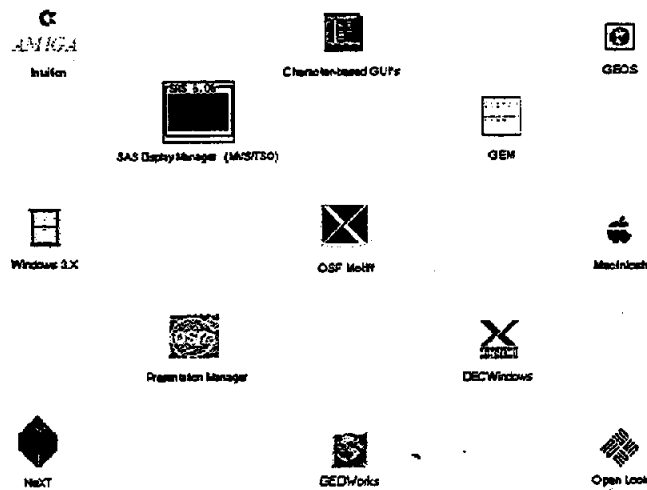




How to implement a Graphical User Interface using SAS 6.06 under MVS/TSO, and the reasons for doing so (... without confusing your present users)

By James Hefner
Production Engineer
Gulf States Utilities Company

October 13, 1992

Acknowledgments

The author wishes to thank SAS Institute Inc., for their help. Without the software documentation, technical support, and SAS *Communications* magazine, neither this paper nor my application interfaces would have been possible.

The author would also like to thank his manager, Walter Irwin, and his wife Julie, for their patience in allowing me to complete this paper.

By a new and living way, which he hath consecrated for us, ...
Hebrews 10:20

Picture Credits

Figures 1 through 24 are of the SAS software running under TSO/MVS. Rumba was the terminal emulator used; the screens were copied from Rumba to Microsoft Word for Windows. The default screen colors were altered to enhance printing.

The pictures of Microsoft Windows screens (with the exception noted below) and of the MS-DOS Shell were captured from within Windows, and copied to Microsoft Word for Windows.

The remainder were scanned in from the original documentation, and altered to reproduce the original image as close as was possible. The resulting illustrations were then saved in .PCX format, and imported in Word for Windows. Some changes were made in the process.

The illustration of SAS running under OS/2 is from the brochure *SAS/CONNECT Software - Distributed Processing*, published by SAS Institute Inc., Cary, NC, 1990.

The illustration of SAS running under X-Windows is from page 5 of the First Quarter, 1991 issue of *SAS Communications*, published by SAS Institute Inc., Cary, NC, 1991.

The illustration of SAS running under DECwindows is from page xviii of *SAS Screen Control Language: Reference, Version 6, First Edition*, published by SAS Institute Inc., Cary, NC, 1990.

The illustration of the GEM environment is from page 133 of an article by Tami Peterson. "Desktop Publishing (Ventura Publisher)" *PC Magazine*, December 27, 1988.

The illustration of PCTools is from page 7 of *CENTRAL POINT PCTOOLS, DOS Shell/File Manager*, published by Central Point Software, Inc., 1991.

The illustrations comparing Microsoft Windows to the Macintosh interface were taken from page 37 of an article by Jeffrey Schwartz. "Omnis 7: Equal Access for Macs, Windows." *Communications Week*, July 20, 1992.

The illustration of the NeXT operating system was taken from page 62 of a box by Kathy Coakley. "Is Three a Lucky Number for NeXT?" *PCWEEK*, May 11, 1992.

The illustration of the GEOS operating system is a composite of two pictures taken from pages 21 and 34 of *GEOS 2.0- Users Manual*, published by Berkeley Softworks (now GEOWorks), 1988. The composite picture depicts an actual screen on my home machine.

The illustration of the Amiga Intuition operating system is based on an illustration on page 126 of an article by Bob Ryan. "Commodore Sets Course for Multimedia" *BYTE*, May 1990.

The illustration of Apple Hypercard was taken from an advertisement by Sigma Designs on page 17 of *MACWORLD*, September, 1990.

ABSTRACT

IBM's SAA strategy provided a common graphical interface for all programs on all platforms. Known as CUA, this is the interface that is implemented by SAS under MVS/TSO, OS/2 and other platforms, starting with version 6.06.

However, the full CUA implementation under MVS/TSO assumes that the user has a graphics terminal with a mouse. Many of the terminals in use today do not have a mouse, or even graphics capability. In accommodating users who have such terminals, a programmer using SAS/AF has to design each SAS Application so that the interface does not confuse their non-mouse users, while still being useful for mouse users.

This paper outlines the steps in creating a sample application. The main menu screen and help screens are discussed and built so that they are context-sensitive. A program dialog box is constructed, showing how to design the entry fields so that they guide both mouse and non-mouse users in the entry of data. Guidelines for making the application CUA compliant are given. A fully mouse-driven interface is then created by modifying the sample application. Finally, examples of graphical user interfaces are given to demonstrate that implementing the CUA interface could reduce employee retraining.

INTRODUCTION

Recent years have seen the introduction and growth of graphical user interfaces on various PC platforms and operating systems. Even DOS character-based applications that do not require a graphical operating system, such as PCTools include their own character-based interface that mimics a graphical user interface. They all have several features in common, including the use of a cursor and a hardware pointing device such as a mouse to perform most operations. The use of the mouse or pointing device, and even some keystrokes, are very similar for all of these interfaces. They also share identical graphical features such as icons, windows, scroll bars, menus, radio knobs, and push buttons. IBM, along with Apple, is partly responsible for this by defining a standard for interfaces. This standard, known as CUA, or Common User Access, is part of IBM's SAA strategy. It is used by developers writing for IBM Mainframe, OS/2, X-Windows, Microsoft Windows, and DOS.

At least one graphical user interface is available for every PC platform and operating system. IBM Mainframe users and application developers, however, have been forced to stand on the sideline and watch. That is, until the introduction of terminals and terminal emulators that support a mouse, and SAS version 6.06. Now, mainframe application developers can write windowing applications that fully support the use of a mouse. However, most departments have terminals that do not support a mouse. They also have users who have never used, or do not like using, a graphical user

interface. This paper is about writing SAS applications using AF that support a mouse. This is not a paper on an original work of programming, since all of the techniques described can be found in the SAS 6.06 documentation and SAS *Communications Magazine*. Rather, it outlines a particular approach to writing SAS 6.06 applications. It addresses writing applications that are 'mouse-able' without confusing or isolating users who are still locked into using applications 'the old way.' A sample program will be written, and guidelines for writing CUA-compliant applications will be given. Displays from various graphical user interfaces will be shown in order to demonstrate their similarities.

APPLICATION CONSTRUCTION

Let's write a SAS application that allows the user to edit the data in a crime database. (For the purpose of this example, the dataset SASUSER.CRIME, that is created by SAS when you first call up SAS ASSIST, will be used.) The creation of reports and graphs based on this data are needed as part of the application. Context-sensitive help will also be provided. At first, only a few users will be using a mouse, so the transition from ISPF/DMS-style screens to windows will be somewhat transparent. Later on, when all users are using a mouse, the application will be fully mouse-driven. Finally, the application will be CUA-compliant, to make the application easier to use.

Main Menu

The first task is to determine what the main menu will look like. A menu could be designed using a PROGRAM entry, similar to SAS ASSIST, as shown in Figure 1. However, have a mainframe veteran encounter this screen, and immediately they may be at a loss as to what to do. Also, there is a limit of 12 items on each menu screen. For a large application, this may mean snaking through several menu screens.

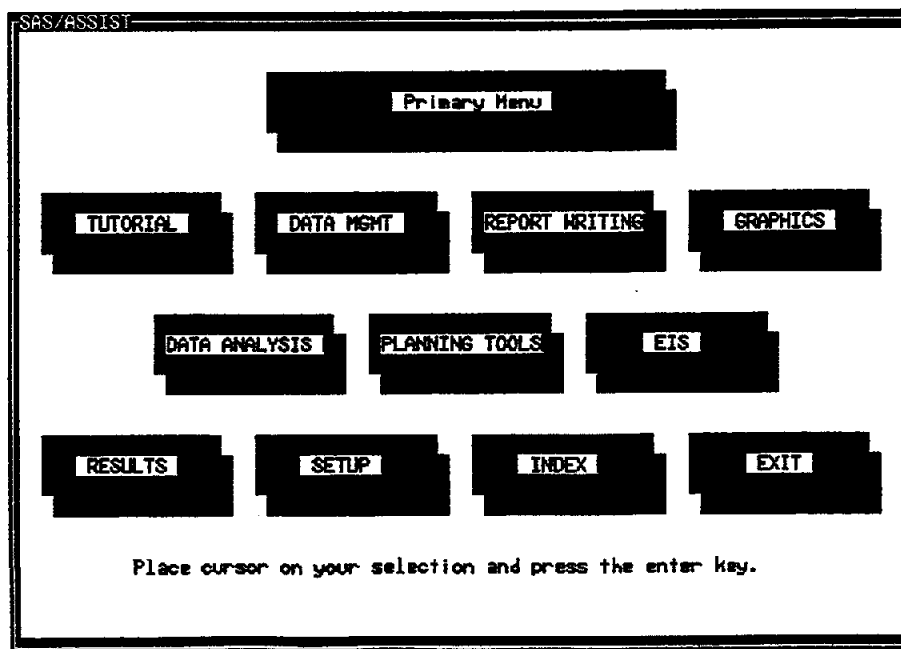


Figure 1

A MENU entry could also be used, as in previous versions of SAS. In order to satisfy the mouse users, a PMENU could be written to create a pull-down menu. The final result is shown in Figure 2. This was the initial approach used to make the existing applications 'mouse-able,' but it presented two problems. First, those who were using

