,y;
```

Description:

Draws a filled rectangle with one corner at the draw pointer's current position and the opposite corner at *x,y*. This function does not update the draw pointer.

Parameters are:

path = OS-9 path number for the window
x = x-coordinate for the corner opposite the draw pointer's current position
y = y-coordinate for bar corner opposite the draw pointer's current position

RBar
Draw a Bar Relative to DP

Usage:

```
RBar(path,xo,yo)
int path,xo,yo;
```

Description:

Draws a bar with one corner at the draw pointer's current position and the opposite corner at offset *xo,yo*. This function does not update the draw pointer.

Parameters are:

path = OS-9 path number for the window
xo = x-coordinate offset from the draw pointer's current position
yo = y-coordinate offset from the draw pointer's current position

FFill
Flood Fill Area

Usage:

```
FFill(path)
int path;
```

Description:

Sets the pixels surrounding the draw pointer to the foreground color. The fill operation continues outward until it reaches either the edge of the screen or pixels that are a color other than the pixel at the draw pointer.

The parameter is:

path = OS-9 path number for the window

Circle
Draw a Circle

Usage:

```
Circle(path,radius)
int path, radius;
```

Description:

Draws a circle with the draw pointer position as the center.

Parameters are:

path = OS-9 path number for the window
radius = radius of the circle, based on the x axis

Ellipse
Draw an Ellipse

Usage:

```
Ellipse(path,xrad,yrad)
int path,xrad,yrad;
```

Description:

Draws an ellipse with an x-radius of *xrad* and a y radius of *yrad*. The center is located at the draw pointer.

Parameters are:

path = OS-9 path number for the window
xrad = width of the ellipse
yrad = height of the ellipse

Arc
Draw an Arc

Usage:

```
Arc(path,xrad,yrad,xo1,yo1,xo2,yo2)
int path,xrad,yrad,xo1,yo1,xo2,yo2;
```

Description:

Draws an elliptical or circular arc. The center of the arc is at the draw pointer. The offsets *xo1,yo1,xo2,* and*yo2* define a clipping line for the arc.

Parameters are:

path = OS-9 path number for the window
xrad = width of the arc
yrad = height of the arc
xo1,yo1 = beginning and ending coordinates of a line that
xo2,yo2 dissects the arc and clips it

PutGC
Put the Graphics Cursor

Usage:

```
PutGC(path,x,y)
int path,x,y;
```

Description:

Positions the graphics cursor at *x,y*. If you are using an auto-follow mouse, do not call this function. See _ss_mous().

Parameters are:

path	= OS-9 path number for the window
x	= x-coordinate for the cursor position
y	= y-coordinate for the cursor position

G. General System Status Functions

The following functions correspond to the GetStat and SetStat calls under OS-9. Use them to read or set the status of system devices and global parameters. See the Technical Reference

_gs_opt
Get the Path Options

Usage:

```
#include <sgstat.h>

_gs_opt(path,buffer)
 int path; struct sgbuf *buffer;
```

Description:

Gets the options for the specified *path* and places them in the area pointed to by the specified *buffer*.

_ss_opt
Set the Path Options

Usage:

```
#include <sgstat.h>

_ss_opt(path,buffer)
int path; struct sgbuf *buffer;
```

Description:

Uses the options from the specified *buffer* to set the options for the specified *path*.

_gs_size
Get the File Size for RBF Devices

Usage:

```
long _gs_size(path)
int path;
```

Description:

Returns the size of the file specified by *path*.

_ss_size
Set the File Size for RBF Devices

Usage:

```
_ss_size(path,size)
int path; long size;
```

Description:

Sets the file *size* to the specified number of bytes.

_gs_rdy
Check for Characters Ready on Input

Usage:

```
int _gs_rdy(path)
int path;
```

Description:

Checks for available data on the input path. If data is available, the function returns the number of bytes; otherwise, it returns a value of -1.

_ss_ssig
Set up for Signal on Data Ready

Usage:

```
_ss_ssig(path,signo)
int path,signo;
```

Description:

Tells the system to send a signal when data is available on a device. *signo* is the number of the signal to send.

_ss_rel
Release a Device from a Send Signal Request

Usage:

```
_ss_rel(path) int path;
```

Description:

Releases the process specified by *path* from a previous _ss_ssig() call or from a _ss_msig() mouse signal setup call.

_gs_scsiz
Get the Size of the Window

Usage:

```
_gs_scsiz(path,horsiz,versiz)
int path,*horsiz,*versiz;
```

Description:

Sets the horizontal and vertical sizes of a screen window.

Parameters are:

path = OS-9 path number for the window
horsiz = horizontal size
versiz = vertical size

_gs_palt
Get Palette Information

Usage:

```
_gs_palt(path,palbuf)
int path; char *palbuf;
```

Description:

Returns the values for the 16 palette registers in the character buffer pointed to by *palbuf*.

_ss_mgpb
Map a Get/Put Buffer Into User Address Space

Usage:

