

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0383324 9

ORNL/TM-12373

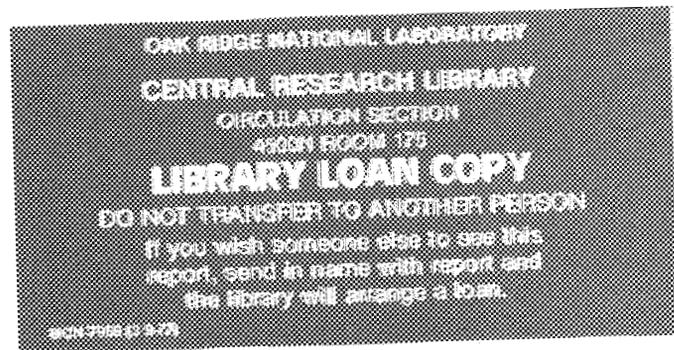
ornl

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

FASTPLOT—An Interface to Microsoft® FORTRAN Graphics

Richard C. Ward



MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DCSS contractors in print form at Office of Scientific and Technical Information, P.O. Box 601, Oak Ridge, TN 37831-0601; also available from (615) 576-8401, FT-616-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

This report was prepared by the contractor under the sponsorship of the United States Government and is hereby stated that the Government, and any agency thereof, nor any of its employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, including any product or product disclosed, or represents that its use would infringe on any trademark rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or approval by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

485
32

Computing Applications Division

**FASTPLOT - An Interface to
Microsoft® FORTRAN Graphics**

Richard C. Ward

Date Published - March 1994

Research sponsored by the Office of Health and Environmental Research, U.S. Department of Energy, Washington, DC 20545, and the Office of Radiation and Indoor Air, U. S. Environmental Protection Agency, Washington, DC 20460, under Interagency Agreement 1824-C148-A1, EPA NO. DW89934657-3.

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



CONTENTS

ACKNOWLEDGMENTS	v
ABSTRACT	vii
1. INTRODUCTION	1
2. GENERAL DESCRIPTION OF THE FASTPLOT LIBRARY	3
3. FPDEMO - ADDITIONAL FASTPLOT EXAMPLES	35
4. LIST OF FASTPLOT SUBROUTINE AND FUNCTION CALLS	107
5. FASTPLOT LIBRARY REFERENCE	109
Appendix A. SYMBOL TABLE FOR FASTPLOT	129
Appendix B. FASTPLOT EXTENDED KEY CODES	133
Appendix C. FASTPLOT RESERVED UNITS, COMMON BLOCKS, AND VARIABLE NAMES	135
Appendix D. DEFAULT COLORS FOR THE IBM PC	139
Appendix E. FASTPLOT COLOR MAPS	141

ACKNOWLEDGMENTS

The author wishes to thank Keith Eckerman of Health Sciences Research Division for his support in developing the FASTPLOT software. In addition, he thanks Jeffrey Ryman and Thomas Evers of Computing Applications Division for reviewing the manuscript and testing the FASTPLOT software. Thanks are also due to Mark Cristy of Health Sciences Research Division for comments on the FASTPLOT library. The symbol table used by FASTPLOT was originally obtained from Malcolm Patterson of Computing Application Division. The author thanks William Brosey, Donald Carpenter, and Mark Taylor of the Y-12 Manufacturing Technology Development Center for providing the X-ray tomography image shown in the text. Finally, the author would like to express his appreciation to Jan Anderson for her attention to the details in preparation of this manuscript and to Catherine Shappert for her attentive editing of the manuscript.

ABSTRACT

Interface routines to the Microsoft® FORTRAN graphics library¹ (GRAPHICS.LIB) are provided to facilitate development of graphics codes. These routines are collected into the FASTPLOT library (FASTPLOT.LIB). The FASTPLOT routines simplify the development of applications utilizing graphics and add capabilities not available in GRAPHICS.LIB, such as plotting histograms, splines, symbols, and error bars. Specifically, these routines were utilized in the development of the mortality data viewing code, MORTVIEW,² for the U.S. Environmental Protection Agency. Routines for color imaging,³ developed for use with the X-ray Computer Tomography (XCT) imaging code, and examples are also provided in the FASTPLOT library. Many example uses of FASTPLOT.LIB are contained in this documentation to facilitate applications development. The FASTPLOT.LIB library, source, and applications programs are supplied on the accompanying FASTPLOT diskette.

¹Microsoft® *FORTRAN Advanced Topics*, Version 5.1, 1991, pp. 179-352.

²R. C. Ward and K. F. Eckerman, *MORTVIEW - Software for Examining Mortality Data*, ORNL/TM-12713 (in progress).

³P. L. DeVries, *A First Course in Computational Physics*, Wiley, New York, 1994.

1. INTRODUCTION

FASTPLOT.LIB is a collection of FORTRAN-callable screen graphics routines for an IBM-compatible personal computer (PC) that serve as an interface to the Microsoft® FORTRAN graphics library GRAPHICS.LIB. The FASTPLOT library was originally created to facilitate development of the MORTVIEW code for the U.S. Environmental Protection Agency. The FASTPLOT routines simplify the development of applications utilizing graphics and add capabilities such as plotting histograms, splines, symbols, and error bars; and creating menus. This collection is not a comprehensive plotting package; the user will likely need to use other routines from the Microsoft® FORTRAN graphics library in addition to the FASTPLOT routines (e.g., to draw arcs, ellipses, or polygons).

The FASTPLOT library (FASTPLOT.LIB) contained on the disk accompanying this document, was compiled with Microsoft® FORTRAN Version 5.1 using the math co-processor library (compile switch /FPi87) and optimization (compile switch /Ox). The source for the FASTPLOT library (FASTPLOT.FOR), and the FASTPLOT Examples and Demo (FPDEMO), and the assembler routine (FPKEYINT.ASM) are also included on the disk.

In this document, FASTPLOT library routines are designated by capital letters (e.g., SYMBOL), Microsoft® FORTRAN graphics routines by bold, lower-case letters (e.g., **setvideomode**), and FASTPLOT routine parameters by italics (e.g., *icolor*).

2. GENERAL DESCRIPTION OF THE FASTPLOT LIBRARY

The FASTPLOT library is a collection of FORTRAN-callable screengraphics routines for an IBM-compatible personal computer which serve as an interface to the Microsoft® FORTRAN graphics library, GRAPHICS.LIB. The FASTPLOT routines simplify the development of applications utilizing graphics and add additional capabilities that include: histograms, splines, symbols, and error bars. In addition, two routines (QUERY and MENU) can be used to develop simple menus.

The library is not intended to be comprehensive; the user will likely need to use other routines from the Microsoft® FORTRAN graphics library in addition to the FASTPLOT routines (e.g., to set the color palette) to develop plotting applications. The FASTPLOT routines have proved very useful in developing plotting capabilities internal to a software package.

This section describes in general terms how to use the FASTPLOT routines in developing plotting applications. The section FASTPLOT LIBRARY REFERENCE contains detailed descriptions of each FASTPLOT routine.

GENERAL DEFINITIONS

Some variables have general definitions in the FASTPLOT routines. The color of a line, symbol, or font is designated *icolor*, which ranges from 0 to 15. A list of the default colors on the IBM PC is in Appendix D.

Many routines designate the type of an element or curve with the variable *itype*. In most cases *itype* is a two-digit number. In AXIS, TICKS and NUMBERS, the left digit of *itype* specifies the axis. In AXIS, the right digit of *itype* is zero. In TICKS, the right digit specifies inside or outside tick marks. In NUMBERS, the right digit specifies linear or log axis. In the routines CURVE, SYMBOL, and ERRBAR, *itype* designates the type of axis. The left digit of *itype* designates the type of axis for the horizontal plot variable (*X*) and the right digit of *itype* the type of axis for the vertical plot variable (*Y*).

Routines MFRAME, LINETYPE, and CURSOR use a single-digit *itype*. In MFRAME, *itype* specifies the type of frame (single or double). In LINETYPE, *itype* specifies the type of line (solid, dash, dot). In CURSOR, *itype* specifies the type of cursor (underline, double underline, block).

The text row and column for text routines is designated by (*ROW*, *COL*).

The arrays defining the curve to be plotted are designated (*X* and *Y*) in routines CURVE, SYMBOL, and ERRBAR. The dimension of the *X* and *Y* arrays is designated by *npoints*. The minimum and maximum range of (*X* and *Y*) in ERRBAR is (given by the *Xrange* and *Yrange* arrays). The dimensions *Xrange* and *Yrange* are (2,*npoints*).

The port, or viewport, is defined by screen pixel coordinates and is also referred to as the SCREEN COORDINATES. The port is defined by giving the upper- left pixel location (x,y) and the lower-right pixel location (x,y).

The window, or graphics window, is defined by the user's graph coordinates, also referred to as the VIRTUAL COORDINATES. The window is defined by giving the upper-left virtual coordinate location (x,y) and the lower right virtual location (x,y). The y axis is assumed to increase from screen bottom to screen top.

USE OF FASTPLOT LIBRARY

This section describes the general use of FASTPLOT routines and gives specific examples of their use. The user should read the FASTPLOT LIBRARY REFERENCE for a detailed description of the capabilities of the routines.

Before initializing graphic or text modes, the user should load the fonts and symbols into memory. If any text is to be placed on the graph, the user must call REG_FONT to register the fonts (i.e., load them into memory). The user should then select a specific font by calling the routine SET_FONT, with a specified font selection (*ifont* = 1-6) and font height (*ifonth*) and width (*ifontw*). The fonts available for a given typeface are described under subroutine REG_FONT. The SET_FONT subroutine will select the best fit if the height and width specified are not exactly those shown for bit-mapped fonts.

If the user intends to use FASTPLOT symbols, the routine SET_SYMBOL should be called to load the FASTPLOT symbol table. The selection of the symbol to use is usually done when the curve is plotted (see below). For example, to load the fonts and symbols, and select the bit-mapped Helv font (*ifont* = 2) with height of 12 pixels and width of 7 pixels, use the following:

```
CALL REG_FONT
CALL SET_FONT (2, 12, 7)
CALL SET_SYMBOL
```

The routine REG_FONT assumes the Microsoft® font files are in the current directory unless the user specifies the location (path) in the file FP_FONT.LOC. Likewise, the routine SET_SYMBOL assumes the file FP_SYMBL.TAB is in the current directory unless the user specifies the location (path) in the file FP_FONT.LOC.

The user should next use the GET_DISPLAY routine to determine the system's graphics display capabilities. This routine returns the display type: CGA (for CGA graphics adapter and display or EGA graphics adapter and CGA display), EGA (for EGA graphics adapter and EGA display), VGA (for VGA graphics adapter and display), or TXT (if no graphics adapter detected). The routine returns EGA if a HERCULES graphics adapter is detected. For example, if you wish to plot only when the display is a VGA, use the following:

```
CALL GET_DISPLAY (display)
IF (display .NE. 'VGA') STOP 'Display is not VGA.'
```

The user then calls BGNPLOT to initialize the graphics mode. BGNPLOT sets the video mode to the maximum resolution allowed by the graphics adapter. The background color is set to *icolor*. Before placing anything on the plot, the user should clear the screen using the graphics routine CLEAR. CLEAR calls Microsoft® FORTRAN graphics subroutine *clearscreen* with the parameter \$GCLEARSCREEN which clears the entire graphics screen.

To initialize the graphics with a black background and clear the screen, use the following:

```
CALL BGNPLOT (0)
CALL CLEAR
```

At the end of the program, call ENDPLOT to return to text mode. The program halts when ENDPLOT is called. The array *key* contains keys which when pressed will clear the screen.

```
CALL ENDPLOT (beep, numkey, key)
```

After calling BGNPLOT and clearing the screen and before defining the screen port, the user must know the screen *width* and *height* in pixels. These dimensions can be determined by calling SCREEN:

```
CALL SCREEN (width, height)
```

At this point the user should establish the port and window. The call to PORT is as follows:

```
CALL PORT (0, 0, width, height)
```

The port, or viewport, is defined by screen pixel coordinates, which are also referred to as the SCREEN COORDINATES. The port is defined by giving the upper-left pixel location (*iul_x, iul_y*) and the lower-right pixel location (*ilr_x, ilr_y*). This convention is the same as the one used in the Microsoft® FORTRAN graphics subroutine *setviewport* called by this routine. Assuming the user wishes to set the port to the entire screen, the upper-left corner is (0,0) and the lower-right corner is (*width, height*) as returned from call to SCREEN.

The window, or graphics window, is defined by the user's graph coordinates. It is also referred to as the VIRTUAL COORDINATES. The window is defined by giving the upper-left corner virtual location (*ul_x, ul_y*) and the lower-right virtual location (*lr_x, lr_y*). This convention is the same as the one used in the Microsoft® FORTRAN graphics subroutine *setwindow* called by this routine. The parameter (*finvert*) passed to the graphics subroutine *setwindow* is set to TRUE so that the y-axis increases from the screen bottom to the screen top.

Generally, two calls to WINDOW are made: the first is used to set a larger window for writing the text LABELS. The second is used to set the user's graph coordinates for the actual plot. Assuming that the upper-left corner is defined by (*Xmin, Ymax*) and the lower-right corner by (*Xmax, Ymin*), then the call to WINDOW would be the following:

CALL WINDOW (*Xmin, Ymax, Xmax, Ymin*)

Now the user is ready to create a plot. When drawing frames, tick marks and when plotting curves, symbols with or without error bars, one must remember to set the line type (SOLID, DASH, DOT-DASH). To select a solid line, call LINETYPE with *itype* set to 0.

CALL LINETYPE (0)

To create a plot, first use the routine FRAME to draw a frame around the existing graphics window (virtual coordinates) as specified by the last call to subroutine WINDOW. The color of the frame is determined by the parameter *icolor* which can range from (0 to 15).

For example, to draw a yellow frame, use:

CALL FRAME (14)

Alternatively, one could draw each axis separately by calling the routine AXIS, with *icolor* and the axis type, *itype*, specified:

	<i>itype</i>
X-axis along the bottom	10
Y-axis along the left side	20
X-axis along the top	30
Y-axis along the right side	40

For example, to draw a yellow frame with AXIS, use:

```
CALL AXIS (14,10)
CALL AXIS (14,20)
CALL AXIS (14,30)
CALL AXIS (14,40)
```

Labels can be placed on the graphics window using the LABEL routine. The call to LABEL is as follows:

CALL LABEL (*icolor, ibackcolor, iback, icenter, xloc, yloc, string*)

This routine puts a label *string* at SCREEN COORDINATE location (*xloc, yloc*) in graphics mode. Note that this is at SCREEN COORDINATE location, not VIRTUAL COORDINATE LOCATION! The color of the text is set by *icolor*, which can range from (0 to 15). If *icenter* is 1, the string is centered; if *icenter* is 0, the string is placed a location, *xloc*. If *iback* is 1, the background will be colored with the color *ibackcolor*. This allows labels to be printed on top of a plotted curve. Legends placed on a plot must be printed after the curve is drawn.

For example, to place the string 'HELLO WORLD' at SCREEN COORDINATE location (200,200) using red as the color, use:

```
string = 'HELLO WORLD'  
CALL LABEL (4, 0, 0, 0, 200, 200, string)
```

The parameter *string* is a character variable of length 200.

Example 1:

The simple example below uses the FASTPLOT graphics routines discussed to this point. To create a graphics window in which the label 'HELLO WORLD' is printed at SCREEN COORDINATES (160,200). Bright white (*icolor* = 15) is used to draw the frame and the text. The background screen color is set to black (*iback* = 0). After halting, the user is told to type q (for quit) to clear the screen and return to default (text) mode.

```
c Simple test of FASTPLOT GRAPHICS routines
  character*200 string
  character*3   display
  integer*4    width,height
c
  call REG_FONTS
  call SET_FONT(2,15,8)
  call SET_SYMBOL
c
  call GET_DISPLAY(display)
  if (display.NE.'VGA') STOP 'Display is not VGA.'
c
c Background color (iback) set to black (0).
  iback = 0

c Initialize graphics mode
  call BGNPLOT(iback)

c Set PORT and WINDOW for labels
  call SCREEN(width,height)
  call PORT(0,0,width,height)
  call WINDOW(0.,1000.,1000.,0.)

  call LINETYPE(0)
  call FRAME(15)

c Place labels
  string = 'HELLO WORLD'
  call LABEL(15,0,0,0,160,200,string)

  string = 'TYPE q TO QUIT'
  call LABEL(15,0,0,0,160,230,string)
c
c Return to text mode
  call ENDPLOT(.TRUE.,1,'q')
c
  STOP
  END
```

This example produces the output shown in Fig. 1.

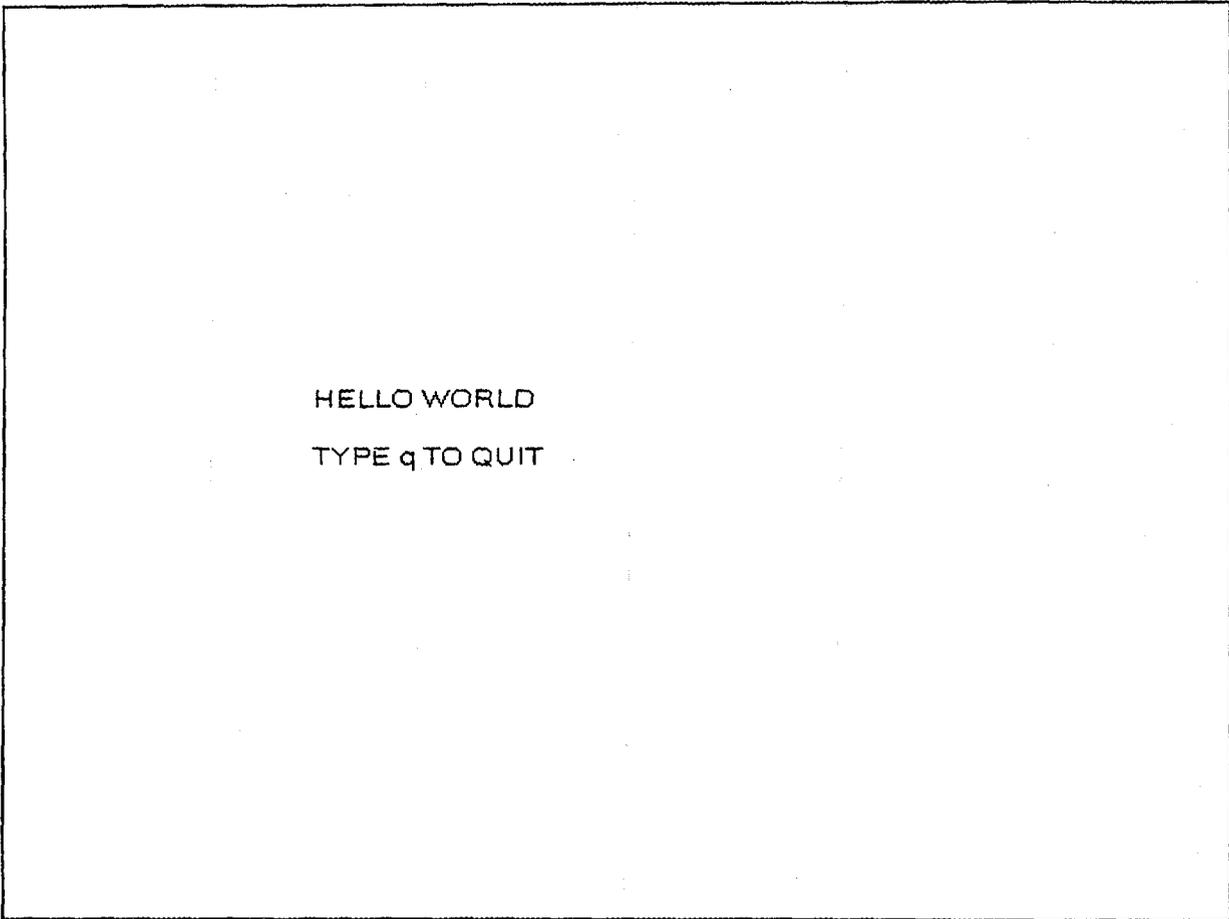


Fig. 1. Screen output for Example 1.