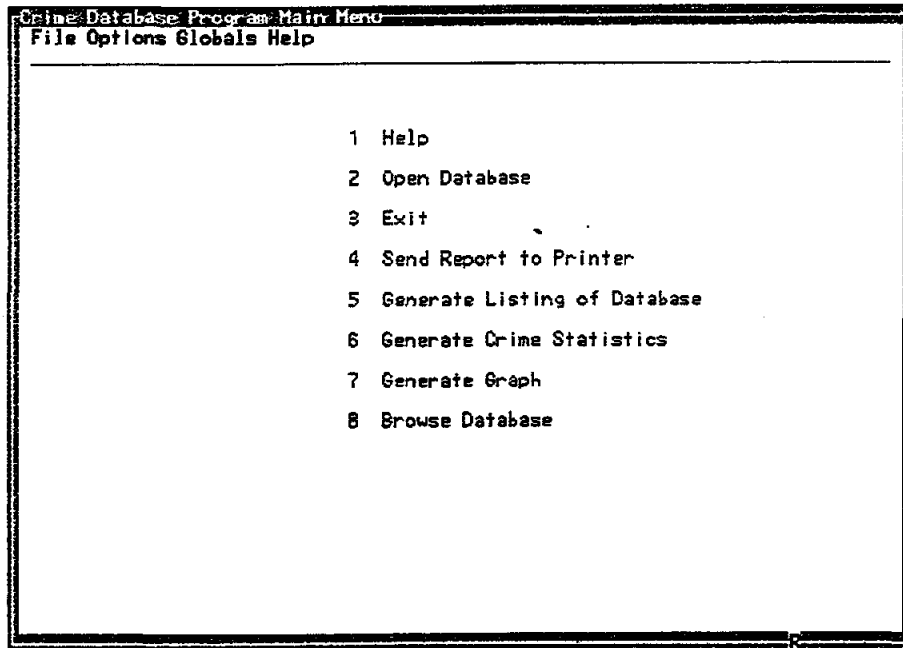


Figure 2

a mouse tended to click on the words in the middle of the screen, instead of the pull-down menu. Of course, with a MENU entry, nothing happened. Second, those who were used to typing something on a command line were lost. Of course, telling them to press the appropriate function key solved that problem. Still, there must be a better way...

The answer is to use a CBT type for the menu screens. The advantage of this approach is that all of the text on the screen can be clicked on to do something or call something up. That is, the menus become hypertext menus. Also, the PMENU can be left out and a command line can appear at the top of the window. That way for example, a person can press PF1, or type a 1 on the command line, or click on the word Help on the menu screen to get help.

Next, the main menu will be set up. First, the entries can be typed in. In trying to keep the application CUA compliant, remember that option 1 should always be Help and option 3 should always be End. Option 2 is the option for opening the database, and option 4 the option for sending a results file to a printer. This makes the application consistent with IBM's Interactive Chart Utility (ICU). Next, the frame indicator lines are typed in. These lines designate what to do when an item is selected, as shown in Figure 3. Notice the frame indicator line above the menu that names the menu screen. Note also the frame indicator lines at the bottom, which

```

BUILD: DISPLAY MAIN.CBT (5)
File Edit View Locals Globals Help

?1 LOCK name=main

1 Help
2 Open Database
3 Exit
4 Send Report to Printer
5 Generate Listing of Database
6 Generate Crime Statistics
7 Generate Graph
8 Browse Database

#1 > primary.cbt      select=(17,24,6,6) menu=1 help=primary.cbt
#2 X update.program   select=(17,49,7,7) menu=2 help=primary.cbt
#3 X leave.program    select=(17,25,8,8) menu=3 help=primary.cbt
#4 X print.program    select=(17,39,18,18) menu=4 help=primary.cbt
#5 X listing.program  select=(17,41,11,11) menu=5 help=listing.cbt
#6 X stats.program    select=(17,33,13,13) menu=6 help=stats.cbt
#7 X graph.program    select=(17,48,14,14) menu=7 help=graph.cbt
#8 X browse.program   select=(17,46,15,15) menu=8 help=primary.cbt

```

Figure 3

indicate where to go when each item is picked, either by clicking on it (the select statement) or by typing in the number (the menu statement). Also, when passing control to another CBT entry (such as primary.cbt in this figure), notice that it is preceded with a '>' instead of '><'. The only downside of using a CBT entry for menu screens instead of a MENU entry is that the links to submenus must be set up manually.

In order to be able to use the function keys to select an item, they must be redefined. Figure 4 shows how to define them for a menu screen. Notice how PF1 and PF3 are not numbered, but remain HELP and END. Since item 1 is already designated to be

```

Crime Database Program Main Menu -KEYS <AF>
File Options Globals Help      File Edit View Globals Help

Key      Definition
1 Help   F1      help
2 Open   F2      2
3 Exit   F3      end
4 Send    F4      4
5 Gener  F5      5
6 Gener  F6      6
7 Gener  F7      7
8 Brows  F8      8
          F9      9
          F10     10
          F11     11
          F12     12
          F13     help
          F14     icon
          F15     end
          F16     prevend
          F17     nfind
          F18     rchange
          F19     backward
          F20     forward
          F21     swap
          F22     left
          F23     right
          F24     none
          MB2     menu

```

Figure 4

Help, and item 3 to be End or Exit, it is easier to leave them set up as they are. In this illustration, PF13 to PF24 are left at their default setting; if there are more than 12 menu items, they can be renumbered as well.

Finally, to complete the menu screen, the general attributes for the CBT entry must be defined. Figure 5 shows the general attributes screen for the example menu screen. Notice the window title that was typed in the first blank. **COMMAND** was selected to

```

BUILD: DISPLAY MAIN.CBT (E)
BUILD: GATTR MAIN.CBT (E)
File Edit View Locals Globals Help

Window      Name:  Global Database Program Menu Menu
            Start row:  col:
            Number of rows:  cols:  Banner:  COMMAND  SELECT  NONE

General      Help:  PRIMARY.CBT
            Keys:  AF
            Lookup data set:
            Command menu:
            Prompt character:
            System options:  NO EXIT  RESIDENT  EXTENDED TABLE

Parent      Name:
            Type:
            Libref:
            Catalog:

Window Type  STANDARD  DIALOG  DIALOG BOX  LIST  HELP

Scroll Bars  HORIZONTAL  VERTICAL

#1 > primary.cbt      select=(17,24,6,6)  menu=1  help=primary.cbt
#2 << update.program  select=(17,49,7,7)  menu=2  help=primary.cbt
#3 << leave.program   select=(17,25,8,8)  menu=3  help=primary.cbt
#4 << print.program    select=(17,39,18,18) menu=4  help=primary.cbt
#5 << listing.program  select=(17,41,11,11) menu=5  help=listing.cbt
#6 << stats.program    select=(17,33,13,13) menu=6  help=stats.cbt
#7 << graph.program    select=(17,48,14,14) menu=7  help=graph.cbt
#8 << browse.program   select=(17,46,15,15) menu=8  help=primary.cbt

```

Figure 5

appear as the banner at the top of the screen. Since no command menu (PMENU) is assigned to this menu, a command line proceeded by the word 'Command ==>' will appear at the top of the screen. The function keys that were set up previously are entered as the Keys for this screen. The window type is set at standard (this will also cause a pull down menu to appear at the top, when a PMENU is assigned later). Finally, no scroll bars are selected to appear since none are necessary.

Help Screens

Next, the primary help screens, PRIMARY.CBT, will be set up. Since the help screens are also CBT-type screens, the procedure for setting them up is similar to the menu screen, with some small differences.

Figure 6 shows the main, or index screen for help system. Notice how the main frame or screen is named Index, using the frame indicator line at the top of the screen. By

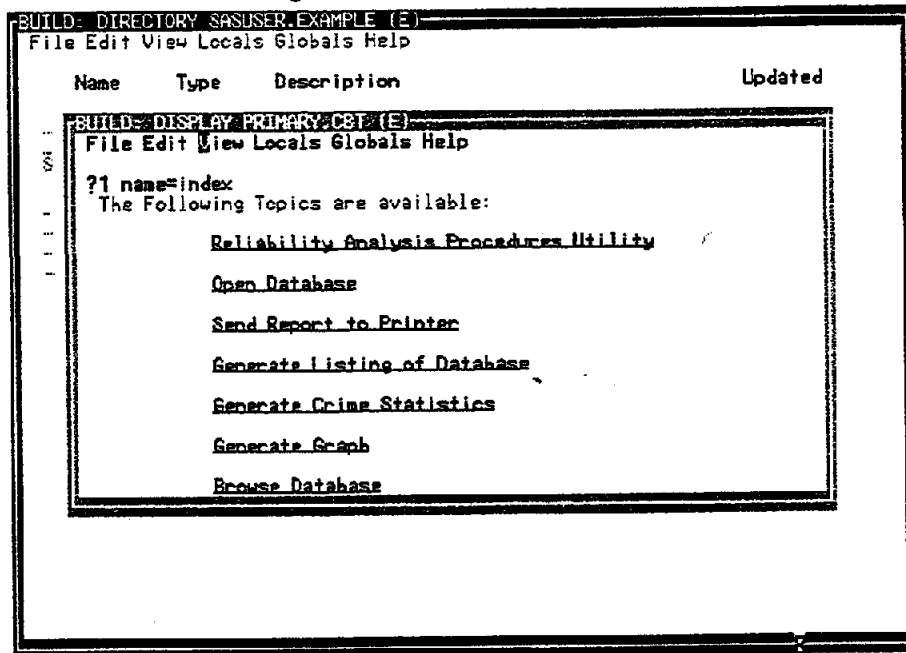


Figure 6

naming the main screen Index, and placing the following line at bottom of each frame, a push button labeled Index can be set up that can always be used to return to the index screen:

```
#0 >> primary.cbt frame=index menu=index.
```

Notice how the regular text appears as one color, and the text that can be selected (hypertext) appears as another color and underlined. This helps to make the hypertext stand out. The resulting help screens look and feel very much like a Microsoft Windows help screen (see the illustration in Appendix A for a sample Windows help screen).