```
char *_ss_mgpb(path,grpnum,bufnum)
int path,grpnum,bufnum;
```

Description:

Maps the requested get/put buffer into the user address space and returns a pointer to the buffer. If the function does not successfully map the buffer, it returns a null value.

Parameters are:

path	= OS-9 path number for the window
grpnum	= group number of the buffer
bufnum	= buffer number

_gs_styp
Get Screen Type

Usage:

```
_gs_styp(path,type)
int path,*type;
```

Description:

Returns in the parameter *type* a value indicating the type of screen associated with *path*.

_gs_fbrg
Get Foreground/Background Colors

Usage:

```
_gs_fbrg(path,fore,back,bord)
int path,*fore,*back,*bord;
```

Description:

Gets the palette registers used for foreground, background, and border colors and returns them in the specified parameters.

The parameters are:

path = OS-9 path number for the window
fore = the parameter for the foreground color
back = the parameter for the background color
bord = the parameter for the border color

_ss_gip
Set the Global Information Parameters

Usage:

```
_ss_gip(path,msres,msport,kbdstrt,kbdrpt)
int path,msres,msport,kbdstrt,kbdrpt;
```

Description:

Sets the global information parameters for the system, including the mouse resolution, the active mouse port, the keyboard repeat speed, and the keyboard repeat start count.

Parameters are:

path = window device associated with the changes
msres = mouse resolution setting (0= lo res; 1= hi res)
msport = mouse port to be active (1= right side; 2= left)
kbdstrt = keyboard repeat start count (range= 0-255; actual useful values= 0-50)
kbdrpt = keyboard repeat speed (range= 0-255; actual useful values= 0-15)

_ss_dfpl
Set the Default Palette Information

Usage:

```
_ss_dfpl(path,palbuf)
int path; char *palbuf;
```

Descripton:

Sets the default palette register values to be those passed in *palbuf*.

_ss_mtyp
Set the Monitor Type

Usage:

```
_ss_mtyp(path,montype)
int path,montype;
```

Description:

Establishes the monitor type for the system to the type specified by *montype*.

Montype can be:
0 = composite monitor or television
1 = analog RGB monitor
2 = monochrome composite monitor

H. Mouse Handling Functions

The following functions set up and read the mouse/joystick information, allowing interactive programs to use mouse input instead of keyboard input. The structures required for mouse information are defined in the mouse.h header file, which you should include in your program.

_ss_mous
Set the Mouse Status

Usage:

```
_ss_mous(path,sample_rate,timeout,follow)
int path,sample_rate,timeout,follow;
```

Description:

Sets the mouse parameters as required by an application.

Parameters are:

path	= OS-9 path number for the window
sample_rate	= rate, in ticks, at which you want the mouse read(*sample* =0 only when getstat is called)
timeout	= time, in ticks, for the buttons to timeout
follow	= a flag that tells whether the system automatically reads the mouse position and updates the graphics cursor (follow=1), or whether the system won't follow mouse values (follow =0)

_ss_msig
Send a Signal on Mouse Click

Usage:

```
_ss_msig(path,signo)
int,path,signo;
```

Description:

Causes the system to send a signal to the process when a mouse button click occurs.

Parameters are:

path = OS-9 path number for the window
signo = signal number to send

_gs_mous
Get the Mouse Status

Usage:

```
#include <mouse.h>

_gs_mous(path,*mspacket)
int path; MSRET mspacket;
```

Description:

Returns the mouse's location, button status, and so on, as defined in the MSRET structure in Mouse.h.

I. Alarm Functions

The following functions access the alarm capabilities of the OS-9 Level Two system. You can set one alarm event in the system at a time. The bell (a beep through the TV or monitor speaker) sounds for the first 15 seconds of the minute for the set time.

SetAlarm
Set the Alarm Time

Usage:

```
#include <time.h>
struct sgtbuf *timbuf; SetAlarm(timbuf)
```

Description:

Sets the system alarm to sound at the time stored in the buffer pointed to by*timbuf*.

GetAlarm
Get the Alarm Time

Usage:

```
#include <time.h>
struct sgtbuf *timbuf; GetAlarm(timbuf)
```

Description:

Returns the alarm time and date in the buffer pointed to by *timebuf*.

ClrAlarm
Clear the Alarm

Usage:

```
ClrAlarm()
```

Description:

Turns off the system alarm and clears it.

J. Multi-Vue Windowing Functions

You can use the following functions with the WindInt interface module instead of the standard GrfInt interface module.

GrfInt lets you establish windows without borders or windows for which you want to design your own borders. However, Multi-Vue uses the WindInt module to support framed windows, menu bars, pull-down menus, and scroll bars. The structures required for these functions are defined in the Wind.h header file. You should include this header file in your program. The StdMenu.h file defines standard system menus and you should include it in application programs running under Multi-Vue.

_ss_wset
Set the Type of Window

Usage:

```
_ss_wset (path, wintype, windat)
int path, wintype;
wndscr*windat;
```

Description:

Sets the type of frame that a window is to have.

Windint provides six types of windows. These are defined in Wind.h as WT_'s. For framed windows (WT_FWIN and WT_FSWIN), you must pass a pointer to the window data structure using _ss_wset(). This establishes the menu and title bar for the window. For other window types, set this pointer to NULL.

The types of windows and their uses are:

WT_FWIN	Framed window for applications.
WT_FSWIN	Framed window with scroll bars for applications.
WT_SBOX	Shadowed box for pull-down menus.
WT_DBOX	Double box for dialog/warning messages.
WT_PBOX	Plain box for general application use.
WT_NBOX	No box window. This is a standard GrfInt window type.

_gs_msel
Get Menu Selection

Usage:

```
int _gs_msel(path,*itemno)
int path,itemno;
```

Description:

Call this function when the mouse is in the control region of a window. The function automatically checks to determine whether the mouse is on a menu bar item. If it is, the funciton handles the pull-down menu routine.

If the menu selection is valid, the function returns the menu id and sets the *itemno* parameter to the item selected within the menu. If the selection is invalid, or if the user does not make a menu selection, the function returns 0.

_ss_umbar
Update the Menu Bar

Usage:

```
_ss_umbar(path)
int path;
```

Description:

Causes the system to reread the menu bar information from the user space and redraws the menu bar accordingly. Use this call to enable or disable menus and menu items.

_ss_sbar
Set Scroll Marker Positioning

Usage:

```
_ss_sbar(path,horbar,verbar)
int path,horbar,verbar;
```

Description:

Positions the scroll markers on a WT_FSWIN window. This function expects the positions of the horizontal and vertical markers to be absolute character coordinates from either the far left scroll region or from the top scroll region.

Parameters are:

path	= OS-9 path number for the window
horbar	= horizontal scroll marker position
verbar	= vertical scroll marker position

Wind.h Code

```
/ *Wind.h - Defs for window data structure */

/* window type defs */
#define WT_NBOX 0
#define WT_FWIN 1
#define WT_FSWIN 2
#define WT_SBOX 3
#define WT_DBOX 4
#define WT_PBOX 5

#define MNENBL 1
#define MNDSBL 0
#define WINSYNC 0xCOCO

/* default menu ids */
#define MN_MOVE 1
#define MN_CLOS 2
#define MN_GROW 3
#define MN_USCRL 4
#define MN_DSCRL 5
#define MN_RSCRL 6
#define MN_LSCRL 7
#define MN_TNDY 20
#define MN_FILE 21
#define MN_EDIT 22
#define MN_STYL 23
#define MN_FONT 24
```

```
/* window - menu data structures */
    typedef struct mistr {  /* menu item descriptor */
    char _mnttl [15];       /* name of item */
    char _mienbl;           /* is item available? */
    char _mires[5];         /* reserved */
} MIDSCR;                   /* menu descriptor */
typedef struct mnstr {
    char _mittl[15];        /* name of menu */
    char _mnid,             /* menu id number */
         _mnxsiz,           /* width of menu */
         _mnnits,           /* # of items in menu */
         _mnenabl,          /* is item available? */
    struct mistr$* _mnitems; /* pointer to menu */
} MNDSCR;                   /* item descriptor array */
typedef struct wnstr {      /* window descriptor */

    char _wnttl[20];        /* title of window */
    char _nmens;            /* # of menus on window */
    char _wxmin,            /* min. window width */
         _wymin;            /* min. window height */
    short _wnsync;          /* synch bytes $COCO */
    char _wnres[7];         /* reserved */
    struct mnstr$* _wnmen; /*pointer to menu descriptor's array */
} WNDSCR;
```

StdMenu.h Code

```
/* StdMenu.h - standard menu definitions */

/* Default Tandy Menu */
#ifndef MAIN
 extern
#endif
MIDSCR _tanitms[]
#ifdef MAIN
 = {
     {{'C','a','l','c'},1},
     {{'C','l','o','c','k'},1},
     {{'C','a','l','e','n','d','a','r'},1},
     {{'C','o','n','t','r','o','l'},1},
     {{'P','r','i','n','t','e','r'},1},
     {{'P','o','r','t'},1},
     {{'H','e','l','p'},1},
     {{'S','h','e','l','l'},1}
     {{'C','l','i','p','b','o','a','r','d'},0},
}
#endif
;
```

```
/* Default Files Menu */
#ifndef MAIN
 extern
#endif
 MIDSCR _filitms[]
#ifdef MAIN
  = {
     {{'N','e','w'},1},
     {{'O','p','e','n'},1},
     {{'S','a','v','e'},1},
     {{'S','a','v','e',' ','A','s','.','.','.'},1},
     {{'A','b','a','n','d','o','n'},1},
     {{'P','r','i','n','t'},1},
     {{'Q','u','i','t'},1}
}
#endif
;