After placing labels on the plot, the user can set the plot window and viewport. A grid can be placed on the plotting region by a call to GRID. The color of the grid is determined by *icolor* and the type by *itype* (1 - horizontal, 2 - vertical). The major spacing of the grid is determined by *step*. If the user wishes grid lines at minor tick marks, specify *minor* in the same way as for TICKS (see below).

CALL GRID (*icolor, itype, step, minor*)

Tick marks and numbers are placed on a specified axis using the TICKS and the NUMBERS routines, respectively. The calls to TICKS and NUMBERS take the following forms:

CALL TICKS (*icolor, itype, tpix_h, tpix_w, step, minor*)
 CALL NUMBERS (*icolor, itype, form, step, skip*)

The color of the tick marks or numbers is determined by the parameter *icolor* which can range from 0 to 15. The placing of the tick marks or numbers is determined by the parameter *step*.

For tick marks, the height (roughly in pixels) of the X-axis tick marks is determined by the parameter *tpix_h*. The width (roughly in pixels) of the Y-axis tick marks is determined by the parameter *tpix_w*. No minor tick marks are placed if *minor* is zero. For a linear plot, minor tick marks are placed every *step/minor*, for positive values of *minor*. For a log plot, specify *minor* as -1 to obtain minor tick marks.

For tick marks, the axis is specified by the parameter *itype*.

	<i>itype</i>
X-axis, bottom of graph	
Inside axis	10
Outside axis	11
Both sides of axis	12
Y-axis, left side of graph	
Inside axis	20
Outside axis	21
Both sides of axis	22
X-axis, top of graph	
Inside axis	30
Outside axis	31
Both sides of axis	32
Y-axis, right side of graph	
Inside axis	40
Outside axis	41
Both sides of axis	42

Placement of numbers on an axis is specified by *itype*, defined as follows:

	<i>itype</i>
linear X-axis	11
log X-axis	12
linear Y-axis	21
log Y-axis	22

The numbers are placed along the axis at the interval specified by the parameter *step*. The parameter *form* is the format for the numbers (e.g., *form* = 'F5.1'). If *skip* is set to .TRUE., the first number on the axis is not displayed.

Now the user is ready to plot a curve with or without symbols or error bars. Symbols and error bars must be placed prior to drawing the curve. Symbols can be plotted with or without error bars. The SYMBOL and ERRBAR routines are separate routines, but function similarly. To use either, the user must first select a symbol and specify the height (and width) of the symbol. This step is done with the routine MARKER.

MARKER is used to select a particular symbol to use with either the SYMBOL or the ERRBAR routines. The user specifies the symbol *id* and the symbol *height* (roughly in pixels).

CALL MARKER (*id*, *height*)

The default symbols available with FASTPLOT are the following:

<i>id</i>	SYMBOL
0	open square
1	open circle
2	open triangle
3	vertical cross (+)
4	lazy cross (x)
5	open diamond
6	open inverted triangle
7	filled square
8	filled circle
9	filled triangle
10	filled diamond
11	filled inverted triangle
12	horizontal bar (error bar)
13	vertical bar (error bar)

For example, to use the filled diamond symbol of roughly 12 pixels height in plotting a symbol curve, call subroutine MARKER as follows:

CALL MARKER (10, 12)

Now the user can plot the selected symbol using subroutine SYMBOL or error bars using subroutine ERRBAR at locations determined by the arrays X and Y. The routine SYMBOL plots a symbol at every *isym* value of the arrays X and Y specified in the existing graphics window (virtual coordinates). ERRBAR plots an error bar at every *ierr* value of the arrays X and Y specified in the existing graphics window (virtual coordinates). The number of points in the arrays is *npoints*. The color of the symbol is determined by the parameter *icolor*, which can range from 0 to 15. The placement of symbols is controlled by the parameter *itype*, which takes on the same values as for the subroutine CURVE (see below), except that values of 31 and 55 are not used.

For a call to SYMBOL, the symbol used is determined by the previous call to subroutine MARKER. Filled symbols are noted in the symbol table (FP_SYMBL.TAB) and are filled using the Microsoft® FORTRAN graphics call `floodfill_w`. When using filled symbols on top of a curve, call SYMBOL prior to calling CURVE.

For a call to ERRBAR, the symbol used at the top/bottom of a vertical error bar and at the left/right of a horizontal error bar is determined by the previous call to MARKER. The range of the horizontal error bar is specified by *Xrange(1,npoints)* and *Xrange(2,npoints)*. The range of the vertical error bar is specified by *Yrange(1,npoints)* and *Yrange(2,npoints)*. The calls to SYMBOL and ERRBAR are as follows:

```
CALL SYMBOL (icolor, itype, isym, npoints, X, Y)
CALL ERRBAR (icolor, itype, isym, npoints, X, Y, Xrange, Yrange)
```

Finally, subroutine CURVE is called to draw a curve through the points given in the coordinate arrays X and Y, and specified in the existing graphics window (virtual coordinates). The number of points in the arrays is passed as the parameter *npoints*. The color of the curve is determined by the parameter *icolor*, which can range from 0 to 15. The type of curve is specified by *itype*. The following types are allowed:

<i>itype</i>	Curve type	
	<u>X-axis</u>	<u>Y-axis</u>
11	linear	linear
12	linear	log ₁₀
21	log ₁₀	linear
22	log ₁₀	log ₁₀
31	histogram	linear
55	linear	linear (spline fit)

Note that these are the same curve types as allowed for SYMBOL and ERRBAR with the addition of histogram (31) and spline (55). If *itype* is 31, CURVE plots a histogram versus linear x. The histogram spacing, *xstep*, must be provided. In addition, the user must specify *fill* as either .TRUE. for filled histogram or .FALSE. for unfilled histogram. If *itype* is 55, CURVE plots a B-spline through the points given by X and Y. The spline fit is restricted to 2000 points. The call to CURVE is as follows:

CALL CURVE (*icolor, itype, fill, xstep, npoints, X, Y*)

Often it is useful to display a plot, halt to allow the user to view the screen, and then clear the screen for the next plot. The routine HALTCLR was created to perform these functions. Upon completion of all plotting, the user should call ENDPLOT to return the screen to text mode.

Examine the FASTPLOT demo (see Sect. 3) for use of the LABEL, GRID, TICKS, NUMBERS, SYMBOL, ERRBAR, and CURVE routines.

HALTCLR and ENDPLOT pause and wait for the user to type one of the characters in the array *key*, which is dimensioned to the number of keys *numkey* which have been specified.

Examples for the values of the array *key* are: 'Q', 'q', the escape key [CHAR(27)] and the return key [CHAR(13)]. When one of the specified keys is pressed, the subroutine clears the screen. HALTCLR functions exactly like ENDPLOT except that it clears the screen without restoring the default screen mode. For example, to halt without beeping and wait until the user type 'q', use:

```
CALL HALTCLR (beep, numkey, key)  
CALL ENDPLOT (beep, numkey, key)
```

For example, to beep and wait until the user types 'q', use:

```
CALL HALTCLR (.TRUE., 1, 'q')
```

Finally, the user can free memory allocated to the graphics fonts by calling the routine UNREG_FONTS.

```
CALL UNREG_FONTS
```

Example 2:

The following is an example using the FASTPLOT graphics routines GRID and CURVE. A graphic window is created, and the label 'SIMPLE EXAMPLE PLOT' is centered at vertical SCREEN COORDINATE of 10. The aspect ratio relative to a VGA monitor, *FP_ASPCT*, which is computed when routine BGNPLOT is called, is passed through common FP_INFO. This is used here to control placement of the labels on an EGA monitor. A second viewport and graphics window are defined for the plot itself. A red grid (*icolor* = 4) is placed on the plot frame. The subroutine CURVE is called to plot the data defined by the *X* and *Y*. arrays Bright white (*icolor* = 15) is used to draw the frame and text; light yellow (*icolor* = 14) to draw the curve. The background screen color is set to black (*iback* = 0). After halting, the user is told to type q (for quit) to clear the screen and return to default (text) mode.

```
c Simple test of FASTPLOT GRAPHICS routines
  character*200 string
  character*3   display
  integer*4    width,height
  real*4       X(11),Y(11)
c Aspect ratio is stored in FP_INFO after call to BGNPLOT.
  character*3   FP_DISPLY
  real*4       FP_ASPCT
  common /FP_INFO/ FP_DISPLY, FP_ASPCT

c define plot data
  data X /0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10./
  data Y /0., 2., 4., 6., 8., 10., 8., 6., 4., 2., 0. /

c
  call REG_FONTS
  call SET_FONT(2,15,8)
  call SET_SYMBOL

c
  call GET_DISPLAY(display)
  if (display.NE.'VGA') STOP 'Display is not VGA.'

c
c Initialize graphics mode
c Background color (iback) is set to black (0).
  iback = 0
  call BGNPLOT(iback)
  call SCREEN(width,height)

c Set PORT and WINDOW for title
  call PORT(0,0,width,height)
  call WINDOW(0.,1000.,1000.,0.)
```

```

call LINETYPE(0)
call FRAME(15)
c
string = 'SIMPLE EXAMPLE PLOT'
IY = 10 * FP_ASPCT
call LABEL(15,0,0,1,0,IY,string)
string = 'TYPE q TO QUIT'
IY = 455 * FP_ASPCT
call LABEL(15,0,0,1,0,IY,string)
c
c Set PORT and WINDOW for plot
call PORT(30,30,width-30,height-30)
call WINDOW(0.,10.,10.,0.)

c Draw grid
call GRID(4,1,1.,1)
call GRID(4,2,1.,1)

call FRAME(15)
c
c Draw curve
npoints = 11
call CURVE(14,11,.FALSE.,0 npoints,X,Y)
c
c Halt until 'q' is pressed, Restore text mode
call ENDPLOT(.TRUE.,1,'q')
c
STOP
END

```

This example produces the output shown in Fig. 2.

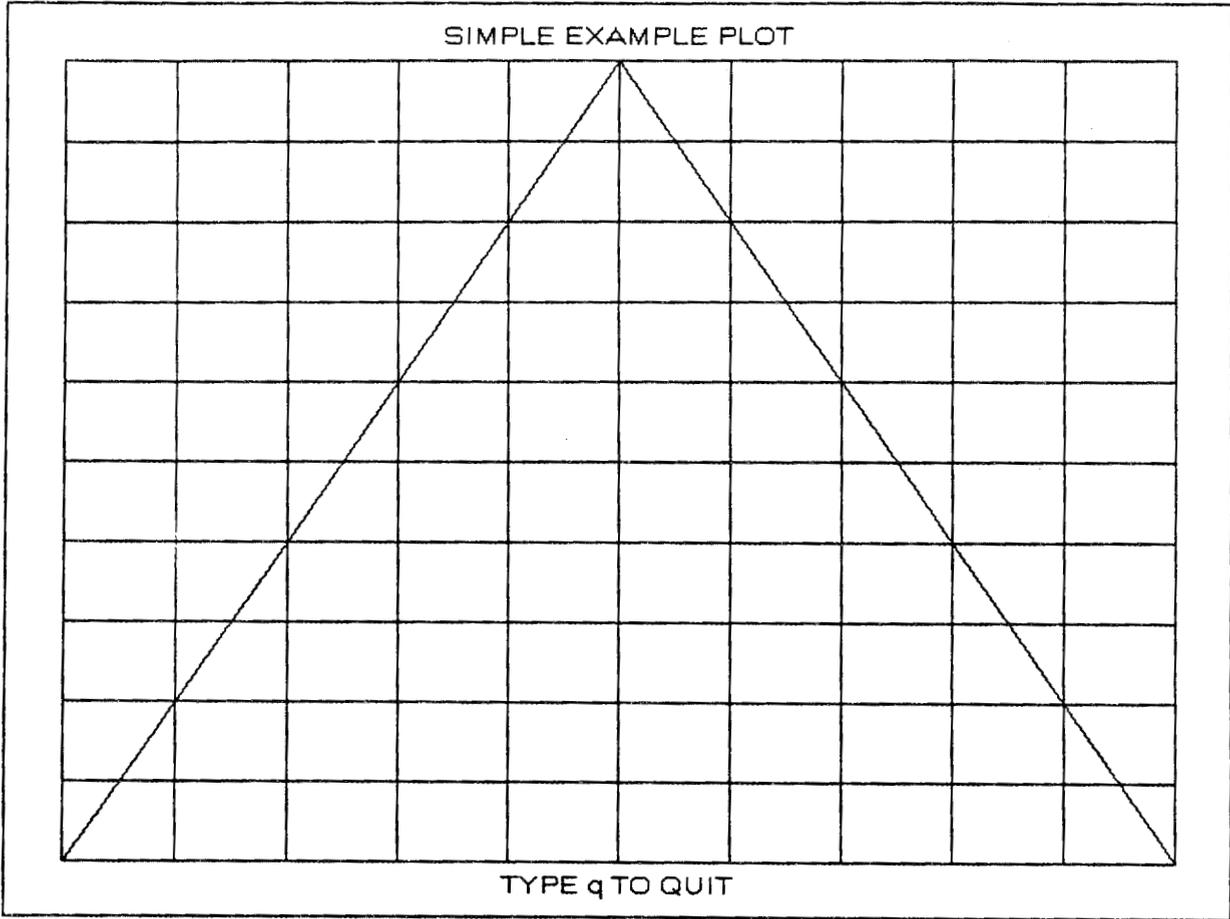


Fig. 2. Screen output for Example 2.

Example 3:

The following is an example using the FASTPLOT graphics routines TICKS and NUMBERS. A graphics window is created, and the label 'SIMPLE EXAMPLE PLOT' is centered at vertical SCREEN COORDINATE of 10. Again, the aspect ratio relative to a VGA monitor, *FP_ASPECT*, is used to control placement of the labels on an EGA monitor. A second viewport and graphics window are defined for the plot itself. A red grid (*icolor* = 4) is placed on the plot frame. Tick marks and numbers are drawn at every interval specified by *step*. The subroutine CURVE is called to plot the data defined by the *X* and *Y* arrays. Bright white (*icolor* = 15) is used to draw the frame and text; light yellow (*icolor* = 14) to draw the curve. The background screen color is set to black (*iback* = 0). After halting, the user is told to type q (for quit) to clear the screen and return to default (text) mode.

```
c Simple test of FASTPLOT GRAPHICS routines - TICKS and NUMBERS
  character*200 string
  character*3  display
  character*1  key(4)
  integer*4   width,height
  real*4      X(11),Y(11)
c Aspect ratio is stored in FP_INFO after call to BGNPLOT.
  character*3  FP_DISPLAY
  real*4      FP_ASPECT
  common /FP_INFO/ FP_DISPLAY, FP_ASPECT

c define plot data
  data X /0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10./
  data Y /0., 2., 4., 6., 8., 10., 8., 6., 4., 2., 0. /
c action keys
  key(1)='q'
  key(2)='Q'
  key(3)=CHAR(13)
  key(4)=CHAR(27)

c
  call REG_FONTS
  call SET_FONT(2,15,8)
  call SET_SYMBOL

c
  call GET_DISPLAY(display)
  if (display.NE.'VGA') STOP 'Display is not VGA.'

c
c
c Initialize graphics mode
c Background color (iback) is set to black (0).
  iback = 0
  call BGNPLOT(iback)
  call SCREEN(width,height)
```

```

c Set PORT and WINDOW for title
  call PORT(0,0,width,height)
  call WINDOW(0.,1000.,1000.,0.)

  call LINETYPE(0)
  call FRAME(15)
c
  string = 'SIMPLE EXAMPLE PLOT'
  IY = 10 * FP_ASPCT
  call LABEL(15,0,0,1,0,IY,string)
  string = 'Y axis'
  IY = 20 * FP_ASPCT
  call LABEL(15,0,0,0,10,IY,string)
  string = 'X axis'
  IY = 455 * FP_ASPCT
  call LABEL(15,0,0,1,0,IY,string)
c
c Set PORT and WINDOW for plot
  call PORT(60,60,width-60,height-60)
  call WINDOW(0.,10.,10.,0.)

c Draw grid
  call GRID(4,1,1.,1)
  call GRID(4,2,1.,1)

c Draw frame
  call FRAME(15)
c Draw tick marks
  call TICKS(15,11,8,6,1.,0)
  call TICKS(15,21,8,6,1.,0)
  call TICKS(15,31,8,6,1.,0)
  call TICKS(15,41,8,6,1.,0)
c Place numbers on axes
  call NUMBERS(14,11,'F5.1 ',1.,.FALSE.)
  call NUMBERS(14,21,'F5.1 ',1.,.FALSE.)
c
c Draw curve
  npoints = 11
  call CURVE(14,11,.FALSE.,0, npoints,X,Y)
c
c Halt until 'q','Q',<Esc>, or <Enter> is pressed, Restore text mode
  call ENDPLOT(.TRUE.,4,key)
c
  STOP
  END

```

This example produces the output shown in Fig. 3.