Always set up a screen describing what the application is used for, and a screen for each option that is available. If the help screens for each option are placed in a separate file, the application can be set up so that if the user asks for help within an option, he or she will instantly receive help for that particular option. This is known as context-sensitive help. By having an Index button appear at the bottom of each help screen, the user can click on it to call up the Index. Also, if the current option interacts with another option, provide a field that allows the user to access help information on the interacting option. Figure 7 shows an example. Since the Listing option of this program generates an output file, the user may want to send the file to a printer. Therefore, a field is provided for the user to click on to get more information on sending the output file to a printer. This figure also illustrates adding the above frame indicator line to allow the user to call up the help index. Also notice that the screen is stored under a separate file named LISTING.CBT. By making LISTING.CBT

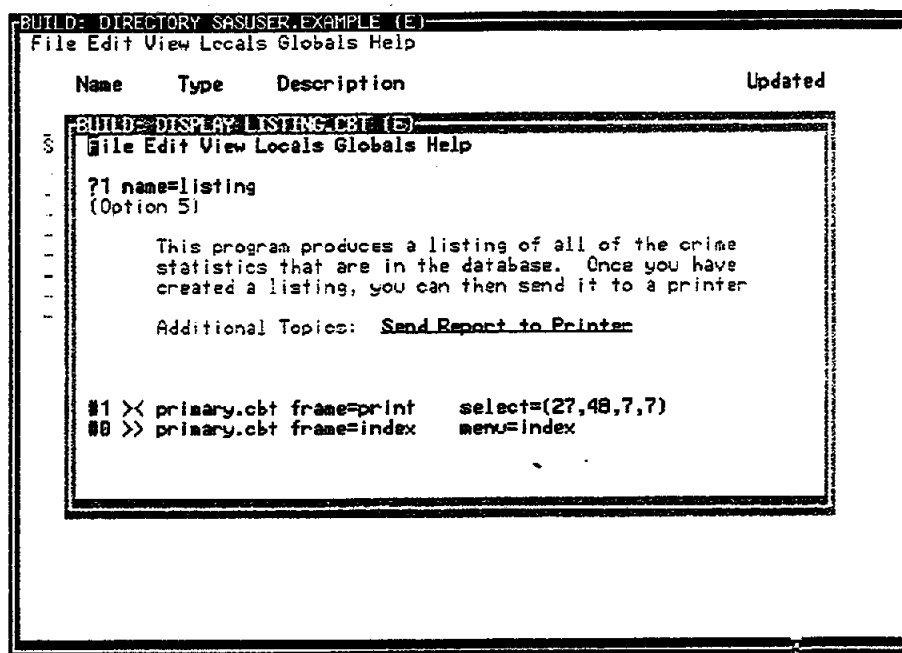


Figure 7

the help screen for LISTING.PROGRAM, the user can get context-sensitive help from within the program by pressing PF1.

Once the help screens have been typed in, their general attributes can be defined. Figure 8 shows the general attributes for all of the help screens. If your help screens

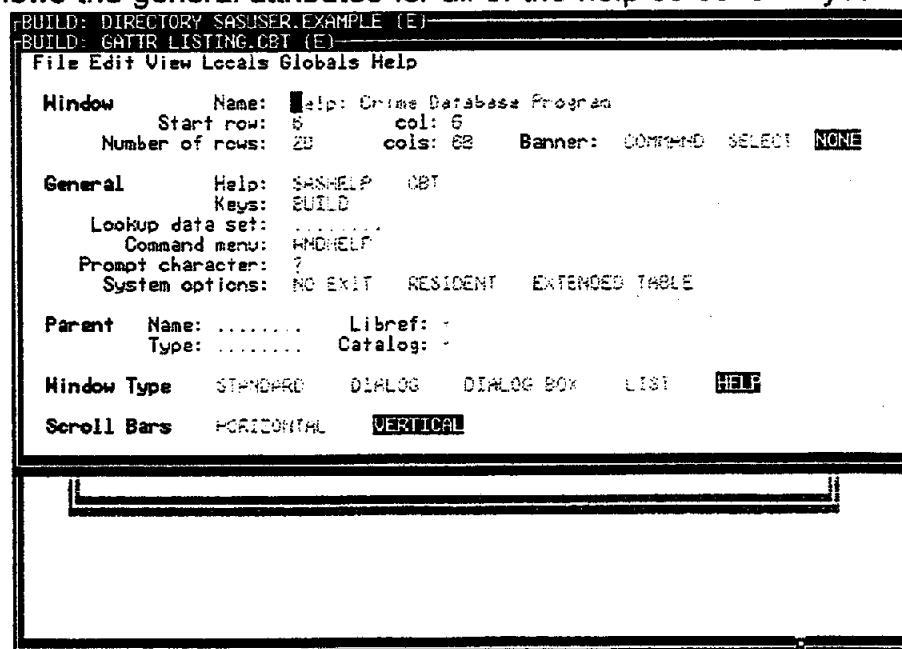


Figure 8

are stored in separate files, then set the general attributes the same for all of them. Appendix B lists the source code used to create the Command menu attribute WNDHELP.PMENU for all of the help screens. The main screen for SAS 6.06 is listed as the help for the help screens themselves. That way, if a user wants help on the SAS system itself, it is available by pressing PF1 from within the application help. The only problem seems to be that in order to implement this, the SAS CBT help files must be found, (usually under the libname SASHELP), and copied to the program catalog. Note that the Window Type is Help, and that a vertical scroll bar is added for scrolling through the help screens.

Program Dialog Boxes

At this point, the menu screens and help screens are complete. The next task at hand is to develop the PROGRAM entries to execute the procedures. Very often, the procedures to be incorporated are already written. It may already have an AF interface, or may have a TSO interface which in turn submits the SAS code for processing. The goal in incorporating the option into the program is to provide an interface that makes it intuitive and easy for your users to make choices and execute that option. Let's write the interface for the graphing option of this sample application.

Figure 9 shows the Graph module as it originally appeared. This screen may have originally been created in version 5 of SAS, or was an ISPF/DMS panel. The objective

```
Create Graph of Crime Statistics

Specify which types of crime you would like to graph:
  _ Murder _ Robbery _ Rape _ Assault _ Burglary _ Larceny _ Auto

Select which States:
  48 22 48 5  _ (? for list)

Select a destination for the graphs:
  _ My Graphics Terminal X Printer ( P394 ? for list)

When you are through, press PF3 to exit and run the program.
```

Figure 9

is to take this screen and make it 'mouse-able' without making it confusing for users on non-graphics terminals. The first step is to set up choice groups for your types of crimes and graph destinations. To do this, replace the text for each choice with a field. Figure 10 shows how the screen looks once this is done. Next, set up the field

```

BUILD: DISPLAY GRAPH.PROGRAM (E)
File Edit View Locals Globals Help

Specify which types of crime you would like to graph:

  & &C1___ & &C2___ & &C3___ & &C4___ & &C5___ & &C6___ & &C7___

Select which States:

  &1 &2 &3 &4 &5 (? for list)

Select a destination for the graphs:

  & &TERM_____ & &LINE___ ( &PRINT_ ? for list)

  When you are through, press F10 to exit and run the program.

```

Figure 10

attributes for each field. The field attributes for the first field are shown in Figure 11. The field is an action field, and is part of a choice group named CRIME1. To allow

```

BUILD: ATTR GRAPH.PROGRAM (E)
File Edit View Locals Globals Help

Use the scroll commands or function keys to review the fields.

Field name: FIELD1   Frame: 1   Row: 4   Col: 5   Length: 1

Alias: FIELD1   Choice group: CRIME1   Pad: INITIAL
Type: ACTION   Protect: YES NO
Format:         Just: LEFT RIGHT CENTER NONE
Informant:
Error color: BLUE   attr: NONE   Help: .....

List: X

Initial:
Replace:
Options: CAPS CURSOR REQUIRED AUTOSKIP NOPROMPT NON DISPLAY

```

Figure 11

users to tab to the field, the Protect attribute INITIAL is assigned. Normally, this field would become either a check box or a radio knob, depending on whether it is a single-station or multiple-station choice group. However, this field has a List attribute of 'X'. This causes the field to initially be a blank, allowing non-graphics users to tab to it and place an 'X' in it, as they would with version 5 of SAS. By also turning AUTOSKIP off, the cursor will not automatically jump to the next field and cause it to be selected, should the user then press Enter. Figure 12 shows the field attributes for the next field. By naming the Choice Group for this field CRIME1, we have made this field and the previous field a linked action field. If a user places the cursor on either field and presses the mouse button or Enter, an 'X' is placed in the first field and that item is selected. Assign the next pair of fields the Choice Group name of CRIME2, the next pair the Choice Group name of CRIME3, and so on, making each pair a single-station choice group. Also, set up the print destination fields to be a multiple-station choice group named Dest, allowing the user to select either one but not both destinations.

```

BUILD: ATTR GRAPH PROGRAM (E)
File Edit View Locals Globals Help

Use the scroll commands or function keys to review the fields.

Field name: FIELD2   Frame: 1   Row: 4   Col: 8   Length: 6

Alias: FIELD2   Choice group: CRIME1   Pad:
Type: CHAR   Protect: YES NO   INITIAL
Format:         Just: LEFT RIGHT CENTER NONE
Inform:
Error color: BLUE   attr: NONE   Help:
List:
Initial:
Replace:
Options: CAPS CURSOR REQUIRED AUTOSKIP NOPROMPT NON DISPLAY

```

Figure 12

In setting up fields where the user enters either text or numbers, you should set up a HELP entry listing the valid entries for that field. Figure 13 shows how you enter the

```

BUILD: ATTR GRAPH PROGRAM (E)
File Edit View Locals Globals Help

Use the scroll commands or function keys to review the fields.

Field name: 1   Frame: 1   Row: 9   Col: 5   Length: 2

Alias: 1   Choice group:
Type: NUM   Protect: YES NO   Pad:
Format: 2.0   Just: LEFT RIGHT CENTER NONE
Inform: 2.0
Error color: RED   attr: REVERSE   Help:
List: \ ? = STATES.HELP \
Initial: 48
Replace:
Options: CAPS CURSOR REQUIRED AUTOSKIP NOPROMPT NON DISPLAY