/* Default Edit Menu */
#ifndef MAIN
 extern
#endif
MIDSCR _editms[]
#ifdef MAIN
 = {
     {{'U','n','d','o'},1},
     {{'C','u','t'},1},
     {{'C','o','p','y'},0},
     {{'P','a','s','t','e'},0},
     {{'C','l','e','a','r'},0}
     {{'S','h','o','w'}0}
}
#endif
;
```

```
#define TNDY_MN {'T','a','n','d','y'},MN_TNDY,10,\
    sizeof(_tanitms)/sizeof(_tanitms[0]),1,0,0,_tanitms

#define FILE_MN {'F','i','l','e','s'},MN_FILE,10,\
    sizeof(_filitms)/sizeof(_filitms[0]),1,0,0,_filitms

#define EDIT_MN {'E','d','i','t'},MN_EDIT,6,\
    sizeof(editms)/sizeof(_editms[0]),0,0,0,_editms
```

Mouse.h Code

```
/* Mouse.h -Defs for mouse data structures */

/* structure for reading mouse info packet */

typedef struct rmousin [
    char pt_valid,          /* is info valid ? */
         pt_actv,           /* active side */
         pt_totm,           /* timeout initial value */
         pt_rsrv0[2],       /* reserved */
         pt_ttto,           /* time till timeout */
         pt_tsst[2],        /* time since start counter */
         pt_cbsa,           /* current button state Button A */
         pt_cbsb,           /* current button state Button B */
         pt_ccta,           /* click count Button A */
         pt_cctb,           /* click count Button B */
         pt_ttsa,           /* time this state Button A */
         pt_ttsb,           /* time this state Button B */
         pt_tlsa,           /* time last state Button A */
         pt_tlsb,           /* time last state Button B */
         pt_rsrv1[6],       /*reserved */
         pt_stat,           /* window pointer type location */
         pt_res;            /* resolution */
    int  pt_acx,            /* actual x value */
         pt_acy,            /* actual y value */
         pt_wrx,            /* window relative x value */
         pt_wry;            /* window relative y value */
} MSRET;

/* window regions for mouse */

#define WR_CNTNT 0          /* content region */
#define WR_CNTRL 1          /* control region */
#define WR_OFWIN 2          /* off window */
```

Buff.h Code

```
/* Buff.h - definitions for standard get/put buffer usage */

/* buffer group numbers */
#define GRP_FONT  200
#define GRP_CLIP  201
#define GRP_PTR  202
#define GRP_PAT2  203
#define GRP_PAT4  204
#define GRP_PAT6  205

/* font buffers */
#define FNT_S8X8  1
#define FNT_S6X8  2
#define FNT_G8X8  3

/* mouse pointer buffers */
#define PTR_ARR  1
#define PTR_PEN  2
#define PTR_LCH  3
#define PTR_SLP  4
#define PTR_ILL  5
#define PTR_TXT  6
#define PTR_SCH  7
```

```
/*pattern buffers */
#define PAT_SLD 0
#define PAT_DOT 1
#define PAT_VRT 2
#define PAT_HRZ 3
#define PAT_XHTC 4
#define PAT_LSNT 5
#define PAT_RSNT 6
#define PAT_SDOT 7
#define PAT_BDOT 8
```

You can use the graphics font (Fnt_G8X8) in applications other than Multi-Vue. The graphics font supports the following graphics characters:

Char Code	Character
$C1	right arrow (for scroll bars)
$C2	left arrow
$C3	down arrow
$C4	up arrow
$C5	triple bar (for title bar)
$C6	resize box icon (unused)
$C7	close box icon (for menu bar)
$C8	move box icon (unused)
$C9	vertical box (for scroll markers)
$CA	horizontal box
$CB	hourglass (for Desk Utilities on menu bar)
$CC	triple bar with open right side
$CD	triple bar with open left side

Writing Applications for Multi-Vue in C

Defining and initializing an Application Window:

Use the following structure to define a window for an application that uses menus:

```
/* Window Descriptor */
typedef struct wnstr {
    char _wnttl[20];          /* title of window */
    char _nmens;              /* # of menus on window */
    char _wxmin,              /* min. window width */
         _wymin;              /* min. window height */
    short _wnsync;            /* synch bytes $COCO */
    char _wnres[7];           /* reserved */
    struct mnstr* _wnmen;     /* pointer to menu
                                 descriptors' array */
}WNDSCR;
```

The components of the structure are as follows:

_wnttl — title of the window that is displayed at the top of a framed window when it is not the currently selected active window

_nmens — number of menus on the framed window

_wxmin — minimum horizontal size of the window (in character cells)

_wymin minimum vertical size of the window (in character cells)

_wnsync synchronization flag that OS-9 uses to indicate an active menu-driven window

_wnres reserved

_wnmen pointer to the array of menu descriptors for the window's menus

The declaration and initialization of a window descriptor might look like this:

```
#define WXMIN     40
#define WYMIN     24
#define RESERVED      {}

WNDSCR windat = {
    {'A','p','p','l','i','c','a','t','i','o','n','\0'},
    sizeof menus / sizeof menus[0],
    WXMIN,
    WYMIN,
    WINSYNC,
    RESERVED,
    menus
};
```

Defining and Initializing Menus for a Window

Use the following structure to define a number of menus for a framed window. This is the type of structure that is pointed to by the _wnmen parameter in the window descriptor.

```
/* Menu Descriptor */
typedef struct mnstr {
    char _mnttl[10];          /*name of menu */
    char _mnid,               /* menu ID number */
         _mnxsiz,             /* width of menu */
         _mnnits,             /* number of items in menu */
         _mnenabl,            /* is this menu available ? */
         _mnres[2];           /* reserved */
    struct mistr _mnitems;    /* pointer to array of menu item*/
                              /*descriptors */
} MNDSCR;
```

The components of the structure are:

_mnttl — title of a menu that is displayed at the top of the framed window when the window is active (currently selected)

_mnid — identification number (in the range 128-255) of the menu

_mnxsiz — maximum size of the pull-down menu (the length of the longest menu)

_mnnits — number of items in the menu

_mnenabl — flag that indicates whether or not a menu is currently enabled (whether it can be selected and pulled down)

_mnres reserved

_mnitems pointer to an array of the item descriptors for the menu

The declaration and initialization of a menu descriptor might look like this:

```
#define NULL   0
#define ENABLE  1
#define DISABLE NULL

#define TNDY_MN {'T','a','n','d','y'},\
        MN_TNDY,\
        10,\
        sizeof (_tanitms/sizeof(_tanitms [NULL]),\
        ENABLE,\
        NULL,\
        NULL,\
        _tanitms

#define FILE_MN {'F','i','l','e','s'},\
        MN_FILE,\
        10,\
        sizeof(_filitms)/sizeof(_filitms[NULL),\
        ENABLE,\
        NULL,\
        NULL,\
        _filitms
```

```
#define EDIT_MN {'E','d','i','t'},\
        MN_EDIT,\
        6,\
        sizeof(_editms)/sizeof(_editms[NULL]),\
        NULL,\
        NULL,\
        NULL,\
        _editms

MDNSCR menus [] = {
    {TNDY_MN},
    {FILE_MN},
    {EDIT_MN}
};
```

Defining and Initializing the Items of a Menu

Use the following structure to define a number of items within menus. This is the type of structure that is pointed to by the _mnitems parameter in the menu descriptor.

```
/* Menu Item Descriptor */
typedef struct mistr {
    char _mittl[10];        /* name of item */
    char _mienbl;           /* item enable flag */
    char _mires[5];         /* reserved */

} MIDSCR;
```

The components of the structure are:

_mittl — title of the item that appears in the menu's pull-down window when the menu is selected

_mienbl flag to indicate whether or not the item is currently enabled (whether it can be selected)

_mires reserved

The declaration and initialization of a number of items within a menu might look like this:

```
/* Default Tandy Menu */
MIDSCR _tanitms[] = {
    {{'C','a','l','c'},ENABLE},
    {{'C','l','o','c','k'},ENABLE},
    {{'C','a','l','e','n','d','a','r'},ENABLE},
    {{'C','o','n','t','r','o','l'},ENABLE},
    {{'P','r','i','n','t','e','r'},ENABLE},
    {{'P','o','r','t'},ENABLE},
    {{'H','e','l','p'},ENABLE},
    {{'S','h','e','l','l'},ENABLE},
    {{'C','l','i','p','b','o','a','r','d'},ENABLE}
};

/* Default Files Menu */
MIDSCR _filitms[] = {
    {{'N','e','w'},ENABLE},
    {{'O','p','e','n'},ENABLE},
    {{'S','a','v','e'},ENABLE},
    {{'S','a','v','e',' ','A','s','.','.','.'},ENABLE},
    {{'A','b','a','n','d','o','n'},ENABLE},
    {{'P','r','i','n','t'},ENABLE},
    {{'Q','u','i','t'},ENABLE},
};
```

```
/* Default Edit Menu */
MIDSCR _editms[] = {
    {{'U','n','d','o'},ENABLE},
    {{'C','u','t'},ENABLE},
    {{'C','o','p','y'},ENABLE},
    {{'P','a','s','t','e'},ENABLE},
    {{'C','l','e','a','r'},ENABLE}
    {{'S','h','o','w'},ENABLE}
};
```

Using a Mouse on a Framed Window with Menus:

Step 1. Setup a framed window. To create a window in C, use the _ss_wset command. To set up a Multi-Vue window on an already existing standard window, you need to use a type specification and a pointer to a window data structure. For usage:

```
if (_ss_wset(path,WT_FWIN,&windat)<0)
    exit(errno);
```

Step 2. Set up the signal handling capabilities for mouse communication.

OS-9 handles mouse communication with signals. Therefore you need a signal handler to catch (save) the mouse signals.To create the handler in C, make an intercept call with the name of your signal handler as an argument.

For more information on signals, consult the *OS-9 Level Two Technical Reference* manual and the *C Compiler User's Guide*. For usage:

```
static int sigcode;

sighandler(sig)
int sig;
{
    /* save the code of the signal */
    sigcode = sig;
}

main()
{
    /* set up the signal handler */
    intercept (sighandler);
            .
            .
            .
}
```

Step 3. Activate the mouse by making a call to _ss_mous. This call requires parameters for the *cursor update rate*, the *state change timeout value*, and an *auto-follow flag*.

The cursor update rate specifies the rate at which the system checks the mouse for movement to new coordinates. For usage, an update rate of 3 causes the system to check the mouse for new coordinates every three clock interrupts. A value of 0 causes the mouse coordinates to be read only when the getstat of ss.mouse is called.

The state change timeout value specifies the time in clock interrupts before all mouse counters are cleared.

The auto-follow flag indicates whether or not the system updates the graphics cursor to the new mouse coordinates when the mouse changes position. For usage:

```
#define UPDATE  3          /* check for change every
                              three interrupts */
#define TIMOUT 10          /* timeout in ten tics. */
#define FOLLOW  1          /* update graphics cursor
                              when mouse is read and has
                               changed position*/

/* set up the graphics cursor */
SetGC(path,GRP_PTR,PTR_ARR);

/* get the mouse going */
_ss_mous(path,UPDATE,TIMOUT,FOLLOW);
```

Step 4. Set up for a mouse signal. Make a call to _ss_msig to request that your process receive a specific signal at the occurrence of a mouse click. As a parameter, indicate the signal code (*sigcod*) you want sent. For usage:

```
/* define value mouse signal, must be greater than three */
#define MOUSSIG  10

/* first clear your signal code parameter */
sigcode = 0;
_ss_msig(path,MOUSSIG);
```

Step 5. Wait for the mouse signal before continuing with processing. Do this by making a call to tsleep, sending a parameter of zero to make the user process sleep until it receives a signal:

```
if (sigcode == 0)
    /* no mouse signal yet so wait for one */
    tsleep(0);
```

Note: This step is optional. The program can continue processing without waiting for the signal. However, if it does, it must check *sigcode* periodically to see whether a mouse signal has occurred (whether sigcode contains a value greater than 0). In most instances, it is best to wait for a mouse signal before continuing.

On receiving a mouse signal, the system executes the signal handler routine *sighandler*. This signal handler saves the signal and terminates. Program processing resumes at the statement after the tsleep function call. You can now validate that a mouse signal is received, and act accordingly:

```
if (sigcode == MOUSSIG) {
                .
                .
    /* process mouse signal */
                .
                .
```

Step 6. Process the mouse signal.

To begin, you need to acquire a *mouse packet*. The mouse packet contains all information pertaining to the mouse's status and functionality. You can acquire the packet by making a call to _gs_mous, passing a pointer to a mouse packet structure.

For a complete description of the mouse packet structure, see the Mouse.h header file in your DEFS directory.

```
/* declare the mouse packet structure */
MSRET msinfo;

/* make the call */
_gs_mous(path,&msinfo);
```

After acquiring the mouse information, process it in the following manner:

```
int itemno;

if ((msinfo.pt_valid) && (msinfo.pt_stat!=WR_OFWIN)) {
    if (msinfo.pt_stat == WR_CNTRL) {
        /* mouse was clicked on the control region */
        switch (_gs_msel(path,&itemno)) {
            case NM_CLOS:
                quit = TRUE;
                break;
            case MN_TNDY:
                do_tandy_menu(itemno);
                break;
            case MN_FILE:
                do_file_menu(itemno);
                break;
```

```
                case MN_EDIT:
                    do_file_menu(itemno);
                    break;
                default:
                    do_error();
            }
        }
        else {
            /* mouse was clicked on the content region */
        }
    }
```

The preceding section of code first determines that the mouse packet is valid for its window. It does this by verifying that the parameter msinfo.pt_valid contains a value other than 0. It then checks the parameter *msinfo.pr_stat* to see if the mouse was clicked on the window. If it was, further checking determines whether or not the mouse was clicked in the control portion or the content portion of the window.

If the mouse is clicked in the control region (on the menu bar) the routine performs a pull-down of the associated menu by making a call to _gs_msel. This call requires a pointer to an integer as its argument. The _gs_msel function returns the number of the menu selected and the item number in the menu selection chosen by an additional click.

In this usage, the item number of the menu is returned in the parameter *intemno*. The program can then continue processing according to the selections.

Note that the pull-down that is associated with the menu continues in existence until another mouse click makes another selection. If the mouse is clicked on a location other than an item in the pull-down, *itemno* is set to 0.

The code makes provisions for the mouse to be clicked on the control portion of the window. You can determine where on the content region the mouse is clicked by accessing the location information in the mouse packet. The formula for determining the mouse location in 640x192 coordinates on a scaled window is as follows:

```
/*declare parameters */
MSRET msinfo;
int xsz,ysz;
int xwindsz,ywindsz;  /* declare parameters as integers */
int mousx,mousy;

/* acquire size of window */
if (_gs_scsz(path,&xsz,&ysz)< 0) exit (errno);

/*convert character values to pixel values */
xwindsz = xsz * 8;
ywindsz = ysz * 8;
                .
                .
                .
/* now each time the mouse location within the
   content region is desired, the following
   formula is applied */
mousx = ((long)640 * msinfo.pt_wrx) / xwindsz;
mousy + ((long)192 * msinfo.pt_wry) / ywindsz;
```

Appendix

Installing Multi-Vue

Disk Extended Color Basic will use either a DOS or a RUN boot command. To determine the boot command you need to use, do the following:

1. Be sure all equipment has been properly connected.

2. Turn on your computer. The Color Computer 3 displays the following:

```
DISK EXTENDED COLOR BASIC X.X
COPR. 1982, 1986 BY TANDY
UNDER LICENSE FROM MICROSOFT
AND MICROWARE SYSTEMS CORP.

OK
```

3. Look at the numbers at the end of the first line (indicated by X.X in the sample above). If X.X is less than 2.1, you need to boot OS-9 using the RUN command and a BASIC program. If X.X is 2.1 or greater, you system will boot with the DOS command.

If X.X is 2.1 or greater, create your Multi-Vue working diskettes as described in Chapter One. Then, whenever you want to start Multi-Vue, insert Diskette 2 into Drive 0 and type:

DOS [ENTER]

If X.X is less than 2.1, you'll need to install the following BASIC program:

```
10 REM ******
20 REM *BOOT OS-9 FROM BASIC
30 REM ******
40 FOR I= 0 TO 70
50 READ A$
60 POKE &H5000+I, VAL("&H"+A$)
70 NEXT I
80 CLS: PRINT "INSERT MULTI-VUE DISK 2 INTO"
90 PRINT "DRIVE 0 AND PRESS THE SPACE BAR."
100 A$= INKEY$: IF LEN(A$)=0 THEN 100
110 EXEC &H5000
120 DATA 86, 22, 8E, 26, 00, 8D, 0D
130 DATA FC, 26, 00, 10, 83, 4F, 53
140 DATA 26, 03, 7E, 26, 02, 39, 34
150 DATA 20, 10, BE, C0, 06, A7, 22
160 DATA 86, 02, A7, A4, 6F, 21, 6F
170 DATA 23, 6C, 23, AF, 24, 10, BE
180 DATA C0, 06, A6, 23, 81, 13, 27
190 DATA 12, AD, 9F, C0, 04, 4D, 27
200 DATA 06, 6C, 23, 6C, 24, 20, E9
210 DATA 7F, FF, 40, 35, A0, 4F, 20
220 DATA F8
```

Save this program on a formatted diskette by using the following command:

SAVE "*" [ENTER]

Label the diskette for future use.

Create your Multi-Vue working diskettes as described in Chapter One. Then, whenever you want to start Multi-Vue, insert the diskette containg this BASIC program into Drive 0 and type:

RUN "*" [ENTER]

When the *Insert Multi-Vue* prompt appears, insert Multi-Vue Diskette 2 into Drive 0 and press the space bar. Set the time and date prompts when requested. Multi-Vue's copyright message will appear and you will be instructed to place Multi-Vue Disk 1 into Drive 0 and press a key. When you have done this, the GShell screen will come up as described in Chapter 1.

Glossary

Application Information File (AIF). A special file containing an application name, parameters, and screen information that tell GShell how to run an application.

application. A process or group of processes designed to accomplish specific tasks, such as word processing, data management, game playing, and so on.

backup. An identical copy of the contents of one diskette on another diskette.

baud. Bits-per-second. A unit for measuring the speed of data flow between devices.

binary. A numbering system using only two digits, 0 and 1. In this system, shifting the position of a digit to the left raises the value of the digit by the power of 2. For instance, 1 = 1, 10 = 2, 100 = 4, 1000 = 16, and so on.

bit. The smallest unit of a computer's memory. Eight bits form a byte. Each bit can have a value of either 0 or 1.

bitmap. A special module that contains the information that Multi-Vue uses to form an icon.

boot. To load and initialize a system or program.

button. The pad on a mouse or joystick that you press to select an action from a screen display or menu.

click. To press and release the mouse or joystick button.

close. To deallocate the path to a device or file.

command. The name of an OS-9 program or function.

command. The name of an OS-9 program or function.

composite monitor. A type of monitor (TV-type device) that accepts screen information for all screen colors through one input line as a mixed signal.

control panel. A Multi-Vue utility that lets you set environmental parameters, such as screen colors and keyboard response.

current directory. The directory in which OS-9 looks for data files or in which it stores them unless you specify otherwise.

data directory. The directory in which OS-9 automatically saves files unless you specify otherwise.

data file. A file that contains information to be processed by a program.

device. A data source, destination, or both. OS-9 devices can exist in your computer's memory (as in the case of a window or a RAM disk), or they can be external equipment (as in the case of a printer or disk drive).

directory. A file in which OS-9 stores information about other files, including their names, locations on the disk, and attributes.

double click. To press and release the mouse or joystick button twice in quick succession.

env.file. A special Multi-Vue file that contains Multi-Vue format information, such as screen color, monitor type, and keyboard response.

environment. The conditions that exist (screen color, screen size, keyboard response, and so on) while a window is active.

error code. A code that OS-9 displays when it cannot understand what you want it to do or when your computer or a

peripheral malfunctions. Use the displayed code number to look up a description of the error.

execute. To start a procedure, program, or command (cause it to run).

file . A block of information your computer uses for a particular function or program. A file can contain an operating system, a language, an application program, or text.

filename. A set of characters that uniquely identifies and locates a block of data stored on a disk.

fire button. *See* button.

folder. The Multi-Vue name for a disk directory. *See* directory.

font. A character set. A group of characters (letters, numbers, and symbols) of a particular style and shape.

fork. To initialize one procedure from another.

format. To magnetically organize a diskette so that the computer can use it to store data.

get/put buffer. A buffer in which you or the system can store fonts, screen patterns, graphic displays, overlay windows, and other recallable data.

GShell. An OS-9 graphics shell that interprets commands from the user through keyboard and pointer inputs.

hexadecimal. A numbering system to a base of 16. The 16 hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Shifting a hexadecimal digit one place to the left causes its value to be multiplied by 16.

icon. A picture or image representing an OS-9 object (directory, device, or file) or an OS-9 action.

keyboard mouse. An OS-9 function that lets you use the keyboard arrow keys instead of an external mouse device. Press [CTRL][CLEAR] to toggle the keyboard mouse on and off.

keyboard repeat. The function that causes a keypress to repeat if you hold a key down for a predetermined length of time (about one second unless you select otherwise).

line feed. To move the paper in a printer up one line.

load. To transfer data from an external device into your computer's memory.

memory. The portion of your computer that stores data and values.

menu. A screen display from which you select an action for your computer.

module. An OS-9 program or block of data residing in your computer's memory.

monitor. The video display device connected to your computer.

monochrome monitor. A TV-type device that displays characters and graphics in black and one color (white, amber, or green).

mouse. A device you use to control a pointer on the display screen.

object file. A file that contains a program or utility--an executable file.

open. To create a path to a file or device over which data can be transferred.

palette. A register (storage area) that contains a numeric code representing a color.

parity. A system in which the numbers of a code are converted to either even-bit numbers (an even number of 1's or 0's) or odd-bit numbers (an odd number of 1's or 0's).

pointer An image that you can move on the screen to indicate selections from a menu or other display.

port. A junction between devices, through which data flows. An electrical connection between your computer and a peripheral.

program. Code that causes your computer to perform a function or a series of functions.

read. To transfer data from a device into the computer's memory.

resolution. The number of individual screen points (pixels or picture elements) that can be turned on or off.

RGB monitor. A TV-type device that requires color information from three separate signals. One signal represents red, another represents green, and a third represents blue.

run. To execute a program or procedure.

shell. The OS-9 command interpreter. The shell converts input from the keyboard or some other device into code that controls your computer's performance.

start up. To begin operating. To initialize your system or a program.

stop bits. One or two bits that a terminal program sends after each unit of data to indicate that the transmission of the unit is complete.

system. (1) A group of files and programs that provide you with control over your computer. (2) Your computer with all its attached devices.

utility. A short program that performs a frequently required task, usually for the maintenance of your computer system or files.

window. All or a portion of your video screen with a specific format (columns, lines, size, colors, and so on) and type (graphics, text, or both). An area of a screen in which you can run a process or that can receive input.

word length. The number of bits to transmit as one unit.

write. To transfer data from the computer's memory to a device.

Index

B

C

D

I

J

K

L

M

N

O

P

Q

R

V

W

X

RADIO SHACK
A Division of Tandy Corporation
Fort Worth, Texas 76102

11/87-SP 874-8036 Printed in U.S.A.