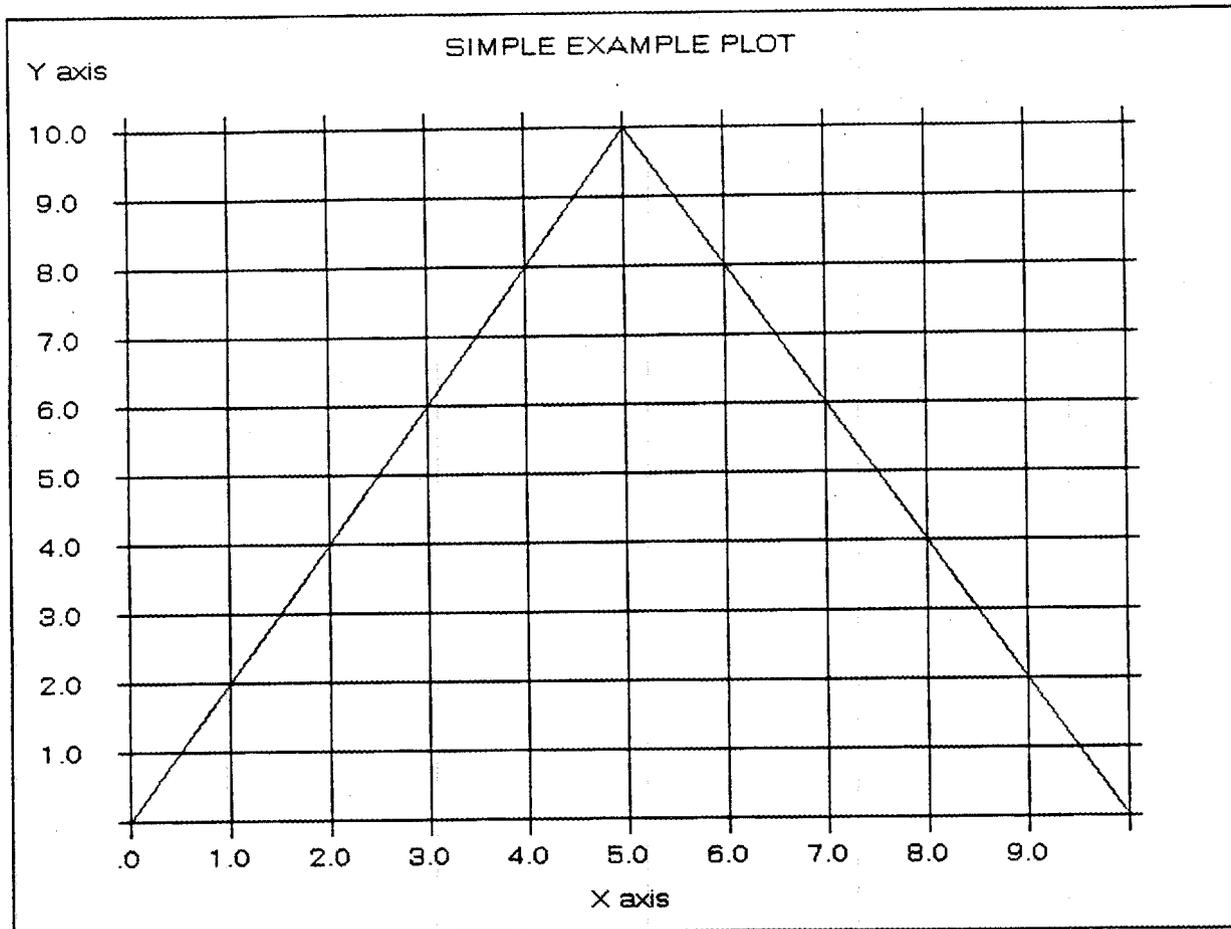


Fig. 3. Screen output for Example 3.

The following routines are used in creating menus: BGNTTEXT, CURSOR, TCLEAR, MFRAME, TLABEL, MENU, QUERY, and ENDTEXT.

The first step in using text routines, such as MENU and QUERY, is to call BGNTTEXT, which saves the original foreground and background colors, clears the screen and sets the background color to *icolor*. It serves a similar purpose as BGNPLOT for graphics mode. Subroutine CURSOR should be called to hide the cursor. The cursor type, *itype* (0 - no cursor, 1 - underline, 2 - double underline, or 3 -block) should be set to 0 to hide the cursor.

To initialize the test mode with a blue background and hide the cursor, use the following command:

```
CALL BGNTTEXT (1)
CALL CURSOR (0)
```

At the end of the program, to restore the *cursor* and the original foreground and background colors, use the following commands:

```
CALL CURSOR (1)
CALL ENDTEXT
```

After calling BGNTTEXT and hiding the cursor, the user should then clear the screen using either the graphics routine CLEAR or, if dealing only with text menus, the text routine TCLEAR, which clears the text screen by sending extended control characters to the ANSISYS driver. The driver file ANSISYS must be loaded in CONFIG.SYS.

```
CALL CLEAR
or
CALL TCLEAR
```

The routine MFRAME draws a frame around text, such as entries in a menu. The call to MFRAME is as follows:

```
CALL MFRAME (icolor, itype, icen, ROWTL, COLTL, ROWBR, COLBR)
```

The row and column of the top-left corner is specified by (*ROWTL, COLTL*); the row and column of the bottom-right corner is specified by (*ROWBR, COLBR*). The color of the frame is determined by the parameter *icolor*, which can range from 0 to 15. The type of frame, single (1) or double (2), is specified by *itype*. The frame will be centered horizontally if *icen* is 1, in which case *COLTL* and *COLBR* are ignored. If *icen* is 0, the frame is placed using the top-left corner (*COLTL*) and bottom-right corner (*COLBR*) values.

Subroutine TLABEL puts a label (text string) at text location (*ROW, COL*). The call to TLABEL is a follows:

```
CALL TLABEL (icolor, ibackcolor, iback, icenter, ROW, COL, string)
```

The color of the text is set by *icolor*, which can range from 0 to 15. If *icenter* is 1 the string is centered; if *icenter* is 0, the value *COL* is used for horizontal placement. If *iback* is 1, the background will be colored with the color *ibackcolor*.

Example 4:

The following is an example using the FASTPLOT text routines discussed to this point. This example creates a text window in which the label 'HELLO WORLD' is printed beginning at text row and column (10,20). Bright white (*icolor* = 15) is used to draw the frame and the text. The background screen color is set to black (*iback* = 0). After halting, the user is told to type q (for quit) to clear the screen and return to default (text) mode.

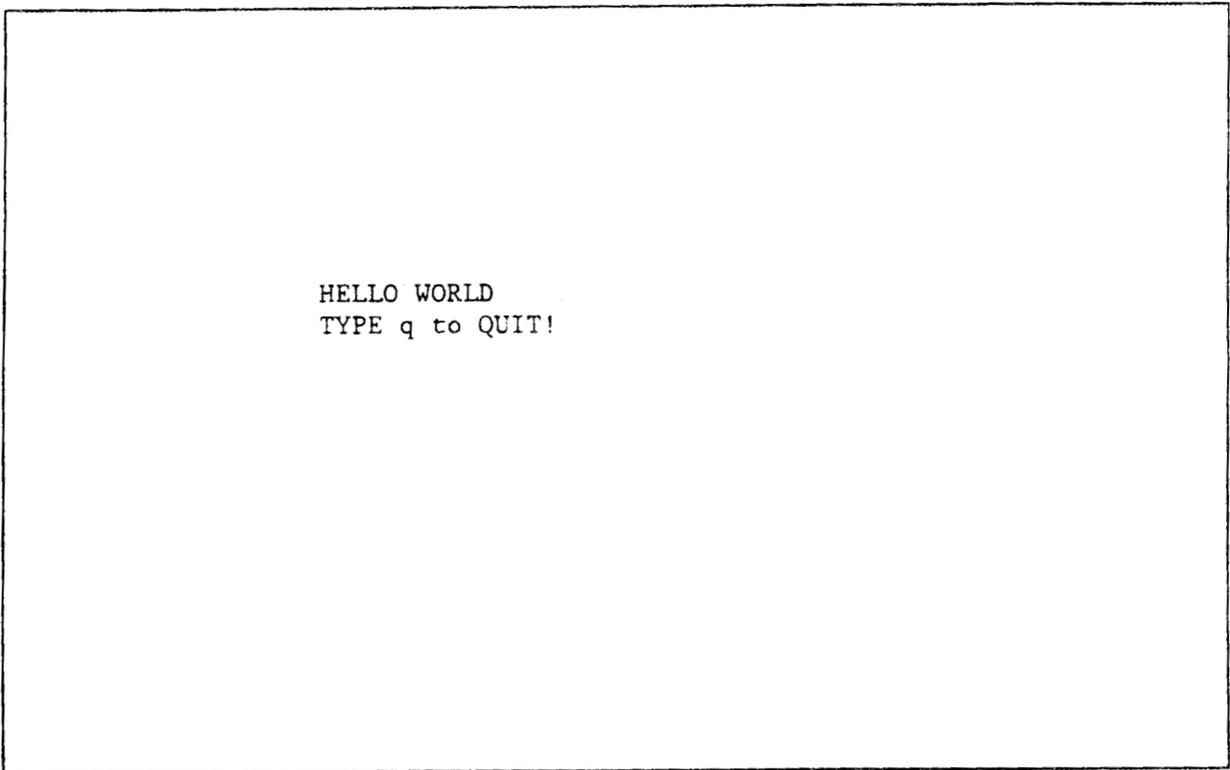
```
c Simple test of FASTPLOT TEXT routines
  character*200 string
c
c Background color (iback) set to black (0).
  iback = 0

c Initialize text mode
  call BGNTXT(iback)
  call CURSOR(0)
  call CLEAR

c
c Draw frame
c
  call MFRAME (15,1,0,1,1,24,79)
c
c Place label
  string = 'HELLO WORLD'
  call TLABEL(15,0,0,0,10,20,string)
c
  string = 'TYPE q TO QUIT!'
  call TLABEL(15,0,0,0,11,20,string)
c
c Halt
  call HALTCCLR(.TRUE.,1,'q')

c Restore cursor and original text colors
  call CURSOR(1)
  call ENDTEXT
c
  STOP
  END
```

This example produces the output shown in Fig. 4.



```
HELLO WORLD  
TYPE q to QUIT!
```

Fig. 4. Screen output for Example 4.

The function routine MENU is used to construct a text menu. The function call to MENU is as follows:

```
entry = MENU (tcol, bcol, hcol, hbcot, icen, ROW, COL,  
              uparrow, dntarrow, ltarrow, rtarrow, accept, quit,  
              mfirst, mcolumn, mtitle, mentry, string)
```

The MENU function returns *entry*, a number corresponding to the menu entry selected or 0 if the *quit* key is pressed.

The user supplies the number of titles and/or help lines for the menu, and the number of entries in the menu. The upper-left corner of the first menu item placed at the text position (*ROW, COL*). The text strings passed are in the array *string*.

The menu action must be passed through the MENU function call. The variables *uparrow*, *dntarrow*, *ltarrow*, and *rtarrow* are passed to the KEYINT routine. to specify the cursor up, cursor down, cursor left, and cursor right keys. The ASCII integer value for the <Enter> key, for example, is passed as *accept* and the ASCII integer value for the <Esc> key is passed as *quit*.

The user should call CLEAR or TCLEAR prior to executing the MENU function. If a menu frame is desired, the user must call MFRAME prior to executing the MENU function, placing the top-left and bottom-right corners of the menu frame just outside the menu entries. See the FASTPLOT LIBRARY REFERENCE for further details of the MENU function.

Example 5:

This example uses the FASTPLOT text function MENU to create a simple three-item selection menu. Upon selecting one of the first two items, the program confirms the selection and returns to the menu. The user exits the program by selecting the third item (Quit) or pressing the <Esc> key. The background screen color is set to black (*iback* = 0).

```
C Simple example of FASTPLOT MENU routine
  EXTERNAL MENU

c Define seven strings: four title lines and three entries.
  character*200 line1, line2
  character*80  string(7)
  character*1   key(1)
  integer*4    ROW(7), COL(7), mentry, mtitle, mfirst
  integer*4    tcol(7), tbc(7), hcol(7), hbcol(7), icen(7)
  integer*4    entry

c Return key is used to return to menu
  key(1) = CHAR(13)
  line2 = ' Press <Enter> to return to menu.'

c Set colors for title lines
  do 10 istr = 1, 4

c Foreground text colors
  tcol(istr) = 15
  tbc(istr) = 0

c Highlight text colors not needed for title lines
  hcol(istr) = 0
  hbcol(istr) = 0

c Center strings
  COL(istr) = 0
  icen(istr) = 1
10  continue

c Set colors for menu entries
  do 20 i = 5, 7

c Foreground text colors
  tcol(i) = 15
  tbc(i) = 0

c Highlight text colors must be defined for menu entries
  hcol(i) = 0
  hbcol(i) = 15
```

```

c Center strings - single column menu
    icen(i) = 1
    COL(i) = 0
20    continue

c Set ROW and titles
    ROW(1) = 5
    string(1) = 'Simple Example of FASTPLOT MENU Routine'
    ROW(2) = 6
    string(2) = '4/27/93'
    ROW(3) = 21
    string(3) = 'Use Arrow Keys To Make Selection'

c Reverse video on this line
    tcol(4) = 0
    tbc(4) = 15
    ROW(4) = 22
    string(4) = 'Press <Enter> To Execute, <Esc> To Exit'

c
c Set ROW and menu selection items
    ROW(5) = 10
    string(5) = 'First Selection'
    ROW(6) = 11
    string(6) = 'Second Selection'
    ROW(7) = 12
    string(7) = 'Quit'

c other menu parameters: # of title lines, # of entries, first entry
    mtitle = 4
    mentry = 3
    mfirst = 1

c
30    continue

c Background color (iback) is set to black (0).
    iback = 0
    call BGNTEXT(iback)
    call CURSOR(0)

c Double line frame
    call MFRAME( 14, 2, 0, 9, 30, 13, 49)

c Put up menu
    entry = MENU(tcol, tbc, hcol, hbcol, icen, ROW, COL,
-       328,336,333,331,13,27, mfirst, 0, mtitle, mentry, string)

c
c If quit key <ESC> is pressed, stop
    IF (entry.eq.0) then

c Restore cursor and original text colors
    call CURSOR (1)

```

```

        call ENDTEXT
        STOP
    ENDIF

    SELECT CASE (entry)

    CASE (1)
        call CLEAR
        line1 = string(5)
        call TLABEL(15,0,0,1,20,0,line1)
        call TLABEL(15,0,0,1,21,0,line2)
        call HALTCCLR(.TRUE.,1,key)

    CASE (2)
        call CLEAR
        line1 = string(6)
        call TLABEL(15,0,0,1,20,0,line1)
        call TLABEL(15,0,0,1,21,0,line2)
        call HALTCCLR(.TRUE.,1,key)

    CASE (3)
    c If quit item selected, stop
        call CLEAR
    c Restore cursor and original text colors
        call CURSOR (1)
        call ENDTEXT
        STOP

    END SELECT
    mfirst = entry

c loop back
    GOTO 30
END

```

This example produces the output shown in Fig. 5.

Simple Example of FASTPLOT Menu Routine
4/27/93

First Selection
Second Selection
Quit

Use Arrow Keys To Make Selection
Press <Enter> To Execute, <Esc> To Exit

Fig. 5. Screen output for Example 5.

The routine QUERY is called to query the user. The routine passes the query as *message*, a default answer as *answer*. The *message* string is placed at position (*ROW*,*COL*), and the *answer* string is placed one-character width from the end of the *message* string. The user presses <Enter> to accept the default answer or types in another answer and presses <Enter>. The user's answer can be compared to a set of acceptable (*good*) answers. If the user's answer does not correspond to any of the acceptable answers, an error message *errmsg* is printed at position (*ROWE*,*COLE*). The error message is centered in row (*ROWE*) if *icen* is 1.

If the query pertains to a filename, the value of (*ngood*) must be negative (-1 through -4). In either case *answer* is concatenated with the first element of *good* (which should be either blank or an extension) (e.g., '.DAT', to produce a filename). Depending on the value of *ngood*, various levels of file checking occur. See the FASTPLOT LIBRARY REFERENCE for further details on the QUERY routine. The call to QUERY is as follows:

```
CALL QUERY (mcol,mbcol,ROW,COL,message,answer,mbeep,ecol,ebcol  
icen,ROWE,COLE,errmsg,ebeep,ngood,good)
```

If text routines have been used, the user should call ENDTEXT at the end of the program. ENDTEXT restores the original foreground and background colors saved using BGNTEXT. Subroutine CURSOR must be called prior to ENDTEXT, to restore the cursor. The cursor type *itype* can be 1 - underline, 2 - double underline, or 3 - block.

Thus, to restore the *cursor* and the original foreground and background colors, use the following commands:

```
CALL CURSOR (I)  
CALL ENDTEXT
```

Examine the FASTPLOT demo menu driver (Sect. 3) for use of these FASTPLOT text routines.

Example 6:

This example uses the FASTPLOT text routine QUERY that asks the user to select among four items. After selecting one of the four items, the program confirms the selection and quits. The background screen color is set to black (*iback* = 0).

```
C Simple example of FASTPLOT QUERY routine
  character*80 mess,errmsg, string
  character*40 answer, accept(4)
  CHARACTER*8  item(4), time(4)
  CHARACTER*1  KEY(1)
  LOGICAL      mbeep,ebeep

  DATA item/'Bicycle ','Car      ','Train  ','Airplane'/
  DATA time/'3 months','6 days ','3 days  ','4 hours '/

  DATA accept/'1','2','3','4'/
  DATA nitem/4/
C RESPOND TO PRESSING CARRIAGE RETURN
  KEY(1)=CHAR(13)

C BEEP ON PAUSE
  mbeep = .TRUE.
  ebeep = .TRUE.

c Background color (iback) set to black (0).
  iback = 0

c Initialize text mode
  call BGNTXT(iback)
  call CURSOR(0)

c Clear the screen
  call CLEAR

c Query user
  string = 'FASTPLOT - QUERY EXAMPLE'
  call TLABEL(14,0,0,1,4,0,string)
  string = 'Select mode of transportation: '
  call TLABEL(14,0,0,1,6,0,string)

c
  do 1 i = 1, nitem
    string = char(i+48)//' - '//item(i)
    call TLABEL(12,0,0,0,i+7,35,string)
  1 continue

c
  mess = 'Enter mode (1-4): '
  answer = '1'
```

```

errmess = 'Incorrect response. Enter (1-4).'
```

c

```
itype = 1
```

c

```
call QUERY(15,iback,16,31, mess, answer, mbeep,14,iback,1,20,0,
#   errmess, ebeep, itype, nitem, accept)
```

c

```
it = ichar(answer)-48
```

c

```
call CLEAR
```

c

c Confirm user selection

```
string = 'Selected mode is: '//item(it)
call TLABEL(14,0,0,1,10,0,string)
```

```
string = 'Travel time is: '//time(it)
call TLABEL(14,0,0,1,12,0,string)
```

```
string = 'Press <Enter> to quit.'
call TLABEL(14,0,0,1,14,0,string)
```

c Halt

```
call HALTCLR(.TRUE.,1,key)
```

c Restore cursor and original text colors

```
call CURSOR(1)
call ENDTEXT
```

```
STOP
END
```

This example produces the output shown in Fig. 6.