```

Figure 13

name of the HELP entry in the List attribute field so that users can access the list by typing a '?' in the blank and pressing Enter. (If the selection list is small and self-explanatory, list them in the List attribute field instead.) Figure 14 shows the HELP entry for the state field shown in Figure 13.

```

BUILD: DISPLAY STATES.HELP (E)
File Edit View Locals Globals Help

Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts

```

Figure 14

In the General Attributes window for each program entry, you should type in a meaningful Window Name. A help screen that is part of the HELP system should be entered in the Help: field. A PMENU named HEENDCAL is entered for the Command menu: field. (The source code for producing this PMENU is given in Appendix B.) And finally, the Window Type is set for Dialog, with no scroll bars. The general attributes screen is shown in Figure 15. These attributes will display a

```

BUILD: GATTR GRAPH PROGRAM (E)
File Edit View Localz Globals Help

Window      Name: Create Graph of Crime Statistics
            Start row: 1      col: 1
            Number of rows: 24 cols: 80  Banner: COMMAND SELECT NONE

General     Help: HEATRATC JST
            Keys: .....
            Lookup data set: .....
            Command menu: HEENDCAL
            Prompt character: ?
            System options: NO EXIT RESIDENT EXTENDED TABLE

Parent      Name: ..... Libref: .....
            Type: ..... Catalog: .....

Window Type STANDARD DIALOG DIALOG BOX LIST HELP
Scroll Bars HORIZONTAL VERTICAL
  
```

Figure 15

window with three buttons labeled 'OK', 'Cancel' and 'Help' along the bottom. Experience has shown that non-graphics users will ignore these buttons, and press PF3 to submit and PF1 for help, while users who have mice will take advantage of them.

Enhancements for Mouse Users

Once program entries are completed using the above guidelines, the finished product is a completed application that is user-friendly to both mouse and non-mouse users. Some of the programs that I have written have not advanced much beyond this point, for the majority of the users do not have a mouse. Let's advance further, and begin adding enhancements that make it easier to use a mouse. In setting up a program to be used with a mouse, the goal is minimize the amount of typing that is required. That way the user does not have to constantly take their hand off the mouse, type something in, and then click on something else. SAS ASSIST is a perfect example of optimizing this approach. . . an absolute minimum amount of typing is required to run ASSIST.

The first step is to go into the PROGRAM entries, and wherever there is a text field that allows the user to pull up a selection list, enter a '?' as the Initial attribute under the field attributes. This causes the selection list to be pulled up automatically when the program is called, allowing the user to click on his/her choice. Another route you can take is to assign a default value to the field, and place a push button next to the field. If the user wants to select something other than the default, they can click on the button and pull up a selection list. In this example, create a push button labeled 'Printers...' that the user can click on to pull up a list of valid printers. It will have an

Alias of 'Printerb', and have a type attribute of PUSHBTNC. The choice group for this button will be 'Printers'. It will have a list attribute of '1' and an Initial attribute of 'Printers...'. Link this field with the printer name field itself, using the following code:

```
INIT:
    CONTROL ASIS;
    CALL V5GETMAC ( ) ;
    DEST=1;
    PRINTERS = 0 ;
    RETURN;

MAIN:
    IF PRINTERS=1 THEN DO;
        DEST=2;
        PRINT=LISTC('PRINTERS.HELP');
        PRINTERS=0;
        END;
    RETURN;
```

where Printers is the choice group for the Printers... button and Print is the label of the field containing the printer name. Dest is the choice group for sending the graph either to the terminal (dest = 1) or the printer (dest = 2).

Also, remove the 'X' that is typed in as a List attribute for each ACTION field. This allows either a radio knob or check box to be displayed. You can also assign a default selection for each choice group. This reduces the amount of mouse movements required.

Finally, build a PMENU for the menu CBT screens. Appendix C lists the source code that was used to create MAINMENU.PMENU for the CRIME program main menu. This will replace the 'Command ==>' line at the top with an action bar with pull-down menus. As will be seen shortly, in addition to arranging the present menu choices in a Windows-style menu, additional items or features not listed in the menu itself can be added.

SAMPLE APPLICATION

Let us take a look at the CRIME program, with all of these enhancements. Appendix A defines all the underlined terms in this section.

Figure 16 shows the Main Menu for the CRIME sample application, as it appeared originally. In comparing this screen to the rewritten program in Figure 2, notice that the Main Menu is still the same. The menu choices are the same in the new version, and can still be selected by pressing the appropriate function key, or by typing its number on the command line. Instead of a command line, however, there is now an action bar in the upper left hand corner. Clicking on an item in the action bar reveals a pull-down menu from which an item may be selected. The border of the screen is now a window, with the window name appearing on the banner in the upper left

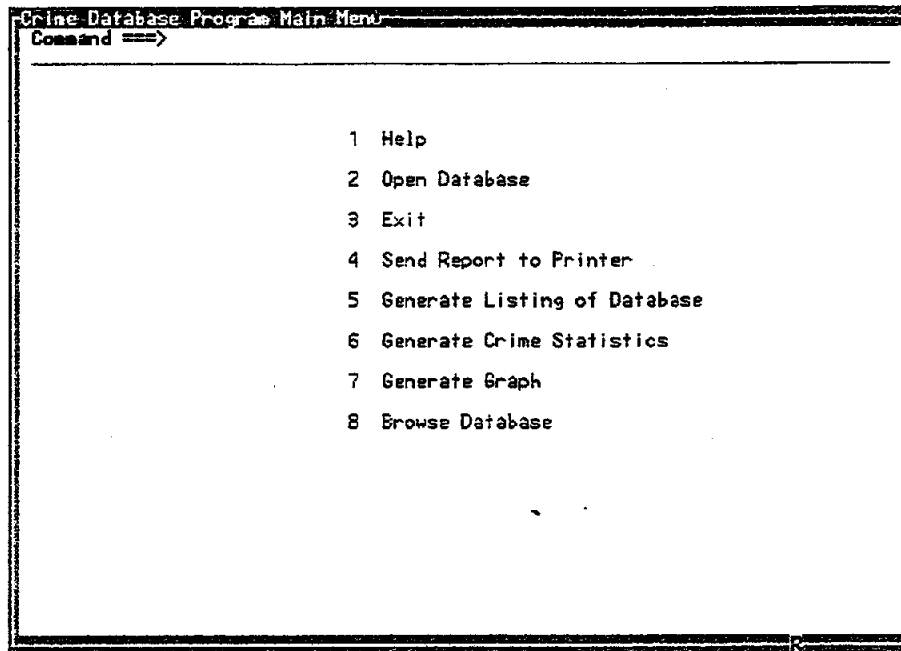


Figure 16

corner. SAS's new interface allows the user to have multiple windows open on the screen at once, with the active window on top. All windows have a meaningful name to them, so that one can always tell what a window is for by reading the banner. In this example, the main menu screen is displayed, so the window is named for the application -- 'Crime Database Program Main Menu.'

Select ANY item on ANY screen, either by tabbing to it with the tab keys and pressing Enter, or by clicking on it with a mouse using the left mouse button. In this SAS application, clicking on the word 'File' reveals a pull-down menu, as shown in Figure 17. This menu has options for updating, browsing, or printing the data file(s) and print file for this application, plus a choice of exiting the program.

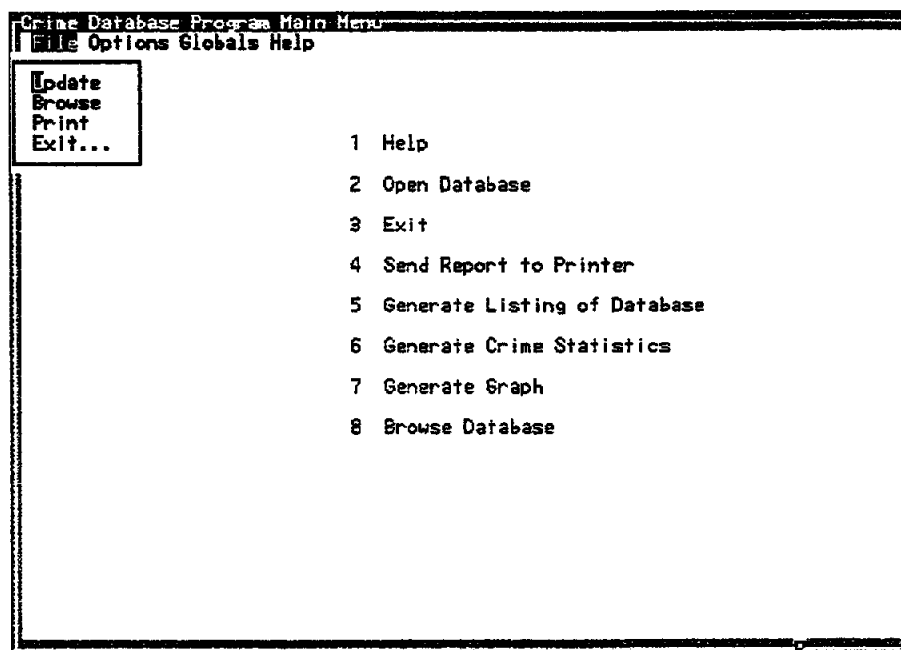


Figure 17

In this SAS application, clicking on Options reveals a menu of action options for this application, as shown in Figure 18. Action options include such things as running calculations and creating graphs and reports. Action items under the sub