FASTPLOT - QUERY EXAMPLE

Select mode of transportation

- 1 - Bicycle
- 2 - Car
- 3 - Train
- 4 - Airplane

Enter mode (1-4): 1

Selected mode is: Bicycle

Travel time is: 3 months

Press <Enter> to quit.

Fig. 6. Screen output for Example 6.

Additional routines within the FASTPLOT library are described below. Any of these routines can be called by the user's FORTRAN code as well.

Function BSPLIN returns the B-spline fit given four consecutive values along the curve (A, B, C, D). The parameter T is the sub-interval. This routine is called by FASTPLOT routine CURVE. The function call to BSPLIN is as follows:

F = BSPLIN (T, A, B, C, D)

Subroutine CAPITAL converts a lower case character, ch , to upper case. The call to CAPITAL is as follows:

CALL CAPITAL (ch)

The assembler routine KEYINT returns the ASCII value, $ikey$, for the key pressed. If the key returns an extended key code, KEYINT returns the extended key code + 256. The extended key codes and corresponding FASTPLOT codes are shown in Appendix 2. This routine is called by subroutines ENDPLOT and HALTCLR to test all keys which the user has selected to return the screen to default mode. The call to KEYINT is as follows:

CALL KEYINT (ikey)

The following routines have been added to FASTPLOT to do imaging with 256 colors in medium resolution graphics (320x200) on a VGA monitor.

**SUBROUTINE BGNPLOTM (icolor)
SUBROUTINE COLORMAP (indx, ired, igreen, iblue)
SUBROUTINE DEFINEMAP (itype)
SUBROUTINE PIXEL (icolor, ix, iy)
SUBROUTINE GREYSCALE (itype)
INTEGER*4 FUNCTION RGB (r, g, b)
SUBROUTINE RESET**

The routine BGNPLOTM is used to set the video mode to medium resolution graphics. The background color is set to $icolor$. Either GREYSCALE or DEFINEMAP is used to specify the color map. Setting the parameter $itype$ to 0 gives 16 levels of grey or color. Setting $itype$ to 1 gives 256 levels of grey or color. The 16 color map file is FP_COLOR.16 and the 256 color map file is FP_COLOR.256. The routine DEFINEMAP looks first in the local directory for these files, then in the directory specified in the file FP_FONT.LOC. The routine COLORMAP and the INTEGER FUNCTION RGB are called by GREYSCALE and DEFINEMAP to establish the color map. COLORMAP assigns the value $indx$ to the intensities $ired$, $igreen$, $iblue$. The function RGB mixes the red (r), green (g), and blue (b) components and returns the RGB value. The routine PIXEL is used to place a pixel of color $icolor$ at SCREEN COORDINATE (ix, iy). Subroutine RESET resets the screen mode to the default (text) mode, without halting for user response. Detailed descriptions of each of the FASTPLOT color imaging routines can be found in the FASTPLOT Library Reference. Appendix E contains a typical 16 color map (FP_color.16) and a typical 256 color map

(FP_color.256). Finally, two examples using these color imaging routines can be found in the FPDEMO program described in the following section.

3. FPDEMO - ADDITIONAL FASTPLOT EXAMPLES

Eight additional example programs using the FASTPLOT library are provided as subroutines in the program FPDEMO.FOR. The main program is a menu driver developed using the FASTPLOT function MENU.

In this section, we describe each of the examples. Three of the examples (10, 11, and 12) require input data files. Examples 11 and 12 require a LABEL file with extension LBL and a DATA file with extension DAT. Example 10 has a single file (EXAMPL10.DAT) which contains both label information and data. Screen dumps to an HP LaserJet + printer of the resulting plots are shown in following each example. The screen dumps are made using a locally written screen dump utility, PRTGRAPH.COM, developed by William Jackson of Computing Applications Division, Oak Ridge National Laboratory. The screen dumps may also be made using GRAPHICS.COM supplied with MS-DOS 5.0 or 6.0 or captured into Wordperfect® using GRAB.COM.

The FPDEMO main program displays a menu from which the user selects the example plot.

```
C*****
C FASTPLOT DEMO                                     12-30-93*
C Developed by Richard C. Ward, Computing Applications Division, ORNL *
C P.O. Box 2008, Oak Ridge National Laboratory, Oak Ridge, TN      *
C 37831-6243 Phone: (615) 574-4559 E-mail: rwd@ornl.gov           *
C*****
C
C*****
C FASTPLOT MENU DRIVER                                     *
C DEMONSTRATES FASTPLOT MENU CAPABILITIES                 *
C*****
      EXTERNAL MENU
      character*80  string(20)
      integer*4    ROW(20), COL(20)
      integer*4    tcol(20), tbc col(20), hcol(20), hbcol(20), icen(20)
c
      integer*4    entry
      integer*4    uparrow, d narrow, ltarrow, rtarrow, accept, quit
      integer*4    mfirst, mborder, mtitle, mentry, mcolumn
c define menu

      do 10 istr = 1, 13
         tbc col(istr) = 0
         hcol(istr) = 0
         hbcol(istr) = 0
         COL(istr) = 0
c to center string, set icen(istr) = 1, COL(istr) is then ignored
         icen(istr) = 1
      10  continue

c specify number of title and help lines
```

```

        mtitle = 4

c specify number of menu entries
        mentry = 9

c maximum number of characters in menu entry
        mchars = 17

c menu box frame and menu box entries
        mborder = 3

c number of entries in each column for multicolumn menus
c or zero for single column menus
        mcolumn = 0

        tcol(1) = 15
        ROW(1) = 5
        string(1) = 'FASTPLOT Plotting Package Examples'

        tcol(2) = 14
        ROW(2) = 6
        string(2) = 'Ver. 1.0 Date: 12/30/93'

        tcol(3) = 14
        ROW(3) = 21
        string(3) = 'Use Arrow Keys To Make Selection'

        tcol(4) = 0
        tbc(4) = 15
        ROW(4) = 22
        string(4) = 'Press <Enter> To Execute, <Esc> To Exit'
c
        do 20 i = 5, 13
c set foreground text colors
        tcol(i) = 12
        tbc(i) = 0
c set highlight text colors
        hcol(i) = 0
        hbcol(i) = 15
c do not center string
        icen(i) = 0
c set ROW
        ROW(i) = i+4
c set column - single column
        COL(i) = 31
20    continue

        string(5) = 'Fonts Available'
        string(6) = 'Symbols Available'
        string(7) = 'Symbol Plot'
        string(8) = 'Histogram Plot'

```

```

        string(9) = 'Curve Plot'
        string(10) = 'Error Bars'
        string(11) = 'Color Image'
        string(12) = 'Fractal Image'
        string(13) = 'FASTPLOT HELP'
c
c MENU actions - add 256 to scan code to get extended key codes
        uparrow = 72+256
        dntarrow = 80+256
        rtarrow = 77+256
        ltarrow = 75+256
c accept = <Enter>, quit = <Esc>
        accept = 13
        quit = 27

        mfirst = 1
30 continue

        call BGNTXT(0)
c hide cursor
        call CURSOR(0)
c put up centered menu frame
        istr = 5
        call MFRAME( 14, 2, 1,
+ ROW(istr)-1, COL(istr) - mborder -1,
+ ROW(istr+mentry-1)+1, COL(istr+mentry-1)+mborder+mchars)
c put up menu
        entry = MENU(tcol, tbcot, hcol, hbcot, icen,
+ ROW, COL,
+ uparrow, dntarrow, ltarrow, rtarrow, accept, quit,
+ mfirst, mcolumn, mtitle, mentry, string)
c
c call TCLEAR
call CLEAR
IF (entry.eq.0) then
c reveal cursor
        CALL cursor (2)
        call ENDTEXT
        STOP
ENDIF

SELECT CASE (entry)
CASE (1)
        CALL EXAMPL7
CASE (2)
        CALL EXAMPL8
CASE (3)
        CALL EXAMPL9
CASE (4)
        CALL EXAMPL10
CASE (5)

```

```
CALL EXAMPL11
CASE (6)
CALL EXAMPL12
CASE (7)
CALL IMAGE
CASE (8)
CALL FRACTAL
CASE (9)
CALL HELP
END SELECT
mfirst = entry
GOTO 30
END
```

The FPDEMO main program displays the selection menu shown in Fig. 7.

FASTPLOT Plotting Package Examples
Ver. 1.0 Date: 12/30/93

Fonts Available
Symbols Available
Symbol Plot
Histogram Plot
Curve Plot
Error Bars
Color Image
Fractal Image
FASTPLOT HELP

Use Arrow Keys To Make Selection
Press <Enter> To Execute, <Esc> To Exit

Fig. 7. FPDEMO selection menu.

Example 7:

The seventh FASTPLOT example displays the Microsoft® FORTRAN graphics fonts available to the user using the FASTPLOT library. The FASTPLOT routine QUERY is used to ask the user for the typeface to display. There are six possible typefaces: Courier, Helv, Tmns Roman, Modern, Script and Roman. For the selected typeface, the subroutine displays a line of text with heights of: 5, 10, 15, 20, 25, 30, 35 and 40 pixels with each line in a different color. The width is taken as 0.67 times the height. For the bit-mapped typefaces (Courier, Helv, and Tmns Roman) the best fit is determined from the specified font height and width (integers). Since all of the bit-mapped typefaces are restricted in pixel height to 15 (courier), 28 (Helv) and 26 (Tmns Roman), the size of the letters remains fixed when the specified height exceeds these limits. On the other hand, the scalable typefaces (Modern, Script and Roman) can be adjusted to the exact height and width specified.

The seventh FASTPLOT example uses subroutine QUERY to query the user as to the typeface desired. It then calls BGNPLOT to put the display in graphics mode, REG_FONTS to register the fonts, SCREEN, PORT and WINDOW to set up the screen coordinates and graphics window. The FASTPLOT routine SET_FONT is called with the selected typeface and with varying font height and width. The LABEL routine is used to write a string in the selected typeface using the specified height and width. Finally, ENDPLOT is called to return the display to default (text) mode.

```

SUBROUTINE EXAMPL7
C*****
C FASTPLOT EXAMPLE 7 - DISPLAY FONTS *
C*****
CHARACTER*200 STRING, TITLE
character*80 mess,errmsg, fnt
character*40 answer, accept(6)
CHARACTER*18 LOGO
CHARACTER*10 FONT(6)
CHARACTER*1 KEY(2)
INTEGER*4 IFONT, IFONTH, IFONTW
LOGICAL mbeep,ebeep, BEEP

DATA FONT/'Courier ', 'Helv ',
% 'Tms Roman', 'Modern ',
% 'Script ', 'Roman '/

C RESPOND TO PRESSING ESCAPE KEY OR CARRIAGE RETURN
KEY(1)=CHAR(13)
KEY(2)=CHAR(27)
C BEEP ON PAUSE
BEEP = .TRUE.
mbeep = .TRUE.
ebeep = .TRUE.
LOGO = 'FASTPLOT Ver 1.0 '
c query user
fnt = 'FASTPLOT EXAMPLE 7 - FONTS'
call TLABEL(14,0,0,1,2,0,fnt)
fnt = 'Select from these typefaces: '
call TLABEL(14,0,0,1,4,0,fnt)
do 1 i=1,6
fnt = char(i+48)//' - '//font(i)
call TLABEL(12,0,0,0,i+4,35,fnt)
1 continue
mess = 'Enter typeface: '
answer = '1'
errmsg = 'Incorrect response. Enter (1-6).'
itype = 1
Naccept = 6
accept(1) = '1'
accept(2) = '2'
accept(3) = '3'
accept(4) = '4'
accept(5) = '5'
accept(6) = '6'
call QUERY(15,0,15,31,mess,answer,mbeep,14,0,1,20,0,
# errmsg,ebeep,itype,Naccept,accept)
ifont = ichar(answer)-48
TITLE = 'FONT: '//font(ifont)

C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK

```

```

        CALL BGNPLOT (0)
C REGISTER FONTS
        CALL REG_FONTS
C OBTAIN SIZE OF SCREEN IN PIXELS
        CALL SCREEN(IPIX_X,IPIX_Y)
C SET VIEWPORT AND GRAPHICS WINDOW TO OUTSIDE EDGES OF SCREEN
        CALL WINDOW(0.,1000.,1000.,0.)
        CALL PORT(0,0,IPIX_X,IPIX_Y)

C SET FONT FOR TITLE TO HELV 22x12
        CALL SET_FONT(2,22,12)
C PUT ON TITLE
        CALL LABEL(15, 0, 0, 1, 0, 10, TITLE)

C LOOP OVER FONT HEIGHT
        DO 200 J = 1, 8
C SET FONT HEIGHT TO (5,10,15,20,25,30,35,40)
        IFONTH = J * 5
C SET FONT WIDTH TO 0.67 * FONT HEIGHT
        IFONTW = FLOAT(IFONTH) * 0.67
C SET FONT
        CALL SET_FONT (IFONT, IFONTH, IFONTW)

C SET UP LABEL STRING
        STRING = LOGO//FONT(IFONT)
C SPECIFY COLOR OF LABEL STRING
        ICOLOR = J + 7
        ICENTER = 0
        IXPOS = 10
        IYPOS = 40 + (J-1)*38
C WRITE STRING IN FONT SELECTED
        CALL LABEL(ICOLOR, 0, 0, ICENTER, IXPOS, IYPOS, STRING)
    200 CONTINUE
C END PLOT
        CALL ENDPLOT (BEEP, 2, KEY)
        RETURN
        END

```

The selection menu for Example 7 is shown in Fig. 8. An example font (script) is shown in Fig. 9.

FASTPLOT EXAMPLE 7 - FONTS

Select from these typefaces:

- 1 - Courier
- 2 - Helv
- 3 - Tmns Roman
- 4 - Modern
- 5 - Script
- 6 - Roman

Enter typeface: 5

Fig. 8. Selection menu for Example 7.

FONT: Script

FASTPLOT Ver 1.0 Script

Fig. 9. Screen output when Script font is selected from Example 7 menu.

Example 8:

The eighth FASTPLOT example displays the symbols available with the FASTPLOT library. The example shows each of the 12 symbols using different colors and with increasing size. The example pauses, and after the user presses either the escape, <Esc>, key or the carriage return, <CR>, key, the example displays the same symbols in reverse order. The symbols, like the scalable fonts, can be scaled as large as the user wishes. This example also lists the symbols available in FASTPLOT.

The eighth FASTPLOT example calls BGNPLOT to put the display in graphics mode, REG_FONTS to register the fonts, SCREEN, PORT and WINDOW to set up the screen coordinates and graphics window. It then calls SET_SYMBOL to load the symbol table (FP_SYMBL.TAB) into memory. The LABEL routine is called to put a title on the top of the plot. It then loops over the symbol ID, from 0 to 13, calling MARKER to select the symbol and SYMBOL to plot the symbol with selected color and size. The routine HALTCLR is then called to pause, requiring user response by typing the specified keys (<Esc> or <CR> in this case). When the user responds, the example, redraws the symbols in reverse order.

This example also lists the symbols available in FASTPLOT. Within FASTPLOT, the positioning of symbols is done using graphics window coordinates whereas the positioning of labels is done using screen coordinates. Note, also, that the graphics window coordinates increase from bottom to top, whereas the graphics window coordinates increase from top to bottom. This procedure can be somewhat confusing. In this part of the example, the graphics window coordinates are set to the screen coordinates to reduce this confusion. For each display type (EGA or VGA) allowed, the position of the symbols and labels is computed so that the resulting plot is identical. Each symbol and its corresponding descriptive text is done in a different color.