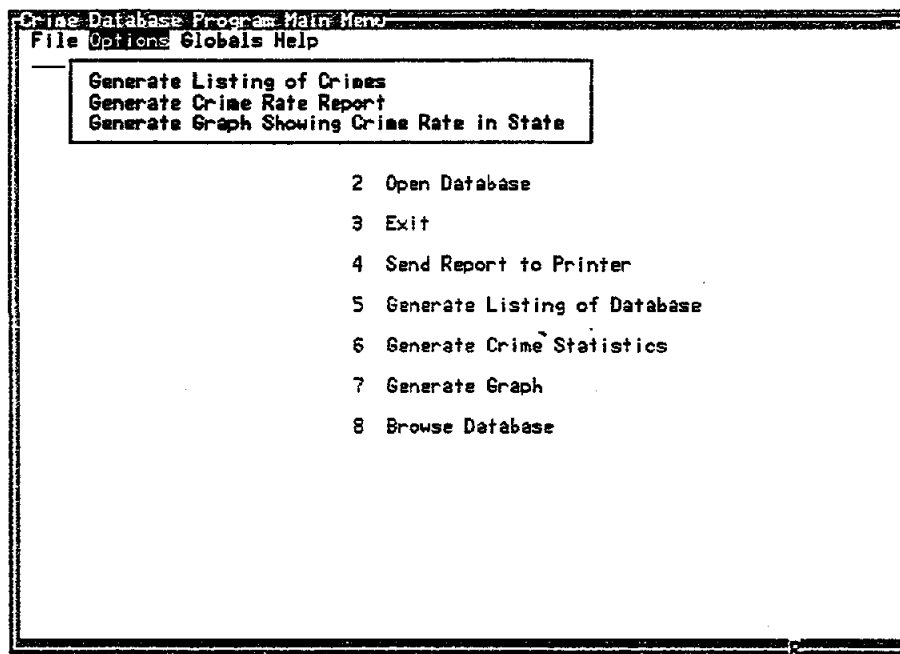


Figure 18

menus can also be accessed from the main menu.

This SAS application has seven items under Globals, as shown in Figure 19.

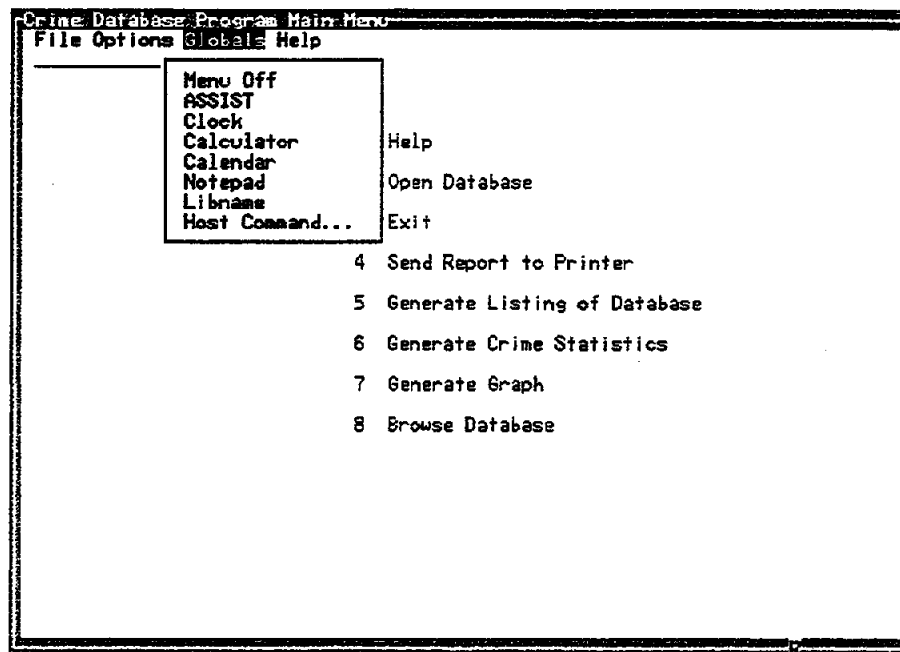


Figure 19

Menu Off allows the user to turn off the pull-down menus, and instead have a command line, just like the previous version of SAS. (**Pressing the right mouse button in the SAS application also will turn the menus off.**) Clock, Calculator, Calendar and Notepad are all like their counterparts in Microsoft Windows. Libname lets the user view a list of the SAS datasets currently allocated to that session for editing or deleting (provided they are allocated in TSO to allow one to do so). Host Command . . . issues a TSO command from within SAS. The user can call up ISPF, ICU, IOF and other programs. The only thing that cannot be ran is another SAS application. These commands could also be issued from the command line. However, the Graphical User interface allows the user to execute these commands without having to remember the syntax.

Clicking on 'Help' reveals a Help Index, instructions on using Help, SAS help screens, and 'About Crime' as shown in Figure 20. All help screens are context

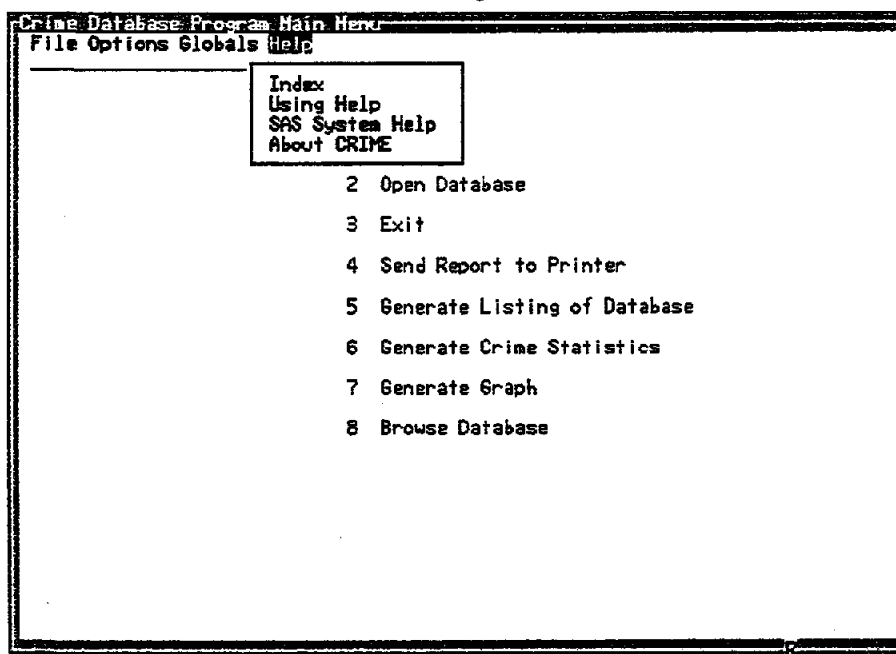


Figure 20

sensitive. Clicking on 'Index' displays the help index, as shown in Figure 21. From this point, select 'Goback' or 'ExitHelp' to return to the Main Menu, 'More' to read more General Help, or 'HELP' to get more help. Help on the SAS System is available by asking for help from the Program Help screens. From anywhere in the general help screens, select 'Index' to return to this screen, and 'Prev' to return to the previous screen.

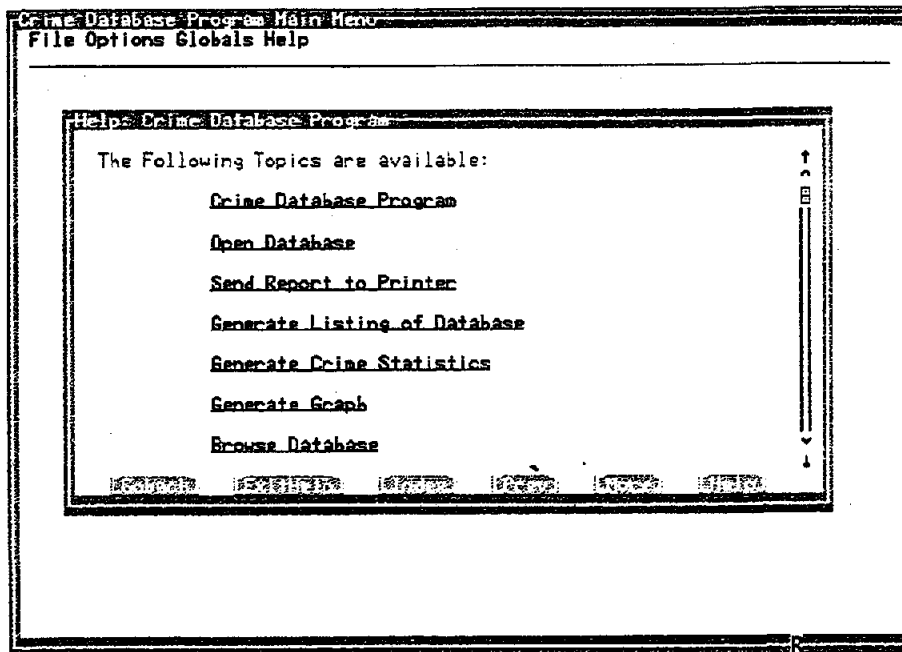


Figure 21

This application offers help for each module of the program. Selecting 'Index' from the pull-down menu reveals a list of help screens for each module. As will be seen later, these help screens can also be accessed by asking for help within the module itself.