Finally, ENDPLOT is called to return the display to default (text) mode.

```

SUBROUTINE EXAMPL8
C*****
C FASTPLOT EXAMPLE 8 - FASTPLOT SYMBOLS *
C*****
      REAL*4      X(1), Y(1), SYMH, SPACING, RATIO
      REAL*4      SCR_N_X, SCR_N_Y
      CHARACTER*1  KEY(2)
      CHARACTER*3  DISPLAY
      CHARACTER*200 TITLE, STRNG
      CHARACTER*200 STRING(0:13)
      LOGICAL      BEEP

      DATA STRING/'OPEN SQUARE','OPEN CIRCLE','OPEN TRIANGLE',
+      'VERTICAL CROSS(+)','LAZY CROSS(x)','OPEN DIAMOND',
+      'OPEN INVERTED TRIANGLE','FILLED SQUARE',
+      'FILLED CIRCLE','FILLED TRIANGLE','FILLED DIAMOND',
+      'FILLED INVERTED TRIANGLE',
+      'HORIZONTAL BAR - USED WITH ERROR BARS',
+      'VERTICAL BAR - USED WITH ERROR BARS'/

C ESTABLISH BREAK-OUT KEYS
      KEY(1)=CHAR(13)
      KEY(2)=CHAR(27)
C BEEP ON PAUSE
      BEEP=.TRUE.
C
      TITLE='FASTPLOT SYMBOLS'
C
C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK
      CALL BGNPLOT (0)

      CALL GET_DISPLAY ( DISPLAY )
      IF (DISPLAY.EQ.'CGA'.OR.DISPLAY.EQ.'TXT')
+      STOP ' Correct graphics mode (EGA or VGA) not available'

C REGISTER FONTS - LOAD FONTS INTO MEMORY
      CALL REG_FONTS

C SELECT FONT
      CALL SET_FONT(2,22,12)

C LOAD SYMBOL TABLE
      CALL SET_SYMBOL

C GET SCREEN PIXEL WIDTH AND HEIGHT
      CALL SCREEN (IX, IY)

C SET PORT - SCREEN COORDINATES
C IF EGA: VIEWPORT IS 640 X 350 INCREASING FROM TOP TO BOTTOM
C IF VGA: VIEWPORT IS 640 X 480 INCREASING FROM TOP TO BOTTOM

```

```

CALL PORT (0, 0, IX, IY)

C SET WINDOW - VIRTUAL COORDINATES
CALL WINDOW (0., 1000., 1000., 0.)

C WRITE TITLE
CALL LABEL (15, 0, 0, 0, 20, 10, TITLE)
CALL LABEL (0, 15, 1, 0, 380, 440, TITLE)

C LOOP OVER SYMBOLS IN SYMBOL TABLE (FP_SYMBL.TAB)
DO 100 ID = 0, 13
C ID - SYMBOL ID
C SYMH - SYMBOL HEIGHT
J = ID + 1
SYMH = J * 4

C SET MARKER
CALL MARKER (ID, SYMH)

C INCREASE SIZE OF SYMBOLS PROGRESSIVELY
X(1) = (J*J + J)*4 + 50
Y(1) = (J*J + J)*4 + 40
NPOINTS = 1

C SELECT COLOR OF SYMBOL
ICOLOR = J + 1

C PUT SYMBOL ON PLOT
CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)
100 CONTINUE

C HALT, WAIT FOR USER TO PRESS SPECIFIED KEYS THE CLEAR SCREEN
CALL HALTCLR(BEEP,2,KEY)

C LOOP OVER SYMBOLS IN SYMBOL TABLE (FP_SYMBL.TAB)
C WRITE TITLE - BLACK ON WHITE BACKGROUND
C CALL LABEL(ICOL, IBACKCOL, IBACK, ICENTER, XLOC, YLOC, STRING)
CALL LABEL (0, 15, 1, 0, 20, 10, TITLE)
CALL LABEL (15, 0, 0, 0, 380, 440, TITLE)

DO 200 ID = 0, 13
C ID - SYMBOL ID
C SYMH - SYMBOL HEIGHT
J = 14 - ID
SYMH = J * 4

C SET MARKER
CALL MARKER (ID, SYMH)

C INCREASE SIZE OF SYMBOLS PROGRESSIVELY
X(1) = (J*J + J)*4 + 50
Y(1) = (J*J + J)*4 + 40
NPOINTS = 1

C SELECT COLOR OF SYMBOL
ICOLOR = J + 1

```

```

C PUT SYMBOL ON PLOT
      CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)
200   CONTINUE

C HALT, WAIT FOR USER TO PRESS SPECIFIED KEYS THE CLEAR SCREEN
      CALL HALTCLR(BEEP,2,KEY)

      RATIO = 1.0
      IF (DISPLAY.EQ.'VGA') RATIO = 480./350.

C SET PORT - SCREEN COORDINATES
C IF EGA: VIEWPORT IS 640 X 350 INCREASING FROM TOP TO BOTTOM
C IF VGA: VIEWPORT IS 640 X 480 INCREASING FROM TOP TO BOTTOM
      CALL PORT (0, 0, IX, IY)

      SCRN_X = IX
      SCRN_Y = IY
C SET WINDOW - VIRTUAL COORDINATES: UPPER LEFT, LOWER RIGHT
      CALL WINDOW (0., SCRN_Y, SCRN_X, 0.)

      TITLE='FASTPLOT SYMBOL TABLE'
C
C SELECT FONT
      IFONT = 3
      IFONTH = 15
      IFONTW = 8

      CALL SET_FONT(IFONT, IFONTH, IFONTW)

C WRITE TITLE
      CALL LABEL (15, 0, 0, 1, 0, 5, TITLE)

C LOOP OVER SYMBOLS IN SYMBOL TABLE (FP_SYMBL.TAB)
      DO 300 ID = 13, 0, -1
C ID - SYMBOL ID
      J = ID + 1
C SYMH - SYMBOL HEIGHT (IN PIXELS):
      SYMH = 12.0
C CALL MARKER
      CALL MARKER (ID, SYMH)

C POSITION OF SYMBOL AT X(1), Y(1)
      X(1) = 100

C SPACING BETWEEN LINES (IN PIXELS):
      SPACING = RATIO * 22.
      Y(1) = ( SPACING * J )

C COLOR OF SYMBOL AND TEXT
      ICOLOR = J + 1

```

```

C POSITION OF SYMBOL ID
      JX = X(1) - 60
C INVERT Y POSITION FOR PLACING LABELS
      JY = IY - Y(1) - RATIO * FLOAT(IFONTH)/2.
      ID10 = ID/10
      STRNG(1:1) = ' '
      IF ( ID10.NE.0 ) STRNG(1:1) = CHAR( ID10 + 48)
      STRNG(2:2) = CHAR( ID - (ID10)*10 + 48)
C PLACE SYMBOL ID ON PLOT
      CALL LABEL (ICOLOR, 0, 0, 0, JX, JY, STRNG)
C
C SET LINETYPE TO SOLID
      CALL LINETYPE(0)
C PLACE SYMBOL ON PLOT
      NPOINTS = 1
      CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)

C POSITION OF TEXT
      JX = X(1) + 60
C INVERT Y POSITION FOR PLACING LABELS
      JY = IY - Y(1) - RATIO * FLOAT(IFONTH)/2.
C PLACE TEXT ON PLOT
      CALL LABEL (ICOLOR, 0, 0, 0, JX, JY, STRING(ID))
300      CONTINUE

C UNLOAD FONTS
      CALL UNREG_FONTS

C END PLOT - RETURN TO DEFAULT MODE
      CALL ENDPLOT(BEEP,2,KEY)
C
      RETURN
      END

```

The screen output for Example 8 is shown in Figs. 10 through 12.

FASTPLOT SYMBOLS

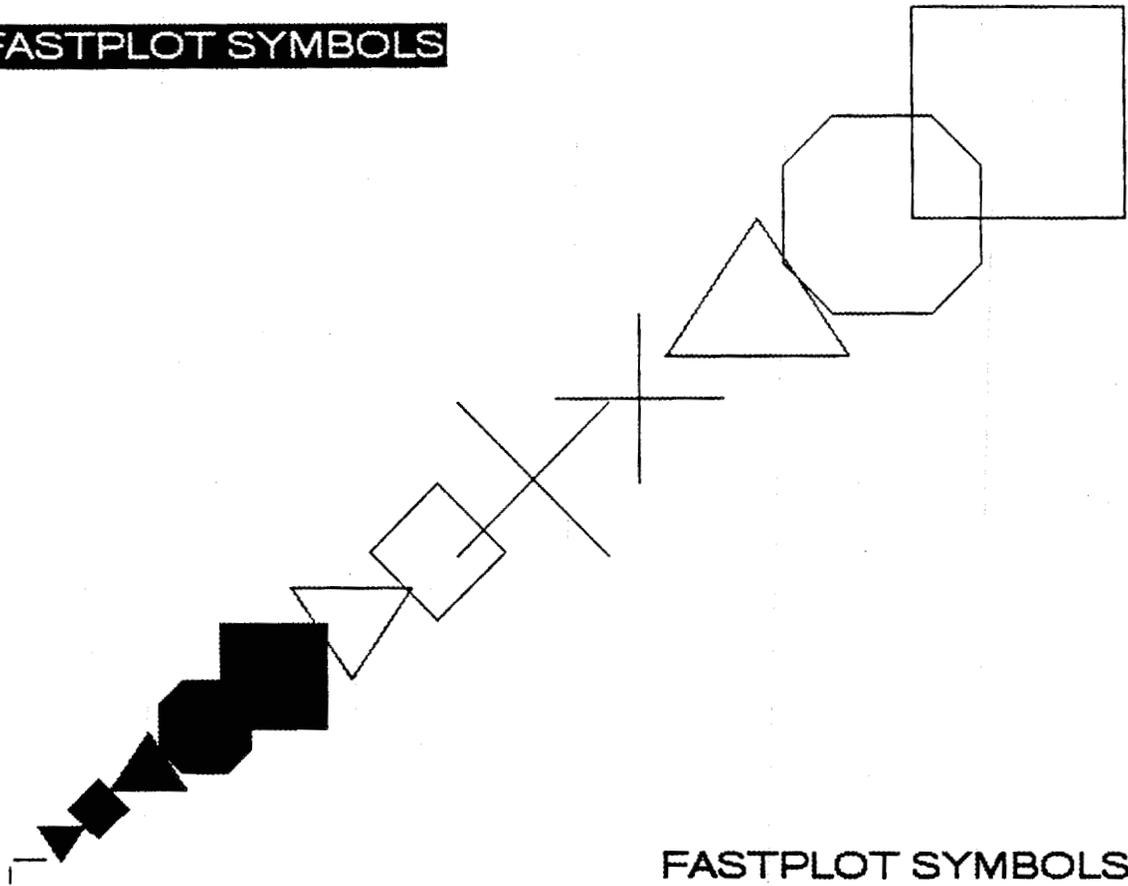


Fig. 11. Screen output for Example 8.

FASTPLOT SYMBOL TABLE

13		VERTICAL BAR - USED WITH ERROR BARS
12	—	HORIZONTAL BAR - USED WITH ERROR BARS
11	▼	FILLED INVERTED TRIANGLE
10	◆	FILLED DIAMOND
9	▲	FILLED TRIANGLE
8	●	FILLED CIRCLE
7	■	FILLED SQUARE
6	▽	OPEN INVERTED TRIANGLE
5	◇	OPEN DIAMOND
4	×	LAZY CROSS(x)
3	+	VERTICAL CROSS(+)
2	△	OPEN TRIANGLE
1	○	OPEN CIRCLE
0	□	OPEN SQUARE

Fig. 12. Screen output for Example 8.

Example 9:

The ninth FASTPLOT example demonstrates using just symbols alone on a plot. In this case the data are generated internally in the example code and three different colored symbols are used to display the data.

Calls to PORT and WINDOW establish the graphics window. A frame is drawn around the window, by calling FRAME. Subroutine GRID is called to place a grid over the graphics window. The tick height and width (roughly in pixels) is specified and the ticks drawn with calls to routine TICKS.

This example demonstrates how to set the tick marks, call the MARKER routine to select the symbol, and call SYMBOL to plot the symbol at every data point.

```

SUBROUTINE EXAMPL9
C*****
C FASTPLOT EXAMPLE 9 - SIMPLE SYMBOL PLOT *
C*****
      REAL*4      X(10), Y(10), SYMH
      CHARACTER*1 KEY(2)
      LOGICAL     BEEP
      CHARACTER*200 TITLE

C
      TITLE='FASTPLOT EXAMPLE 9 - SYMBOL PLOT'

C
C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK
      CALL BGNPLOT (0)
C REGISTER FONTS - LOAD FONTS INTO MEMORY
      CALL REG_FONTS
C SELECT FONT
      CALL SET_FONT(2,22,12)
C LOAD SYMBOL TABLE
      CALL SET_SYMBOL
C GET SCREEN PIXEL WIDTH AND HEIGHT
      CALL SCREEN (IX, IY)
C SET PORT - SCREEN COORDINATES
      CALL PORT (0, 0, IX, IY)
C SET WINDOW - VIRTUAL COORDINATES
      CALL WINDOW (0., 1000., 1000., 0.)
C WRITE TITLE
      CALL LABEL (15, 0, 0, 1, 0, 20, TITLE)

C SET INSIDE PORT - SCREEN COORDINATES
      CALL PORT (60, 60, IX-60, IY-60)
C SET INSIDE WINDOW - VIRTUAL COORDINATES
      CALL WINDOW (0., 11., 11., 0.)
C DRAW GRID, TICK MARKS, FRAME
      call GRID (4,1,1.,2)
      call GRID (5,2,1.,0)
      CALL TICKS (15, 10, 6, 8, 1.,0)
      CALL TICKS (15, 20, 6, 8, 1.,0)
      CALL TICKS (15, 30, 6, 8, 1.,0)
      CALL TICKS (15, 40, 6, 8, 1.,0)
      CALL FRAME(15)
      NPOINTS = 10
C SYMH - SYMBOL HEIGHT
      SYMH = 4.0

C ID - SYMBOL ID
      ID = 7
C SET MARKER
      CALL MARKER (ID, SYMH)
C CREATE PLOT VECTOR (X,Y)

```

```

        DO 100 I=1, NPOINTS
            X(I) = I
            Y(I) = X(I)
100      CONTINUE
            ICOLOR = 14
            CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)

C ID      - SYMBOL ID
            ID      = 8
C SET MARKER
            CALL MARKER (ID, SYMH)
C CREATE PLOT VECTOR (X,Y)
            DO 200 I=1, NPOINTS
                X(I) = I
                Y(I) = X(I)**0.5
200      CONTINUE
            ICOLOR = 13
            CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)

C ID      - SYMBOL ID
            ID      = 9
C SET MARKER
            CALL MARKER (ID, SYMH)
C CREATE PLOT VECTOR (X,Y)
            DO 300 I=1, NPOINTS
                X(I) = I
                Y(I) = X(I)**2.
300      CONTINUE
            ICOLOR = 12
            CALL SYMBOL (ICOLOR, 11, 1, NPOINTS, X, Y)

C UNLOAD FONTS
            CALL UNREG FONTS
C ESTABLISH BREAK-OUT KEYS
            KEY(1)=CHAR(13)
            KEY(2)=CHAR(27)
C BEEP ON PAUSE
            BEEP=.TRUE.

C END PLOT - RETURN TO DEFAULT MODE
            CALL ENDPLOT(BEEP,2,KEY)
C
            RETURN
            END

```

This example produces the screen output shown in Fig. 13.

FASTPLOT EXAMPLE 9 - SYMBOL PLOT

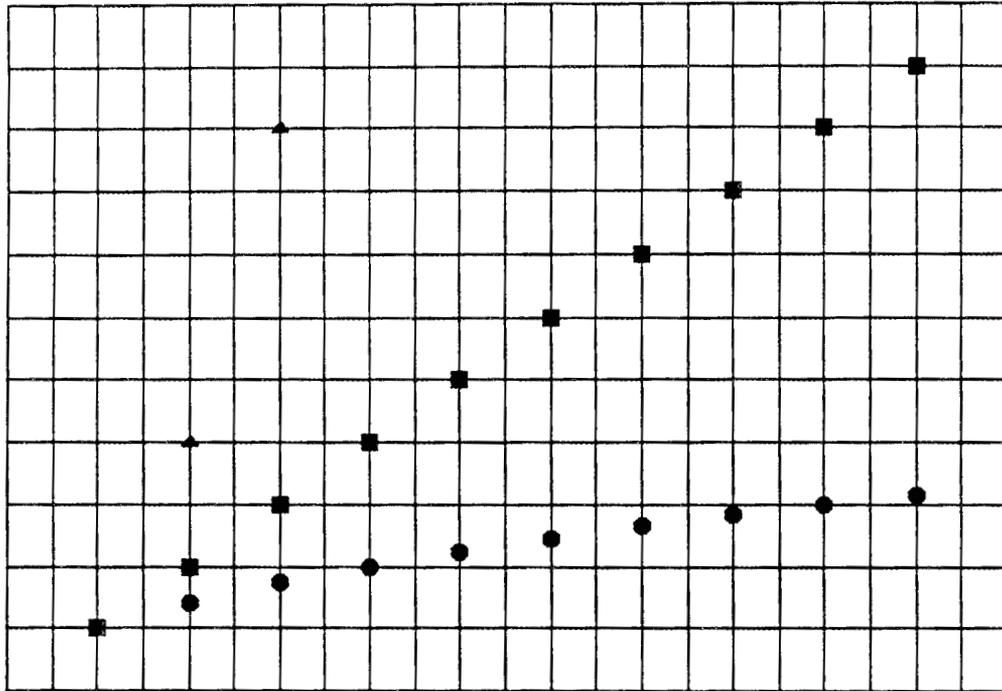


Fig. 13. Screen output for Example 9.

Example 10:

The tenth FASTPLOT example demonstrates a histogram plot. This example uses two input files. The input file, **EXAMPL10.LBL**, contains plot label information and allows the user to select either filled or unfilled histogram bars. The input file, **EXAMPL10.DAT**, contains the data. The program sorts the input data into bins.

Calls to **PORT** and **WINDOW** establish the inside graphics window. Another frame is drawn around the inside window, this time using calls to **AXIS** to draw each axis separately. The tick height and width (roughly in pixels) is specified and the ticks drawn with calls to routine **TICKS**.

The numbers on the vertical and horizontal axes are then drawn. This step requires shifting the plot graphics window defined previously by a specified amount to write the numbers below the bottom axis and to the left of the left-side axis. The user specifies the *format form* of the numbers (e.g., **F5.2** or **1PE7.2**). The length of the format specification is taken from the first number left of the decimal in the format. The user also specifies how often to place the numbers *step* and whether or not to skip *skip* placing the first number. The logical variable *skip* allows for cleaning looking plots in many cases. The **NUMBERS** routine uses the above information, the tick width and height passed from subroutine **TICKS**, and the font width and height passed from **SET_FONT** to shift the graphics window and place the numbers. It then internally resets the graphics window to the original window size for drawing the curves.