This SAS application, like all Windows applications, has an item labeled 'About CRIME' where CRIME is the name of the application. Selecting 'About CRIME' in this CRIME example application displays the dialog box shown in Figure 22. 'About' screens always tell what the name of the application is and who wrote or published

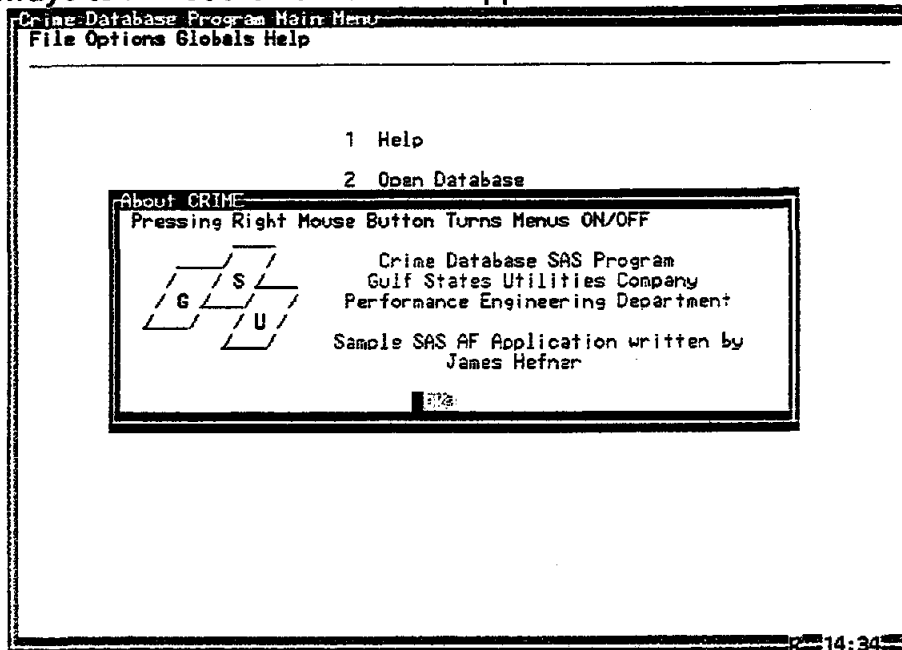


Figure 22

it, with some sort of logo. On several applications (the Microsoft Word for Windows word processor this paper was written on is one example), the 'About' dialog box also provides other information.

Note also in Figure 22 that Clock was selected from the 'Globals' pull-down menu and is now running in the lower right-hand corner.

Lets say that a graph of the murder and auto theft statistics for Texas, Louisiana, and Oklahoma is to be prepared. 'Create Graph of Crime Rate in State' is selected under Options in the pull-down menu, or F7 is pressed. A series of selection lists appears for each blank, allowing the user to select five states. One of these is shown in

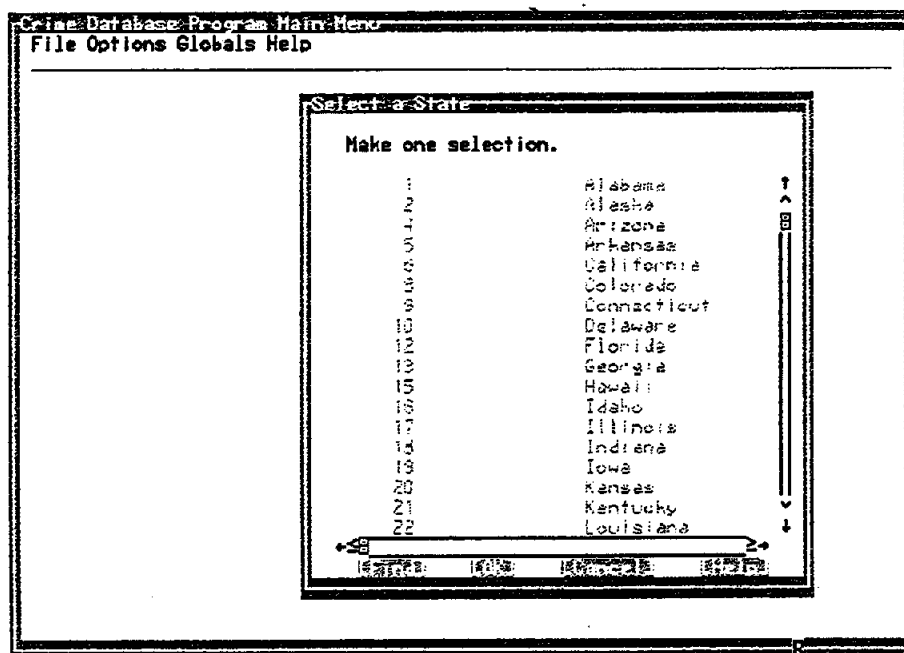


Figure 23

Figure 23. Texas, Louisiana, and Oklahoma are selected for the first three fields, then OK or Cancel is pressed for the next two fields to leave them blank. Next, the dialog window for the Graph module is displayed. Murder and Auto are selected for the graph. The button labeled 'Printers...' is pressed to pull up a selection list of valid printers, and plotter RPLLOT1 is selected. When the Printers... button is pressed, the destination for the graph is automatically changed from the default of Terminal to Printer. The screen then appears as shown in figure 24. The cursor is now on 'OK,' and the user can exit this screen and produce a report by clicking on the left mouse button, or by pressing Enter. If the user changes his/her mind about producing a report, he/she can select 'CANCEL' and will be taken back to the main menu without producing a report.

```

Create Graph of Crime Statistics

Specify which types of crime you would like to graph:
  * Murder  ☐ Robbery  ☐ Rape  ☐ Assault  ☐ Burglary  ☐ Larceny  * Auto

Select which States:
  48  22  48  _  _  (? for list)

Select a destination for the graphs: ☐ Graphics
  ☐ My Graphics Terminal  * Printer ( RPLLOT1 ? for list)

When you are through, press PF3 to exit and run the program.
Press PF1 for Help.

```

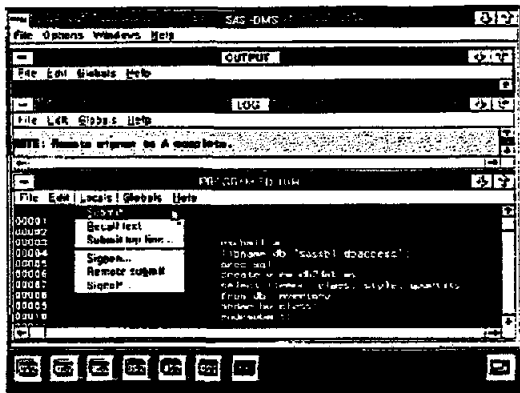
Figure 24

Pressing 'HELP' from within the graph module dialog window calls up help on that particular module. Pressing help again calls up help on the SAS system. These help screens are also available under 'Help' in the main menu Action Bar.

Appendix A uses screens from Microsoft Windows to illustrate the terms used in Graphical User Interfaces. However, both SAS and Microsoft Windows are CUA compliant, so the interfaces and the interaction with them are similar. CUA, or Common User Access, is the user-interface component of the SAA software standard endorsed by IBM. PCTools, the DOS shell under versions' 4.0 and 5.0, Presentation Manger under OS/2, X-Windows under UNIX, and CompuServe's Information Manager are other examples of CUA-compliant applications and operating systems. Once the user knows how to run one of these applications or operating systems, he/she will feel comfortable running the rest. There are some slight differences in running SAS on the mainframe, and Windows on the PC. These differences are due to the differences in the way mainframe terminals and personal computers handle input. (Mainframe terminals will not refresh the screen, or acknowledge input, until the user presses Enter.)

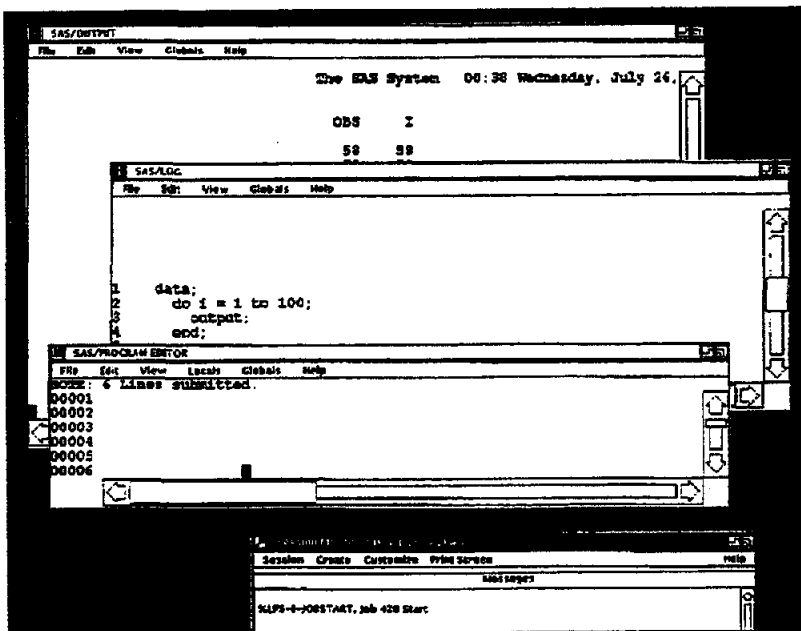
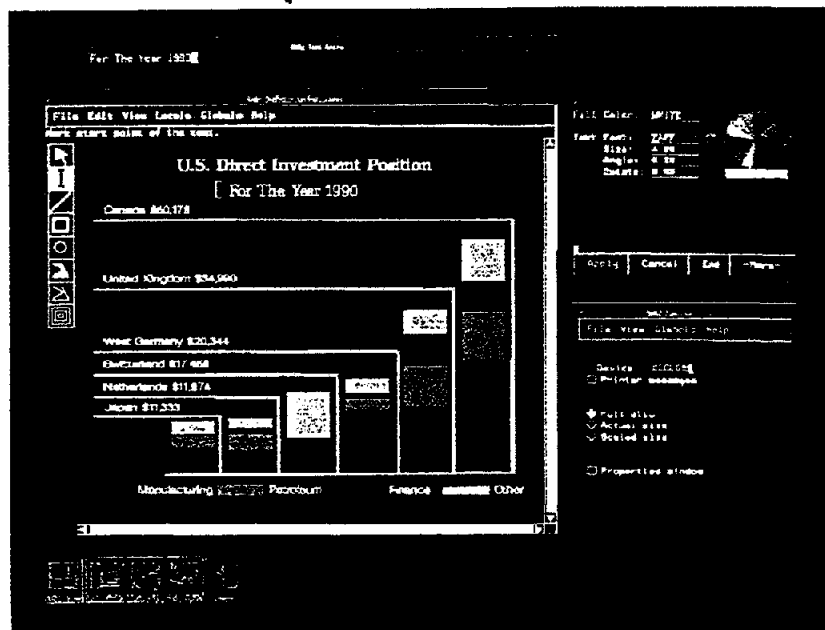
OTHER GRAPHICAL USER INTERFACES

The following are additional examples of Graphical User Interfaces. If the system supports SAS, then a SAS screen is shown. They are all printed at the same scale of 50% actual size to give some idea of the resolution of each display. Notice how the basic parts, such as icons, scroll bars, push buttons, banners, check boxes, and so on are similar for these interfaces.



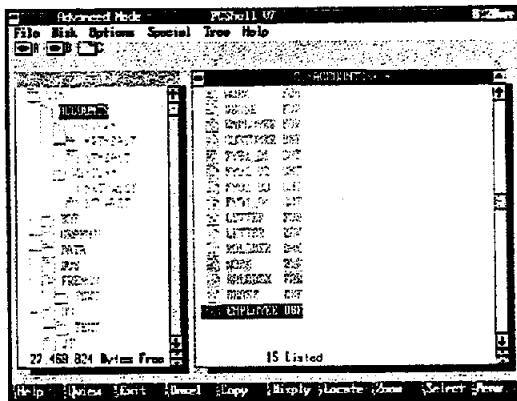
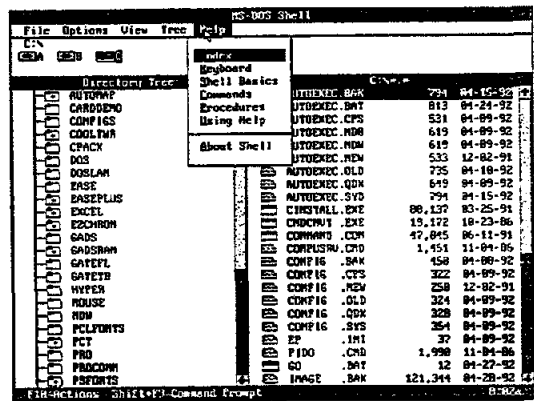
This illustration shows SAS 6.06 running under OS/2 version 1.x. The interface, and the interaction with it, is the same as SAS version 6.07 running under Microsoft Windows.

This illustration shows SAS version 6.07 running under X-Windows. X-Windows (which includes either a Motif interface, as shown, or Open Look) is slightly different from Windows. However, it is CUA compliant, along with OS/2 and SAS on the mainframe, so the interaction with it is also the same.



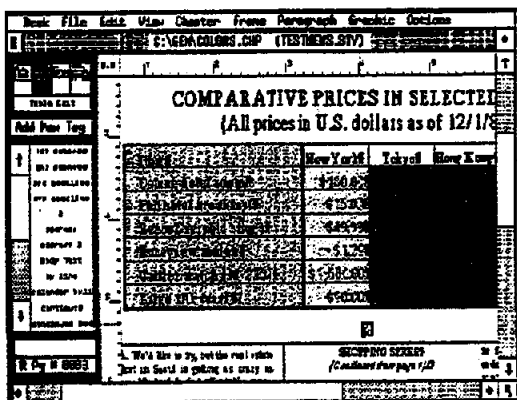
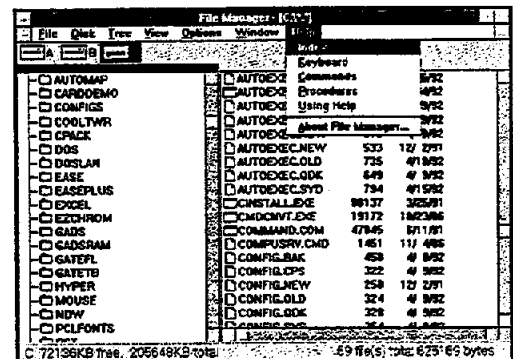
This illustration shows SAS running under DECwindows. Since DECwindows is not CUA compliant, the interface is slightly different. The interaction with it, however, is also the same. DECwindows is Digital Equipment Corp.'s interface for the VMS and ULTRIX operating systems.

This illustration shows the DOS shell on DOS version 5.0. Files can be accessed and programs ran under MS-DOS without having to know the line commands. The help menu is shown pulled down. Note that the help menu choices are the same as the mainframe SAS programs (Figure 5), and the other two screens shown.

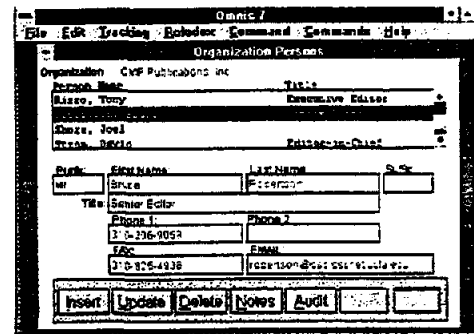
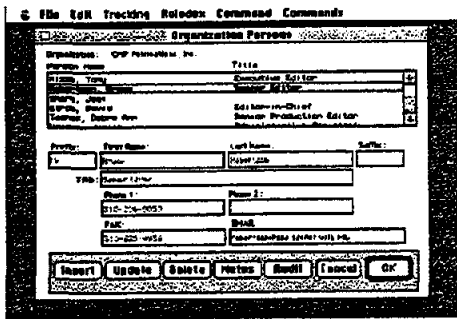


Compare the PCTools shell on the left to the DOS 5.0 shell above. Notice how the screen layout and pull-down menu are the same. Also, pressing F1 calls up help, F10 highlights the main menu (just like the IBM PC LAN menu), and F3 exits from PCTools, just like on the mainframe. Both the DOS 5.0 and PCTools shells are DOS programs with a character-based GUI that does not require Windows. The PCTools screen layout is the same--It even has the control icon in the upper

left hand corner of each window! To illustrate this point, compare it to the Windows File Manager screen at right. Again, the screen looks similar, the function keys are similar, all three have context sensitive help screens, and even the mouse operations are the same! For example, to copy a file or files (or even a directory) to another disk or directory, the file icons are highlighted and dragged to the destination directory. This demonstrates the good side of Graphical User interfaces in general and CUA in particular-- once the user is comfortable running one application, running other applications are easy; with the screen layout, keystrokes, and mouse operations the same for all applications.

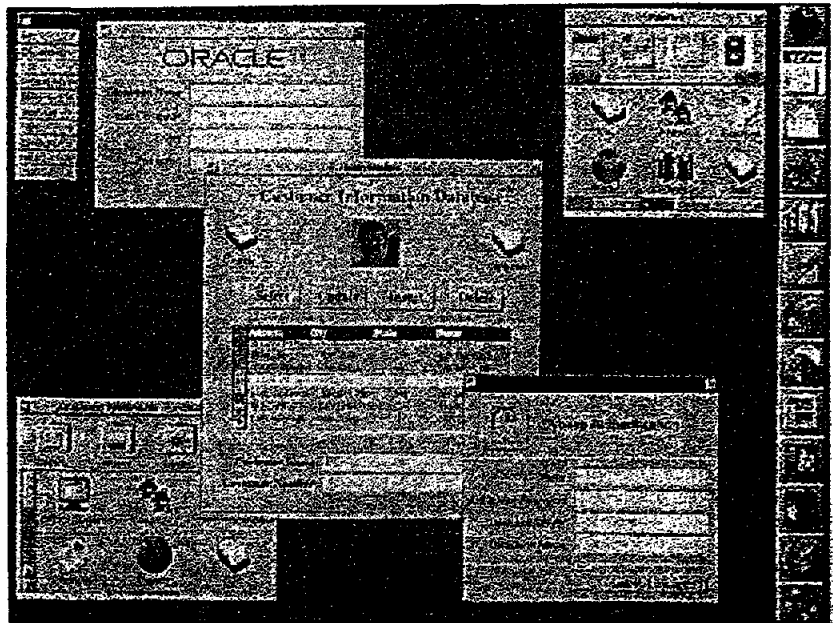


This illustration shows Ventura Publisher running under Digital Research's GEM. GEM was used by some DOS programs and Atari XL computers. Dr. DOS and a handful of DOS programs still use GEM; most are also available for Windows. GEM and the following interfaces are not CUA compliant, but they are similar in appearance. The way the mouse is used is also very similar.

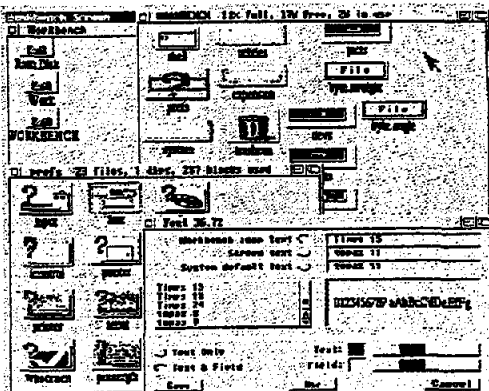


The above illustration on the left shows a database program running under the now-famous Apple Macintosh interface. Compare it to the illustration on the right showing the same program running under Microsoft Windows. Notice that they are similar in appearance.

When founder Steve Jobs left Apple, he started a new company around a computer called the NeXT. The object-oriented NeXTstep environment is shown. NeXT is currently working on porting NeXTstep to 486 machines.