After calling the routine **NUMBERS** to put the numbers on the axes, example 2 calls **CURVE** to draw the histogram. If *fill* is **.TRUE.**, the histogram bars are filled in; if *fill* is **.FALSE.**, the histogram bars are not filled. The parameter *fill* can be modified in the **EXAMPL10.LBL** file.

```

SUBROUTINE EXAMPL10
C*****
C FASTPLOT EXAMPLE 10 - HISTOGRAM PLOT *
C*****
C
C STRINGS FOR HEADING, SUBHEADING, LABELS AND LEGENDS
C TOTAL OF 20 LABELS POSSIBLE
      CHARACTER*200 string(20)
      INTEGER*4 icol(20), istr(20), strx(20), stry(20)
      INTEGER*4 IBACKCOL(20), IBACK(20)
C KEY IS ARRAY WITH KEYS WHICH ARE USED FOR KEYBOARD INTERRUPT WHEN PLOT
C IS FINISHED
      CHARACTER*1 KEY(4)
      LOGICAL BEEP
C DISPLAY is GRAPHICS DISPLAY TYPE (CGA,EGA,VGA or TXT if no graphics)
      CHARACTER*3 DISPLAY

C FORM IS FORMAT PASSED TO NUMBERS ROUTINE
      CHARACTER*6 FORM

C PLOT ARRAY VARIABLE (X,Y)
      REAL*4 X(200), Y(200), XSTEP, YSTEP
      REAL*4 XMIN, XMAX, YMIN, YMAX, Xtick
C HISTOGRAM FILL PARAMETER
      LOGICAL FILL

C GET PLOT DATA
      OPEN (UNIT=10,FILE='EXAMPL10.DAT',STATUS='OLD')
      READ (UNIT=10,FMT=*)NUMSTR
      DO 11 I=1,NUMSTR
          READ (UNIT=10,FMT=*)ICOL(I), IBACKCOL(I), IBACK(I),
+           ISTR(I), STRX(I), STRY(I)
          READ (UNIT=10,FMT=1) STRING(I)
11      CONTINUE
1      FORMAT(A200)
      READ (UNIT=10,FMT=*) XMIN,XMAX,XSTEP, Xtick, YMIN,YMAX,YSTEP

      READ (UNIT=10,FMT=*) FILL
      READ (UNIT=10,FMT=*) NX
      DO 100 I = 1, NX
          X(I) = Xtick*I
      READ (UNIT=10,FMT=*) Y(I)
100     CONTINUE
      CLOSE (UNIT=10)

      PTYPE='H'

      CALL GET_DISPLAY(DISPLAY)
      IF(DISPLAY.EQ.'CGA'.OR.DISPLAY.EQ.'TXT')
+      STOP ' Correct graphics mode (EGA or VGA) not available'

```

```

C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK
    CALL BGNPLOT (0)
C REGISTER FONTS - LOAD FONTS INTO MEMORY
    CALL REG_FONTS
C CLEAR SCREEN
    CALL CLEAR

C GET SIZE OF SCREEN IN PIXELS
    CALL SCREEN (IPIX_X, IPIX_Y)

C SET GRAPHICS WINDOW (VIRTUAL COORDINATES) FOR FRAME
    CALL WINDOW (0., 1000., 1000., 0.)
C SET VIEWPORT (SCREEN COORDINATES) FOR FRAME
    CALL PORT (0, 0, ipix_x, ipix_y)
C DRAW FRAME
    CALL FRAME(15)

C LOAD SYMBOL TABLE INFORMATION - NOT LOADED NOW AS NOT NEEDED IN THIS
EXAMPLE
C     CALL SET_SYMBOL
C SET FONT:
    CALL SET_FONT(2,22,12)

C PUT LABELS (HEADING, SUB-HEADING, XLABEL, YLABEL) ON PLOT:
    DO 20 I = 1, 4
        CALL LABEL (ICOL(I), IBACKCOL(I), IBACK(I),
+             ISTR(I), STRX(I), STRY(I), STRING(I))
    20 CONTINUE

C SET UP PLOT VIEWPORT AND WINDOW
    CALL PORT (80, 60, ipix_x-60, ipix_y-60)
    CALL WINDOW (XMIN, YMAX, XMAX, YMIN)

C PLACE TICK MARKS
    CALL TICKS(15, 11, 8, 6, Xtick,0)
    CALL TICKS(15, 20, 8, 6, YSTEP,0)
    CALL TICKS(15, 30, 8, 6, Xtick,0)
    CALL TICKS(15, 40, 8, 6, YSTEP,0)
C DRAW INSIDE FRAME
    CALL FRAME(14)

C HORIZONTAL AXIS NUMBERS
    FORM = 'F3.1'
    CALL NUMBERS(15, 11, FORM, XSTEP, .FALSE.)

C VERTICAL AXIS NUMBERS
    FORM = 'F5.3'
    CALL NUMBERS(15, 21, FORM, YSTEP, .FALSE.)

C SET LINETYPE TO SOLID
    CALL LINETYPE(0)

```

```

C DRAW PLOT
c icolor is color of histogram
    icolor = 13
    npoints = NX
    CALL CURVE (icolor, 31, FILL, Xtick, npoints, X, Y)

    CALL UNREG_FONTS

C PASS KEYS FOR PLOT INTERRUPT THROUGH ENDPL - IF Q,q,<ESC> or <CR>
C   IS PRESSED, THE PLOT IS CLEARED AND USER IS RETURNED TO TEXT MODE.
    KEY(1)='Q'
    KEY(2)='q'
C ESC CHARACTER
    KEY(3)=CHAR(27)
C CARRIAGE RETURN CHARACTER
    KEY(4)=CHAR(13)
C BEEP ON PAUSE
    BEEP=.TRUE.
    CALL ENDPLOT(BEEP,4,KEY)

    RETURN
    END

```

EXAMPL10.DAT - Input data for Example 10.

```

4      Number of legends                HISTOGRAM EXAMPLE DATA FILE
14 0 0 1 0 10  Title
PATHLENGTH DISTRIBUTION
14 0 0 1 0 30  Sub-Title
NSORG = 3 NTORG = 3 NHIST =10000
14 0 0 1 0 454 (320 for EGA graphics)  X-axis
R (CM)
14 0 0 0 30 30  Y-axis
F (R)
0. 7.0 1.0 0.125 0. 0.04 0.005  xmin xmax xstep xtick ymin ymax
ystep
F  FILL HISTOGRAM BARS (T/F)
56      NUMBER OF BINS
0.001
0.003
0.005
0.009
0.011
0.014
0.019
0.0205
0.023
0.027
0.0265

```

0.028
0.0275
0.0255
0.029
0.0275
0.033
0.032
0.0322
0.0323
0.036
0.0375
0.0335
0.038
0.035
0.0345
0.0342
0.0340
0.0323
0.0355
0.03
0.029
0.027
0.0265
0.021
0.018
0.017
0.0155
0.0147
0.011
0.0095
0.008
0.005
0.0052
0.0045
0.004
0.0027
0.0025
0.0022
0.002
0.001
0.000
0.000
0.000
0.000

Example 10 produces the screen output shown in Fig. 14.

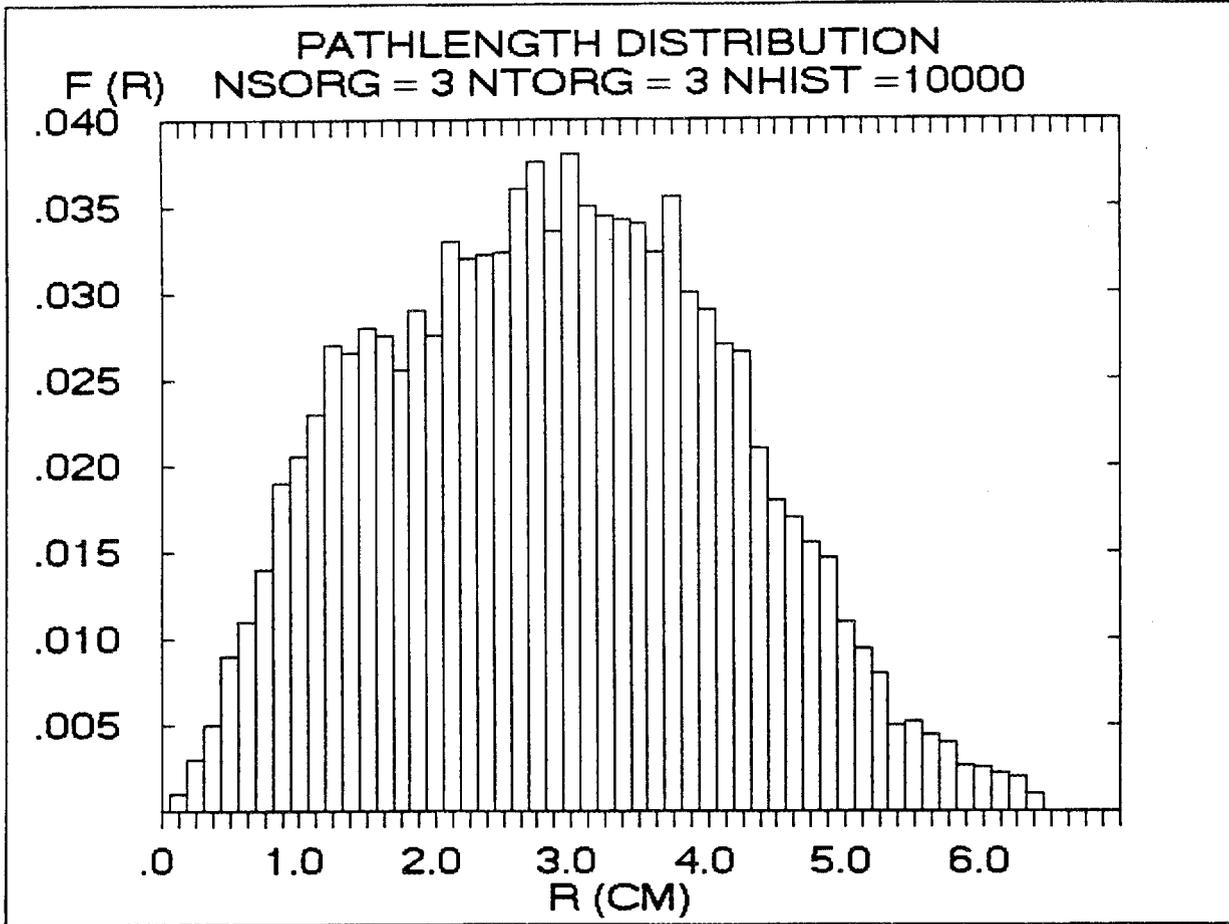


Fig. 14. Screen output for Example 10.

Example 11:

The eleventh FASTPLOT example is the most complex. It demonstrates a typical plot consisting of three curves plotted as a function of an X variable. The curves are mortality data, and they are plotted as a function of an individual's age at death (from 0 to 120 years). The mortality data (deaths from lung cancer for combined population, males and females) are contained in the file EXAMPL11.DAT. The labels for the plot are contained in the file EXAMPL11.LBL.

The user is asked, using the FASTPLOT routine QUERY, for the type of plot to be generated, and, depending on the plot type, whether symbols should be placed on the plot. Six types of curves can be drawn: linear curve ('C'), Bspline-fit linear curve ('B'), Y log - X linear curve ('Y'), X log - Y linear curve ('X'), Y log - X log ('L'), and histogram ('H'). The response may also be in lower case. If the user responds 'Y' to the query **Symbols?**, symbols are drawn on the plot at every fifth data point for all plot types but spline and histogram. This example shows how FASTPLOT can be used to plot multiple sets of data using different types of curves with or without symbols.

The eleventh FASTPLOT example initially reads the input label (EXAMPL11.LBL) and data (EXAMPL11.DAT) files. It then queries the user as to plot type and plotting of symbols. GET_DISPLAY is called to check that the user has either an EGA or a VGA display. Next BGNPLOT is called to put the display in graphics mode, REG_FONTS to register the fonts, SCREEN, PORT and WINDOW to set up the screen coordinates and graphics window for the outside frame. The routine then calls FRAME to draw that frame in the specified color, SET_SYMBOL to load the symbol table (FP_SYMBL.TAB) into memory, and SET_FONT to select a font for the plot labels. The LABEL routine is called to put the labels on the plot. The example then computes the vertical min, max and step and sets the horizontal min, max and step. Calls to PORT and WINDOW establish the inside graphics window. Calls to AXIS are used to draw only the left axis side and bottom axes. The tick height and width (roughly in pixels) are specified and the ticks drawn with calls to routine TICKS.

The numbers on the vertical and horizontal axes are then drawn. This procedure requires shifting the plot graphics window defined previously by a specified amount to write the numbers below the bottom axis and to the left of the left-side axis. The user specifies the format *form* of the numbers (e.g., F5.2 or 1PE7.2). The length of the format specification is taken from the first number left of the decimal in the format. The user also specifies how often to place the numbers *step* and whether or not to skip *skip* placing the first number. The logical variable *skip* allows for cleaning looking plots in many cases. The NUMBERS routine uses the above information, the tick width and height passed from subroutine TICKS, and the font width and height passed from SET_FONT to shift the graphics window and place the numbers. It then internally resets the graphics window to the original window size for drawing the curves.

After calling the routine NUMBERS to put the numbers on the axes, Example 11 loops over the number of curves to be drawn (in this case three), calling the routine CURVE to

draw the selected curve. Note that the type of curve (linear, log-linear, linear-log or log-log) must be passed to the CURVE routine. If symbols are also to be drawn, the example calls MARKER with a different symbol ID for each curve and the SYMBOL routine to plot the symbols. Note that the type of curve (linear, log-linear, linear-log or log-log) must also be passed to the SYMBOL routine. Symbols could have been drawn on the histogram or the B-spline curve, but were not in this example. Calls to SYMBOL are completely independent of calls to CURVE. The user must remember to set the line style (SOLID, DOT, DASH) and the symbol id before calling the SYMBOL routine.

```

SUBROUTINE EXAMPL11
C*****
C FASTPLOT EXAMPLE 11 - PLOT CURVES AND SYMBOLS
C*****
CHARACTER*200 STRING(20)
character*80 mess,fnt,errmess
character*40 answer, accept1(14), accept2(4)
integer*4 Naccept1, Naccept2
INTEGER*4 ICOL(20), ISTR(20), STRX(20), STRY(20)
INTEGER*4 IBACK(20), IBACKCOL(20)
C PTYPE IS TYPE OF PLOT
CHARACTER*1 PTYPE

C KEY IS ARRAY WITH KEYS WHICH ARE USED FOR KEYBOARD INTERRUPT
CHARACTER*1 KEY(4)
LOGICAL mbeep, ebeep, BEEP
C DISPLAY IS GRAPHICS DISPLAY TYPE (CGA,EGA,VGA or TXT if no graphics)
CHARACTER*3 DISPLAY

C PARAMETERS FOR INPUT AND SELECT ROUTINES
INTEGER*4 NDATA, NCOLUMN, NCURVE

C PLOT VARIABLE X(NDATA), Y(NDATA,NCOLUMN)
REAL*4 X(200), Y(200,8), XSTEP, YSTEP
REAL*4 IMAX, XMIN, XMAX, YMIN, YMAX
LOGICAL DOSYMBL

C COMMON BLOCK CONTAINING MARKER INFO
LOGICAL FP_MARKFL
REAL*4 FP_MARKX(9), FP_MARKY(9)
INTEGER*4 FP_MARKN
COMMON /FP_MARK/ FP_MARKN, FP_MARKX, FP_MARKY, FP_MARKFL

C COMMON FOR PLOT DATA
COMMON /DATA/ X,Y
COMMON /IDATA/ NDATA, NCOLUMN, NCURVE

C Array accept1 contains acceptable answers to first query.
DATA accept1/'C','c','B','b','H','h','X','x','Y','y','L',
+ '1','Q','q'/

C GET LABELS
OPEN (UNIT=41,FILE='EXAMPL11.LBL',STATUS='OLD')
READ (UNIT=41,FMT=*) NUMSTR
DO 10 I=1,NUMSTR
READ (UNIT=41,FMT=*) ICOL(I), IBACKCOL(I), IBACK(I),
+ ISTR(I), STRX(I), STRY(I)
READ (UNIT=41,FMT=1) STRING(I)
1 FORMAT(A200)
10 CONTINUE

```

```

        CLOSE (UNIT=41)
c set mbeep, ebeep to .TRUE.
        mbeep = .TRUE.
        ebeep = .TRUE.
c query user
200  continue
        istart = 2
        fnt = 'FASTPLOT EXAMPLE 11 - CURVE PLOT'
        call TLABEL(14, 0, 0, 1, istart, 0, fnt)

        fnt = 'Select type of plot: '
        call TLABEL(14, 0, 0, 1, istart+2, 0, fnt)

        fnt = ' C - Linear Curve'
        call TLABEL(12, 0, 0, 0, istart+4, 29, fnt)

        fnt = ' B - Spline Fit'
        call TLABEL(12, 0, 0, 0, istart+5, 29, fnt)

        fnt = ' H - Histogram'
        call TLABEL(12, 0, 0, 0, istart+6, 29, fnt)

        fnt = ' X -          Y vs log10 X'
        call TLABEL(12, 0, 0, 0, istart+7, 29, fnt)

        fnt = ' Y - log10 Y vs X'
        call TLABEL(12, 0, 0, 0, istart+8, 29, fnt)

        fnt = ' L - log10 Y vs log10 X'
        call TLABEL(12, 0, 0, 0, istart+9, 29, fnt)

        fnt = ' Q - Quit and return to menu'
        call TLABEL(12, 0, 0, 0, istart+10, 29, fnt)

        mess = 'Enter type (or Quit): '
        answer = 'C'
        errmess = 'Incorrect response. Enter (C,B,H,X,Y,L or Q).'
        itype = 1
        Naccept1 = 14
c Array accept1, which contains acceptable answers, is set in DATA
statement.

        call QUERY(15,0,14,28,mess,answer,mbeep,14,0,1,20,0,
+      errmess,ebeep,itype,Naccept1,accept1)
C PTYPE - TYPE OF CURVE:
C C-LINEAR, B-SPLINE, H-HISTOGRAM, L-LOG/LOG, Y-LINEAR/LOG, X-LOG/LINEAR
        call CAPITAL (answer(1:1))
        ptype = answer
        if (ptype.eq.'Q') return
        if (ptype.eq.'B'.or.ptype.eq.'H')goto 101

```

```

c query user - draw symbols ?
  mess = 'Symbols (y/[n]): '
  answer = 'n'
  errmess = 'Incorrect response. Enter (y/n).'
  itype = 1
  Naccept2 = 4
  accept2(1) = 'y'
  accept2(2) = 'Y'
  accept2(3) = 'n'
  accept2(4) = 'N'

  call QUERY(15,0,16,31,mess,answer,mbeep,14,0,1,20,0,
+   errmess,ebeep,itype,Naccept2,accept2)
  if (answer.eq.'Y'.or.answer.eq.'y') dosymb1 = .TRUE.
  if (answer.eq.'N'.or.answer.eq.'n') dosymb1 = .FALSE.

101  continue
C GET DATA
      CALL INPUT
C SELECT PLOT DATA
      CALL SELECT

      CALL GET_DISPLAY(DISPLAY)
      IF(DISPLAY.EQ.'CGA'.OR.DISPLAY.EQ.'TXT')
+   STOP ' Correct graphics mode (EGA or VGA) not available.'

C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK
      CALL BGNPLOT (0)
C REGISTER FONTS - LOAD FONTS INTO MEMORY
      CALL REG_FONTS
C CLEAR THE SCREEN
      CALL CLEAR
C GET SIZE OF SCREEN IN PIXELS
      CALL SCREEN (IPIX_X, IPIX_Y)

C SET UP PLOT WINDOW (VIRTUAL COORDINATES) FOR FRAME
      XMIN = 0.
      XMAX = 1000.
      YMIN = 0.
      YMAX = 1000.
      CALL WINDOW (XMIN, YMAX, XMAX, YMIN)

C SET UP FRAME VIEWPORT
      CALL PORT (0, 0, ipix_x, ipix_y)
C DRAW FRAME
      CALL FRAME(15)

C LOAD SYMBOL TABLE INFORMATION
      CALL SET_SYMBOL
-----

```

```

C TYPEFACES AVAILABLE:
C 1 - COURIER, 2 - HELV, 3 - TMNS ROMAN (FIXED FONTS)
C 4 - MODERN, 5 - SCRIPT, 6 - ROMAN (SCALABLE FONTS)
c -----
C CALL SET_FONT ( font number, height, width )
c -----
C SELECT 2 - HELV 15x8
CALL SET_FONT ( 2, 15, 8 )

C PUT ON PLOT LABELS (HEADING, SUB_HEADING, XLABEL, YLABEL):
DO 20 J = 1, 4
CALL LABEL (icol(J), ibackcol(J), iback(J),
+ istr(J), strx(J), stry(J), string(J))
20 CONTINUE
C GENERAL LEGEND ON PLOT - IF CURVES ARE BEING DRAWN
IF(PTYPE.NE.'H')THEN
DO 30 J = 5, 7
CALL LABEL (icol(J), ibackcol(J), iback(J),
+ istr(J), strx(J), stry(J), string(J))
30 CONTINUE
ENDIF
C
C SPECIFY MIN, MAX, STEP FOR X AXIS OF PLOT:
IF(PTYPE.EQ.'X'.OR.PTYPE.EQ.'L')THEN
c LOG-LINEAR or LOG-LOG PLOT
XMIN = 1.
XMAX = 3.
XSTEP = 1
ELSE
c LINEAR or LINEAR-LOG or HISTORGRAM PLOT
XMIN = -20.
XMAX = 140.
XSTEP = 20.
ENDIF

C GET MIN, MAX STEP FOR Y AXIS:
YMIN = 0.
IMAX = 0.
DO 40 NP = 1, NDATA
DO 40 NC = 1, NCURVE
IMAX = AMAX1( IMAX, Y(NP,NC) )
40 CONTINUE
CALL GETMAX(IMAX, YMAX)
YSTEP=YMAX/5.

C IF LINEAR-LOG PLOT DESIRED MAKE SURE YMIN IS NOT ZERO!
IF(PTYPE.EQ.'Y'.OR.PTYPE.EQ.'L')THEN
C SET MIN, MAX AND STEP ON LOG AXIS
IF(YMIN.LE.0.)YMIN=1.
YMIN=ALOG10(YMIN)
YMAX=ALOG10(YMAX)

```

```

        YSTEP = 1
    ENDIF

C SET UP INSIDE PORT AND WINDOW
    CALL PORT (80, 60, ipix_x-60, ipix_y-60)

    CALL WINDOW (XMIN, YMAX, XMAX, YMIN)

C CALL AXIS(icolor,itpe) to place axis on bottom and left side only
    CALL AXIS(14,10)
    CALL AXIS(14,20)

c-----
C PUT TICK MARKS ON PLOT:
C first digit of itype indicates axis
C   1 - X axis (bottom)
C   2 - Y axis (left side)
C   3 - X axis (top)
C   4 - Y axis (right side)
C second digit indicates:
C   0 - ticks inside only
C   1 - ticks outside only
C   2 - ticks inside and outside
c height and widths of tick marks in pixels
c-----
c       CALL TICKS(icolor, itype, HEIGHT, WIDTH, XSTEP)
c-----
        IF(PCTYPE.EQ.'X'.OR.PCTYPE.EQ.'L')THEN
            CALL TICKS(15, 11, 8, 6, XSTEP,-1)
        ELSE
            CALL TICKS(15, 11, 8, 6, XSTEP,2)
        ENDIF
        IF(PCTYPE.EQ.'Y'.OR.PCTYPE.EQ.'L') THEN
            CALL TICKS(15, 22, 8, 6, YSTEP,-1)
        ELSE
            CALL TICKS(15, 22, 8, 6, YSTEP,2)
        ENDIF

C PUT NUMBERS ON PLOT
c-----
c itype:
c   11 for linear x-axis
c   12 for log x-axis
c   21 for linear y-axis
c   22 for log y-axis
c   format for numbers
c   step   determines placing of numbers along the axis
c   skip   if .TRUE. do not print first number along axis
c-----
c       CALL NUMBERS(icolor,itpe,format,step,skip)
c-----
C Y-axis

```

```

        IF(PTYPE.EQ.'Y'.OR.PTYPE.EQ.'L') THEN
            CALL NUMBERS(15, 22, '1PE6.0', YSTEP,.FALSE.)
        ELSE
            CALL NUMBERS(15, 21, 'F7.0 ', YSTEP,.FALSE.)
        ENDIF
C X-axis
        IF(PTYPE.EQ.'X'.OR.PTYPE.EQ.'L')THEN
            CALL NUMBERS(15, 12, '1PE6.0', XSTEP,.FALSE.)
        ELSE
            CALL NUMBERS(15, 11, 'F5.0 ', XSTEP,.TRUE.)
        ENDIF
C-----
C DRAW PLOT
        DO 100 I = 1, NCURVE
            icolor = 12+i
            npoints = NDATA
C-----
            IF (PTYPE.EQ.'H') THEN
C SET LINETYPE 0-solid, 1-dash, 2-dot
                CALL LINETYPE( 0 )
                CALL CURVE (icolor,31,.FALSE.,1.,npoints,X,Y(1,I))
            ENDIF
C-----
            IF (PTYPE.EQ.'C') THEN
                IF (DOSYMBL) THEN
C WHEN SPECIFYING FILLED SYMBOLS:
C   DRAW SYMBOLS BEFORE CURVE IF YOU WANT FILLED SYMBOLS
C   DRAW SYMBOLS AFTER  CURVE IF YOU DO NOT WANT FILLED SYMBOLS
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS)
                    CALL MARKER ( I+6, 3.0 )
                    CALL LINETYPE (0)
C PLOT SYMBOL
                    CALL SYMBOL(icolor, 11, 5, npoints, X, Y(1,I))
                ENDIF
C SET LINETYPE 0-solid, 1-dash, 2-dot
                    CALL LINETYPE( I-1 )
                    CALL CURVE(icolor,11,.FALSE.,0.,npoints, X, Y(1,I))
                ENDIF
C-----
            IF(PTYPE.EQ.'B')THEN
                CALL LINETYPE (I-1)
                CALL CURVE(icolor,55,.FALSE.,0.,npoints, X, Y(1,I))
            ENDIF
C-----
            IF(PTYPE.EQ.'Y')THEN
                IF (DOSYMBL) THEN
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS)
                    CALL MARKER ( I+6, 3.0 )
                    CALL LINETYPE (0)
                    CALL SYMBOL(icolor, 12, 5, npoints, X, Y(1,I))
                ENDIF
            ENDIF

```

```

C SET LINETYPE 0-solid, 1-dash, 2-dot
      CALL LINETYPE( I-1 )
      CALL CURVE(icolor, 12, .FALSE., 0., npoints, X, Y(1,I))
      ENDIF
C
      IF(PTYPE.EQ.'X')THEN
        IF (DOSYMBL) THEN
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS)
          CALL MARKER ( I+6, 3.0 )
          CALL LINETYPE (0)
          CALL SYMBOL(icolor, 21, 20, npoints, X, Y(1,I))
          ENDIF
C SET LINETYPE 0-solid, 1-dash, 2-dot
          CALL LINETYPE( I-1 )
          CALL CURVE(icolor, 21, .FALSE., 0., npoints, X, Y(1,I))
          ENDIF

          IF(PTYPE.EQ.'L')THEN
            IF (DOSYMBL) THEN
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS)
              CALL MARKER ( I+6, 3.0 )
              CALL LINETYPE (0)
              CALL SYMBOL(icolor, 22, 20, npoints, X, Y(1,I))
              ENDIF
C SET LINETYPE 0-solid, 1-dash, 2-dot
              CALL LINETYPE( I-1 )
              CALL CURVE(icolor, 22, .FALSE., 0., npoints, X, Y(1,I))
              ENDIF

100    CONTINUE
C-----
      CALL UNREG_FONTS

C USE <ESC> and <CR> TO BREAK AWAY FROM PLOT
C ESC CHARACTER
      KEY(1)=CHAR(27)
C CARRIAGE RETURN CHARACTER
      KEY(2)=CHAR(13)
      BEEP=.TRUE.
      CALL ENDPLOT ( BEEP, 2, KEY )

      goto 200
      RETURN
      END

      SUBROUTINE INPUT
C SUBROUTINE INPUT - READS DATA INTO COLUMNS FOR SELECTING PLOT VARIABLES
C READS UP TO NINE COLUMNS OF DATA (X, AND 8 POSSIBLE Y VALUES)
C UP TO 200 DATA POINTS IN EACH COLUMN
      REAL*4 COL(200,9)

```

```

C NCOLUMN IS NUMBER OF COLUMNS IN DATAFILE - MAXIMUM VALUE IS 9
  INTEGER*4 NCOLUMN

C NPOINTS IS NUMBER OF DATA POINTS - MAXIMUM VALUE IS 3000
  INTEGER*4 NDATA

C SKIP NHEADER LINES OF HEADER BEFORE DATA
  INTEGER*4 NHEADER

C NUMBER OF CURVE TO PLOT
  INTEGER*4 NCURVE
  CHARACTER*78 HEADER

COMMON /COLDATA/ COL
COMMON /IDATA/ NDATA, NCOLUMN, NCURVE

OPEN (UNIT=10,FILE='EXAMPL11.DAT',STATUS='OLD')

read(unit=10,fmt=*) NHEADER, NCOLUMN, NDATA

DO 10 I = 1, NHEADER
  READ (UNIT=10,FMT=1) HEADER
1  FORMAT(A78)
10 CONTINUE
C
DO 20 I = 1, NDATA
  READ (UNIT=10,FMT=*) (COL(I,J),J=1,NCOLUMN)
20 CONTINUE
CLOSE (UNIT=10)
RETURN
END

SUBROUTINE SELECT
C SUBROUTINE SELECT - SELECTS COLUMNS FOR PLOTTING
C COL(DATA POINTS, COLUMNS)
  REAL*4 COL(200,9)

C LOAD SELECTED COLUMN IN X ARRAY, AND THE OTHER SELECTED COLUMNS
C (UP TO 8 COLUMNS) INTO Y ARRAY
C X(200),Y(200,8)
  REAL*4 X(200),Y(200,8)

C NPOINTS IS NUMBER OF DATA POINTS
  INTEGER*4 NDATA

C NCOLUMN IS NUMBER OF COLUMNS IN DATAFILE
  INTEGER*4 NCOLUMN

C NUMBER OF CURVES TO PLOT
  INTEGER*4 NCURVE

```

```
COMMON /COLDATA/ COL
COMMON /DATA/ X,Y
COMMON /IDATA/ NDATA, NCOLUMN, NCURVE
```

```
c Number of CURVES to be plotted:
NCURVE = 3
```

```
DO 10 I = 1, NCURVE+1
```

```
C ASSIGN COLUMNS
```

```
IF(I.EQ.1)ICOL=1
IF(I.EQ.2)ICOL=3
IF(I.EQ.3)ICOL=4
IF(I.EQ.4)ICOL=5
```

```
DO 20 J = 1, NDATA
  IF(I.EQ.1)THEN
    X(J)=COL(J,ICOL)
  ELSE
    Y(J,I-1)=COL(J,ICOL)
  ENDIF
```

```
20 CONTINUE
```

```
10 CONTINUE
RETURN
END
```

```
SUBROUTINE GETMAX ( IMAX,YMAX )
REAL*4 IMAX, YMAX
```

```
YMAX = 10.
IF(IMAX.GE.0. .AND. IMAX.LT.10.)RETURN
```

```
IF (IMAX.GE.10. .AND. IMAX.LT.100.) THEN
  YMAX = INT( IMAX / 10. + 1 ) * 10
  RETURN
ENDIF
```

```
IF (IMAX.GE.100. .AND. IMAX.LT.1000.) THEN
  YMAX = INT( IMAX / 100. + 1 ) * 100
  RETURN
ENDIF
```

```
IF (IMAX.GE.1000. .AND. IMAX.LT.10000.) THEN
  YMAX = INT( IMAX / 1000. + 1 ) * 1000
  RETURN
ENDIF
```

```
IF (IMAX.GE.10000. .AND. IMAX.LT.100000.) THEN
  YMAX = INT( IMAX / 10000. + 1 ) * 10000
  RETURN
ENDIF
```

```

ENDIF
IF (IMAX.GE.100000.)THEN
  YMAX = INT( IMAX / 100000. + 1 ) * 100000
  RETURN
ENDIF
END

```

EXAMPL11.LBL - Plot label file for Example 11.

```

7          NUMBER OF LABELS          EXAMPL11 LABEL FILE
0 15 1 1 0 10      COLOR BACK-COLOR BACK CENTER  XLOC  YLOC
MORTALITY DATA
0 15 1 1 0 25
LUNG CANCERS (ICD 162) 1979-1981
14 0 0 1 0 452 (320 for EGA graphics)
Age (y)
14 0 0 0 20 39
Deaths
0 15 1 0 120 110
SOLID - BOTH
0 15 1 0 120 123
DASH - MALE
0 15 1 0 120 136
DOT - FEMALE
0.000

```

EXAMPL11.DAT - Input data for Example 11.

```

3 5 121          EXAMPL11 DATA FILE
All Years ICD: 162.0 - ICD: 162.9 for RACE: All
LUNG CANCER (ICD 162)

```

Ti	Tf	ALL	MALES	FEMALES
0	1	9	6	3
1	2	1	0	1
2	3	2	2	0
3	4	0	0	0
4	5	0	0	0
5	6	4	1	3
6	7	4	2	2
7	8	1	1	0
8	9	1	1	0
9	10	1	0	1
10	11	0	0	0
11	12	2	2	0
12	13	1	1	0
13	14	2	1	1

14	15	0	0	0
15	16	2	2	0
16	17	4	1	3
17	18	6	2	4
18	19	6	4	2
19	20	6	4	2
20	21	7	6	1
21	22	9	4	5
22	23	17	10	7
23	24	13	9	4
24	25	11	8	3
25	26	18	12	6
26	27	24	16	8
27	28	34	19	15
28	29	36	24	12
29	30	47	31	16
30	31	48	26	22
31	32	74	40	34
32	33	124	74	50
33	34	125	79	46
34	35	179	116	63
35	36	208	121	87
36	37	286	169	117
37	38	380	242	138
38	39	464	300	164
39	40	592	355	237
40	41	703	480	223
41	42	851	542	309
42	43	990	614	376
43	44	1132	747	385
44	45	1465	940	525
45	46	1662	1071	591
46	47	2045	1292	753
47	48	2328	1529	799
48	49	2793	1887	906
49	50	3216	2210	1006
50	51	3810	2586	1224
51	52	4388	2990	1398
52	53	4782	3278	1504
53	54	5609	3911	1698
54	55	6174	4367	1807
55	56	6686	4673	2013
56	57	7307	5146	2161
57	58	7858	5615	2243
58	59	8261	5897	2364
59	60	9054	6517	2537
60	61	9426	6822	2604
61	62	9561	6983	2578
62	63	10164	7383	2781
63	64	10242	7534	2708
64	65	10955	8086	2869

65	66	11512	8449	3063
66	67	11428	8468	2960
67	68	11569	8620	2949
68	69	11490	8608	2882
69	70	11121	8429	2692
70	71	10970	8272	2698
71	72	10661	8121	2540
72	73	10290	7844	2446
73	74	9617	7324	2293
74	75	9108	6941	2167
75	76	8480	6519	1961
76	77	7757	5947	1810
77	78	7197	5424	1773
78	79	6440	4921	1519
79	80	5735	4324	1411
80	81	5073	3791	1282
81	82	4364	3246	1118
82	83	3823	2815	1008
83	84	3419	2488	931
84	85	2924	2124	800
85	86	2544	1770	774
86	87	2143	1490	653
87	88	1710	1171	539
88	89	1349	873	476
89	90	1020	667	353
90	91	802	481	321
91	92	624	375	249
92	93	445	251	194
93	94	317	182	135
94	95	241	132	109
95	96	176	94	82
96	97	89	42	47
97	98	64	22	42
98	99	40	20	20
99	100	29	13	16
100	101	16	10	6
101	102	8	5	3
102	103	5	5	0
103	104	1	1	0
104	105	3	2	1
105	106	4	3	1
106	107	0	0	0
107	108	1	0	1
108	109	0	0	0
109	110	3	0	3
110	111	1	0	1
111	112	0	0	0
112	113	0	0	0
113	114	0	0	0
114	115	0	0	0
115	116	1	1	0

116	117	0	0	0
117	118	0	0	0
118	119	0	0	0
119	120	0	0	0
120	121	0	0	0

The selection menu for Example 11 is shown in Fig. 15. The screen output upon selecting linear curve (C) is shown in Fig. 16. The screen output upon selecting a log-log (L) plot is shown in Fig. 17.

FASTPLOT EXAMPLE 11 - CURVE PLOT

Select type of plot:

- C - Linear Curve
- B - Spline Fit
- H - Histogram
- X - Y vs log₁₀ X
- Y - log₁₀ Y vs X
- L - log₁₀ Y vs log₁₀ X
- Q - Quit and return to menu

Enter type (or Quit): C

Symbols ([y]/n): y

Fig. 15. Selection menu for Example 11.

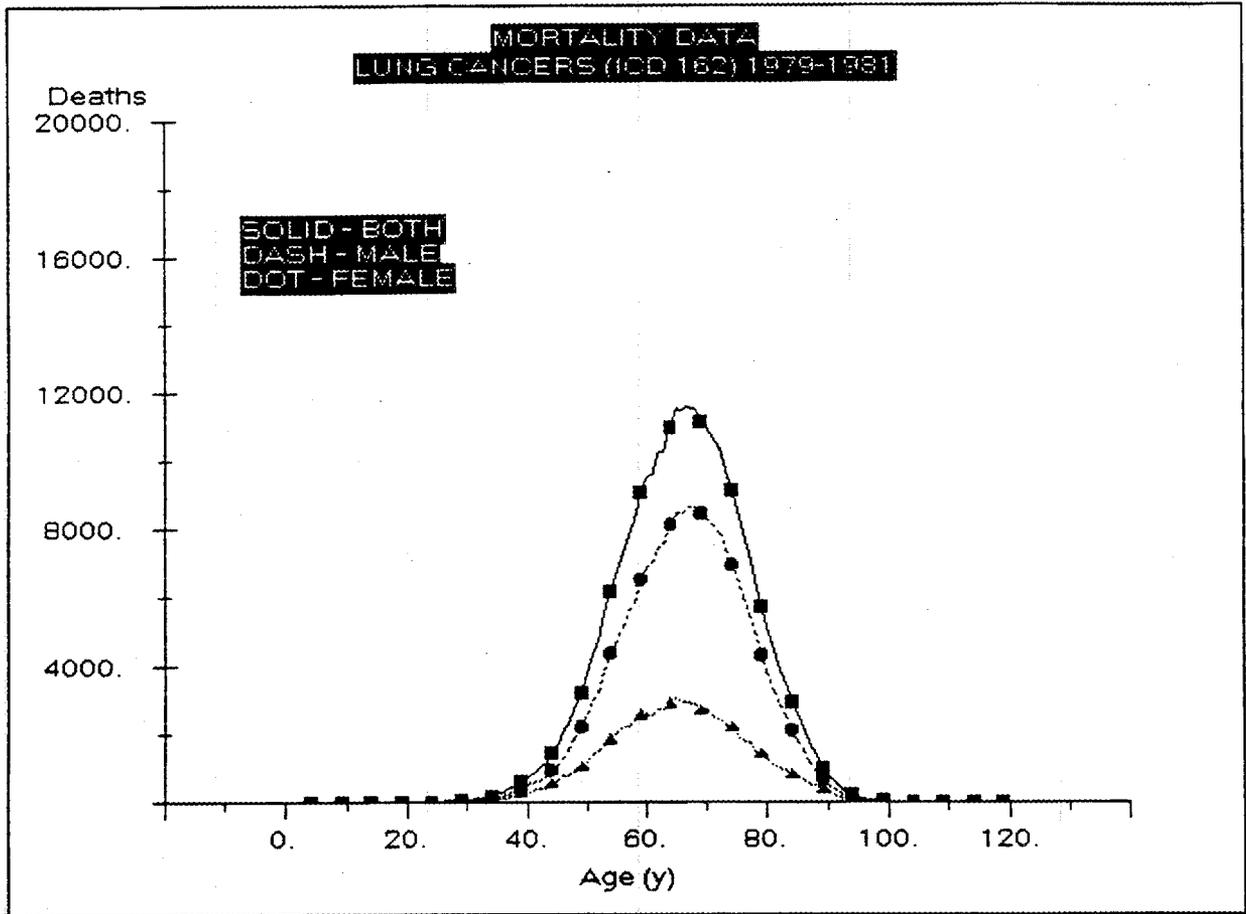


Fig. 16. Screen output for Example 11 upon selecting Linear Curve (C).