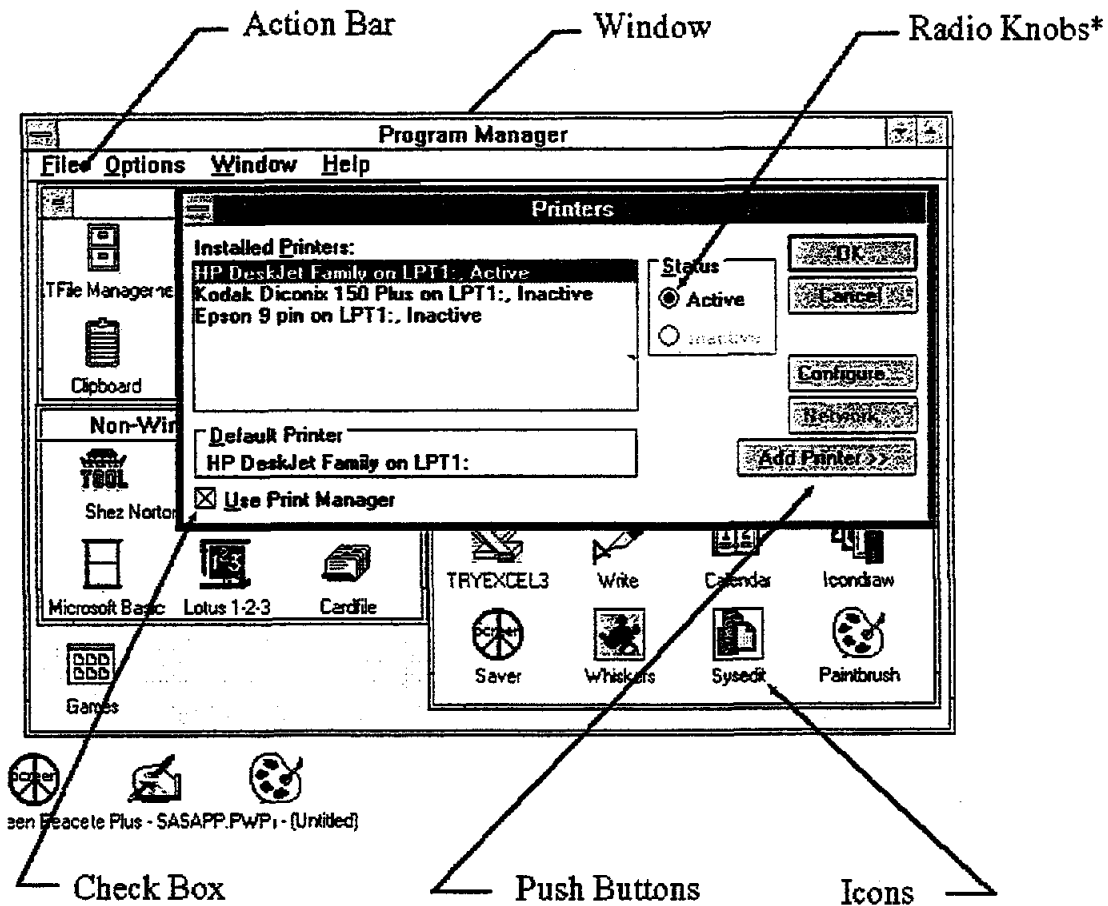


This illustration shows GEOS, running on a Commodore 64. This screen is included to illustrate that large amounts of memory or high display resolution are not necessarily needed to run a GUI, since the C-64 has only 64K of memory, and a screen resolution of 320 x 200! GEOS has been ported to the Apple II and IBM PC (GEOWorks). The illustration on the left is of the Commodore Amiga. The Amiga, like the Atari XL, Macintosh, and NeXT, has a Motorola 680x0 processor, and a built-in graphical operating system.



Appendix A

Illustration of Windowing (CUA) Terms



*So called because you can only make one selection at a time, like car radio station buttons.

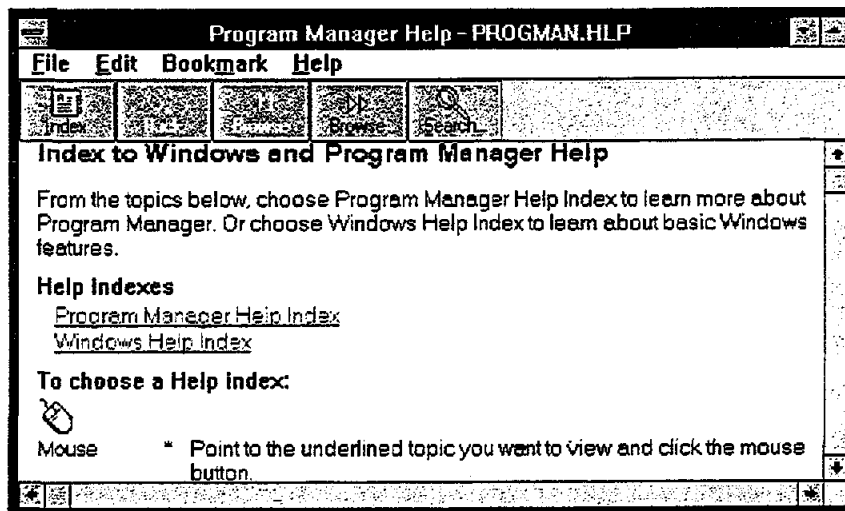
Note the **banner** on top of each window. (In Microsoft Windows the **banner** and **border** of the **active window** are highlighted. So in this example, the window titled 'Printers' is the **active window**.)

Selecting an item on the **action bar**, such as File, opens a **pull-down menu**. Then select an item on the **pull-down menu**, such as Open.

The **window** titled 'Printers' is an example of a **dialog box**. **Dialog boxes** cannot be resized (and sometimes cannot be moved).

The **window** titled 'Program Manager' is an example of a **standard** (or program) window. It can be moved by clicking on the banner and dragging it to a new location. It also can be resized by clicking on the border and dragging it to a new size.

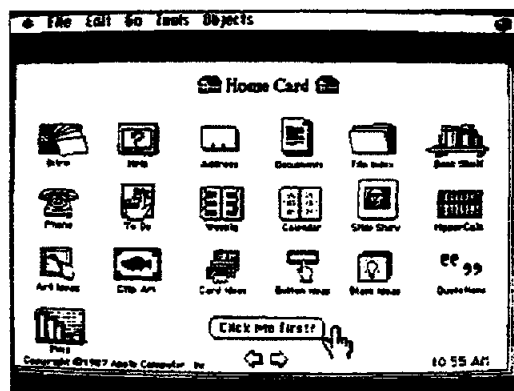
All windows can be shrunk down into **icons**. **Icons** allow the user see what windows are available without taking up space or confusing the screen with overlapping windows.



The above illustration shows a **Help Screen**. All CUA help screens are **context sensitive**, meaning help is immediately available on the item the user is currently working with. Also, additional help is available by clicking on the underlined words or pictures. This is known as **hypertext**, and Apple's Hypercard, shown below, was one of the first hypertext applications. This SAS 6.06 program and CUA-compliant programs offer an index as shown, where a topic is selected by tabbing to it and pressing enter, or by clicking on it with the mouse (see Figure 21). Note the push buttons for pulling up the index, browsing back and forth through help, or going back to a previous topic. System help is also available. This SAS 6.06 program has these buttons, plus an additional button for exiting help.

Note the **banners** in the upper left hand corner of SAS 6.06 windows. SAS 6.06 features **standard windows**, **help windows**, **dialog boxes** and **icons**.

Also shown above are **scroll bars**. **Scroll bars** appear along the right hand and bottom sides of the screen, and are used to scroll through a document using a mouse. SAS help screens and FSBROWSE screens have **scroll bars** on them.



Appendix B

The following source code was used to generate the PMENU WNDHELP.CBT. Change the catalog= option to create the PMENU in the current application catalog.

```
proc pmenu catalog=sas606.example desc'Pmenu for Help Screens';
  menu wndhelp;
  item 'Goback';
  item 'ExitHelp' selection=exithelp;
  item 'Index';
  item 'Prev' selection=prev;
  item 'More' selection=more;
  item 'Help' ;
  selection exithelp 'qcan';
  selection prev 'backward';
  selection more 'forward';
```

The following source code was used to generate the PMENU HEENDCAL.CBT. Change the catalog= option to create the PMENU in the current application catalog.

```
proc pmenu catalog=sas606.example
      desc"Pmenu displaying 'OK', 'Cancel' & 'Help'";
  menu heendcal;
  item 'OK' selection=OK;
  item 'Cancel';
  item 'Help' ;
  selection OK 'end';
```

The following source code was used to generate the PMENU OK.CBT. Change the catalog= option to create the PMENU in the current application catalog.

```
proc pmenu catalog=sas606.example desc"Pmenu displaying 'OK'";
  menu OK;
  item 'OK' selection=OK;
  selection OK 'end';
```

Appendix C

The following source code was used to generate the PMENU WNDHELP.CBT.
Change the catalog= option to create the PMENU in the current application catalog.

```
proc pmenu catalog=sas606.example;
  menu mainmenu;
  item 'File' menu=filemenu;
  item 'Options' menu=optionnu;
  item 'Globals' menu=globemen;
  item 'Help' menu=helpmenu;

  menu filemenu;
  item 'Update' selection=edit;
  item 'Browse' selection=browse;
  item 'Print' selection=print;
  separator;
  item 'Exit' dialog=leave;
  selection edit '2';
  selection browse '8';
  selection print '4';
  dialog leave 'bye';
  text #1 @4 ' ';
  text #2 @4 'Are you sure you want to exit the';
  text #3 @4 ' Crime Database Program? ';

  menu optionnu;
  item 'Generate Listing of Crimes' selection=listing;
  item 'Generate Crime Rate Report' selection=report;
  item 'Generate Graph Showing Crime Rate in City' selection=graph;
  selection listing '5';
  selection report '6';
  selection graph '7';

  menu globemen;
  item 'Menu Off' selection=menuoff;
  item 'ASSIST' selection=assist;
  item 'Clock';
  item 'Calculator';
  item 'Calendar' selection=calendar;
  item 'Notepad';
  item 'Libname';
  item 'Host Command' dialog=host;
  selection menuoff 'PMENU';
  selection assist '99';
```

```

selection calendar 'appointment';
dialog host 'tso @1';
  text #1 @4 '          ';
  text #2 @4 'Enter the Host (TSO) command:';
  text #3 @4 len=15;
  text #5 @4 'NOTE: If you do not specify a command,';
  text #6 @4 'you will go to the host operating system.';

menu helpmenu;
  item 'Index' selection=general;
  item 'Using Help' selection=helphelp;
  item 'SAS System Help' selection=sashelp;
  separator;
  item 'About CRIME' selection=about;
  selection general 'help';
  selection helphelp 'helphelp';
  selection sashelp 'sashelp';
  selection about 'about';

run;quit;

```