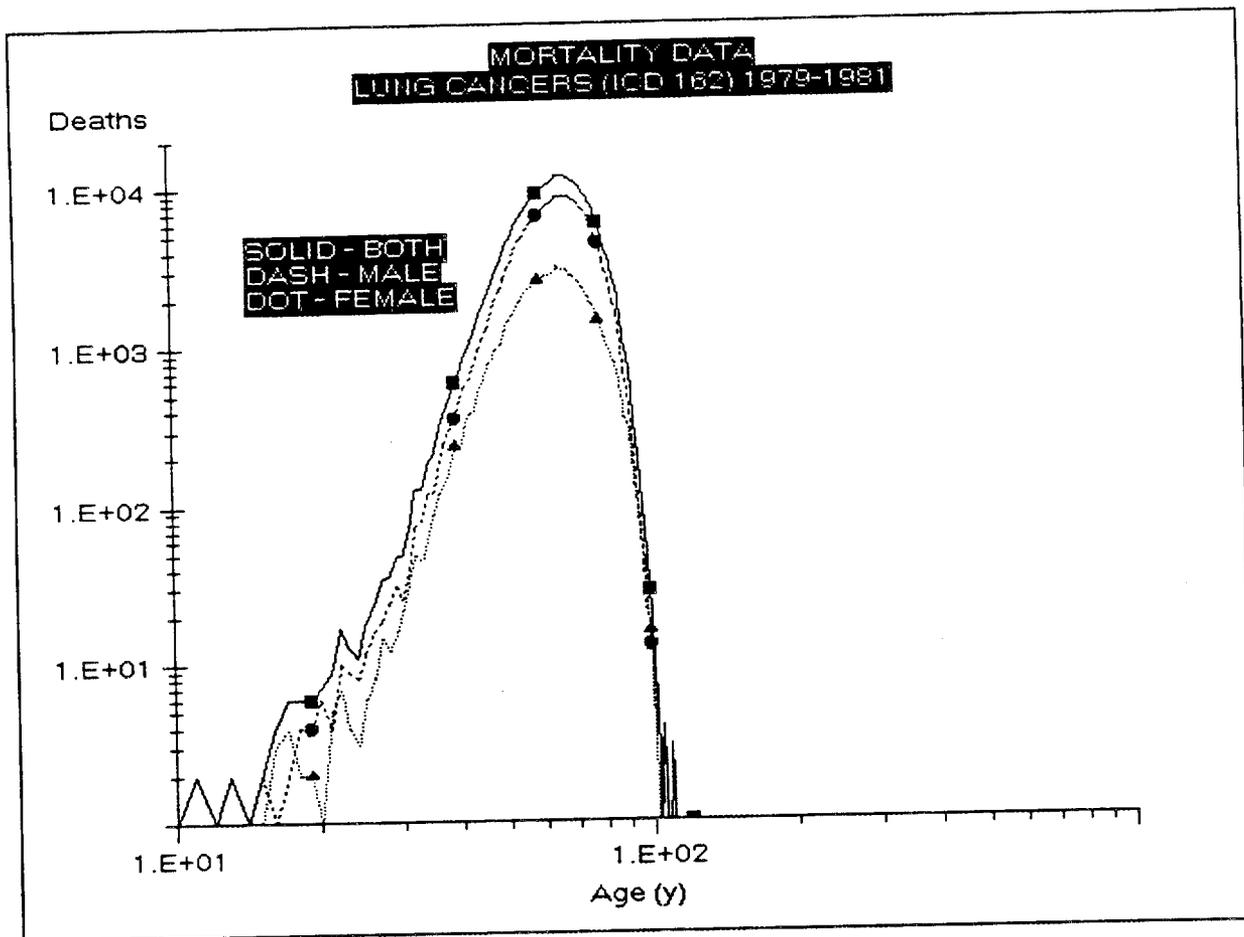


Fig. 17. Screen output for Example 11 upon selecting log-log (L) plot.

Example 12:

The twelfth FASTPLOT example is like the eleventh but adds error bars on the symbols. Only a single curve is plotted, but the user can select the type of plot. The data are contained in the file `EXAMPL12.DAT`. The labels for the plot are contained in the file `EXAMPL12.LBL`.

The user is asked, using the FASTPLOT routine `QUERY`, for the type of plot to be generated, and, depending on the plot type, whether symbols should be placed on the plot. Five types of curves can be drawn: linear curve ('C'), B-spline-fit linear curve ('B'), Y log - X linear curve ('Y'), X log - Y linear curve ('X'), Y log - X log ('L'). Symbols are drawn on the plot at every fifth data point for linear plots and every tenth point for log-linear or log-log plots. Error bars are drawn at each symbol using min/max Y values supplied in the `EXAMPL12.DAT` data file.

The twelfth FASTPLOT example initially reads the input label (`EXAMPL12.LBL`) and data (`EXAMPL12.DAT`) files. It then queries the user as to plot type. `GET_DISPLAY` is called to check that the user has either an EGA or a VGA display. Next `BGNPLOT` is called to put the display in graphics mode, `REG_FONTS` to register the fonts, `SCREEN`, `PORT` and `WINDOW` to set up the screen coordinates and graphics window for the outside frame. The routine then calls `FRAME` to draw that frame in the specified color, `SET_SYMBOL` to load the symbol table (`FP_SYMBL.TAB`) into memory, and `SET_FONT` to select a font for the plot labels. In this example the Modern Font (4), a scalable font, was chosen. The `LABEL` routine is called to put the labels on the plot. The example then computes the vertical min, max and step and sets the horizontal min, max and step. Calls to `PORT` and `WINDOW` establish the inside graphics window. Another frame is drawn around the inside window. The tick height and width (roughly in pixels) is specified and the ticks drawn with calls to routine `TICKS`.

The numbers on the vertical and horizontal axes are then drawn (see `EXAMPLE 11` above). After calling the routine `NUMBERS` to put the numbers on the axes, Example 12 plots the curve and symbols with error bars. Separate calls to `MARKER` control the type of symbol used and the symbols used at the end of the error bars. Vertical and horizontal bar symbols, with symbol ids of 12 and 13, respectively, are provided specifically for error bars, but any of the symbols may be used. The user can select either symbol 12 or 13, the `ERRBAR` routine selects the correct one for the error bar requested.

Note that the type of curve, linear, log-linear, linear-log or log-log, must be passed to the `SYMBOL` routine and the `ERRBAR` routine. Calls to `SYMBOL` and to `ERRBAR` are completely independent of calls to `CURVE`. The user must remember to set the line style (`SOLID`, `DOT`, `DASH`) and symbol id before calling the `SYMBOL` or `ERRBAR` routines.

```

SUBROUTINE EXAMPL12
C*****
C FASTPLOT EXAMPLE 12 - PLOT CURVES AND SYMBOLS WITH ERROR BARS *
C*****
CHARACTER*200 STRING(20)
character*80 mess,fnt,errmess
character*40 answer, accept(12)
integer*4 Naccept
INTEGER*4 ICOL(20), ISTR(20), STRX(20), STRY(20)
INTEGER*4 IBACK(20), IBACKCOL(20), SYM_ID, ERR_ID
REAL*4 SYM_H, ERR_H
C PTYPE IS TYPE OF PLOT
CHARACTER*1 PTYPE

C KEY IS ARRAY WITH KEYS WHICH ARE USED FOR KEYBOARD INTERRUPT
CHARACTER*1 KEY(4)
LOGICAL mbeep, ebeep, BEEP
C DISPLAY IS GRAPHICS DISPLAY TYPE (CGA,EGA,VGA or TXT if no graphics)
CHARACTER*3 DISPLAY

C PARAMETERS FOR INPUT AND SELECT ROUTINES
INTEGER*4 NDATA
C PLOT VARIABLE X(NDATA), Y(NDATA,NCOLUMN)
REAL*4 X(200), Y(200), XSTEP, YSTEP
REAL*4 IMAX, XMIN, XMAX, YMIN, YMAX
REAL*4 Xrange(2,200), Yrange(2,200)

C COMMON BLOCK CONTAINING MARKER INFO
LOGICAL FP_MARKFL
REAL*4 FP_MARKX(9), FP_MARKY(9)
INTEGER*4 FP_MARKN
COMMON /FP_MARK/ FP_MARKN, FP_MARKX, FP_MARKY, FP_MARKFL

C COMMON FOR PLOT DATA
COMMON /DATA2/ X,Y,Xrange,Yrange
COMMON /DATA3/ XMIN, XMAX
COMMON /IDATA2/ NDATA

C Array accept contains acceptable answers to query.
DATA accept/'C','c','B','b','X','x','Y','y','L','l','Q','q'/

C GET LABELS
OPEN (UNIT=41,FILE='EXAMPL12.LBL',STATUS='OLD')
c
READ (UNIT=41,FMT=*) NUMSTR
DO 10 I=1,NUMSTR
READ(UNIT=41,FMT=*) ICOL(I), IBACKCOL(I), IBACK(I),
ISTR(I), STRX(I), STRY(I)
READ(UNIT=41,FMT=1) STRING(I)
1 FORMAT(A200)

```

```

10      CONTINUE
c symbol at point is SYM_ID, symbol height is SYM_H
c symbol at top and bottom of error bar is ERR_ID, symbol height is ERR_H
c
      READ (UNIT=41,FMT=*) SYM_ID, SYM_H, ERR_ID, ERR_H
      CLOSE (UNIT=41)
c set mbeep, ebeep to .TRUE.
      mbeep = .TRUE.
      ebeep = .TRUE.
100    continue
c query user
      irstart = 2

      fnt = 'FASTPLOT EXAMPLE 12 - ERROR BARS'
      call TLABEL(14, 0, 0, 1, irstart, 0, fnt)

      fnt = 'Select type of plot: '
      call TLABEL(14, 0, 0, 1, irstart+2, 0, fnt)

      fnt = ' C - Linear Curve'
      call TLABEL(12, 0, 0, 0, irstart+4, 29, fnt)

      fnt = ' B - Spline Fit'
      call TLABEL(12, 0, 0, 0, irstart+5, 29, fnt)

      fnt = ' X -          Y vs log10 X'
      call TLABEL(12, 0, 0, 0, irstart+6, 29, fnt)

      fnt = ' Y - log10 Y vs X'
      call TLABEL(12, 0, 0, 0, irstart+7, 29, fnt)

      fnt = ' L - log10 Y vs log10 X'
      call TLABEL(12, 0, 0, 0, irstart+8, 29, fnt)

      fnt = ' Q - Quit and return to menu'
      call TLABEL(12, 0, 0, 0, irstart+9, 29, fnt)

c set up call to QUERY
      mess = 'Enter type (or Quit): '
      answer = 'C'
      errmess = 'Incorrect response. Enter (C,B,X,Y, L, or Q).'
      itype = 1
      Naccept = 12
c Array accept, which contains acceptable answers, is set in DATA
statement.

      call QUERY(15,0,14,28,mess,answer,mbeep,14,0,1,20,0,
+      errmess,ebeep,itype,Naccept,accept)
C PTYPE - TYPE OF CURVE:
C C-LINEAR, B-SPLINE, L-LOG/LOG, Y-LINEAR/LOG, X-LOG/LINEAR
      call CAPITAL (answer(1:1))

```

```

        ptype = answer
        if (ptype.eq.'Q') return

C GET DATA
        CALL INPUT2
C CHECK DISPLAY
        CALL GET_DISPLAY(DISPLAY)
        IF(DISPLAY.EQ.'CGA'.OR.DISPLAY.EQ.'TXT')
            STOP ' Correct graphics mode (EGA or VGA) not available.'

C BEGIN PLOT - SET GRAPHICS MODE, BACKGROUND COLOR TO BLACK
        CALL BGNPLOT (0)

C REGISTER FONTS - LOAD FONTS INTO MEMORY
        CALL REG_FONTS

C CLEAR THE SCREEN
        CALL CLEAR

C GET SIZE OF SCREEN IN PIXELS
        CALL SCREEN (IPIX_X, IPIX_Y)

C SET UP PLOT WINDOW (VIRTUAL COORDINATES) FOR FRAME
        CALL WINDOW (0., 1000., 1000., 0.)

C SET UP FRAME VIEWPORT
        CALL PORT (0, 0, ipix_x, ipix_y)
C DRAW FRAME
        CALL FRAME(15)

C LOAD SYMBOL TABLE INFORMATION
        CALL SET_SYMBOL

C -----
C TYPEFACES AVAILABLE:
C 1 - COURIER, 2 - HELV,   3 - TMNS ROMAN (FIXED FONTS)
C 4 - MODERN,  5 - SCRIPT, 6 - ROMAN      (SCALABLE FONTS)
C -----
C          CALL SET_FONT ( font number, height, width )
C -----
C  SELECT 4 - MODERN a scalable font with font size roughly 15x8.
C          CALL SET_FONT ( 4, 15, 8 )

C PUT ON PLOT LABELS (HEADING, SUB_HEADING, XLABEL, YLABEL):
        DO 20 J = 1, 4
            CALL LABEL (icol(J), ibackcol(J), iback(J),
+             istr(J), strx(J), stry(J), string(J))
        20 CONTINUE
C
C SPECIFY MIN, MAX, STEP FOR X AXIS OF PLOT:
        IF(PTYPE.EQ.'X'.OR.PTYPE.EQ.'L')THEN

```

```

c LOG-LINEAR or LOG-LOG PLOT
      XMIN = ALOG10(XMIN+1.0)
      XMAX = ALOG10(XMAX)
      XSTEP = 1
    ELSE
c LINEAR or LINEAR-LOG or HISTORGRAM PLOT
c get XMIN, XMAX from data file
      XSTEP = XMAX/5.
    ENDIF

C GET MIN, MAX STEP FOR Y AXIS:
      YMIN = 0.
      IMAX = 0.
      DO 40 NP = 1, NDATA
        IMAX = AMAX1( IMAX, Y(NP) )
40    CONTINUE
      CALL GETMAX(IMAX, YMAX)
      YSTEP=YMAX/5.

C IF LINEAR-LOG PLOT DESIRED MAKE SURE YMIN IS NOT ZERO!
      IF(PTYPE.EQ.'Y'.OR.PTYPE.EQ.'L')THEN
C SET MIN, MAX AND STEP ON LOG AXIS
      IF(YMIN.LE.0.)YMIN=.1
      YMIN=ALOG10(YMIN)
      YMAX=ALOG10(YMAX)
      YSTEP = 1
    ENDIF

C SET UP INSIDE PORT AND WINDOW
      CALL PORT (80, 60, ipix_x-60, ipix_y-60)

      CALL WINDOW (XMIN, YMAX, XMAX, YMIN)

C DRAW FRAME(icolor)
      CALL FRAME (14)
c-----
C PUT TICK MARKS ON PLOT:
      IF(PTYPE.EQ.'X'.OR.PTYPE.EQ.'L')THEN
        CALL TICKS(15, 11, 8, 6, XSTEP,-1)
        CALL TICKS(15, 30, 8, 6, XSTEP,-1)
      ELSE
        CALL TICKS(15, 11, 8, 6, XSTEP,2)
        CALL TICKS(15, 30, 8, 6, XSTEP,2)
      ENDIF
      IF(PTYPE.EQ.'Y'.OR.PTYPE.EQ.'L') THEN
        CALL TICKS(15, 22, 8, 6, YSTEP,-1)
        CALL TICKS(15, 40, 8, 6, YSTEP,-1)
      ELSE
        CALL TICKS(15, 22, 8, 6, YSTEP,2)
        CALL TICKS(15, 40, 8, 6, YSTEP,2)
      ENDIF

```

```

C PUT NUMBERS ON PLOT
C Y-axis
      IF(PTYPE.EQ.'Y'.OR.PTYPE.EQ.'L') THEN
        CALL NUMBERS(15, 22, '1PE6.0', YSTEP,.FALSE.)
      ELSE
        CALL NUMBERS(15, 21, 'F7.0 ', YSTEP,.FALSE.)
      ENDIF
C X-axis
      IF(PTYPE.EQ.'X'.OR.PTYPE.EQ.'L')THEN
        CALL NUMBERS(15, 12, '1PE6.0', XSTEP,.FALSE.)
      ELSE
        CALL NUMBERS(15, 11, 'F5.0 ', XSTEP,.TRUE.)
      ENDIF
-----
C DRAW PLOT
      npoints = NDATA
-----
      IF (PTYPE.EQ.'C') THEN
        itype = 11
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS)
        CALL MARKER ( SYM_ID, SYM_H )
        CALL LINETYPE (0)
C PLOT SYMBOL
        ISYM = 1
        CALL SYMBOL(12, itype, ISYM, npoints, X, Y)

c error bars for symbols
C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW ERROR BARS
        CALL LINETYPE( 0 )
c call MARKER to set symbol for top and bottom of error bar
        CALL MARKER ( ERR_ID, ERR_H )
        call ERRBAR(12,itype,ISYM, npoints, X, Y, Xrange, Yrange)

C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW CURVE
        CALL LINETYPE( 2 )
        CALL CURVE(15,itype,.FALSE.,0.,npoints, X, Y)
      ENDIF
-----
      IF(PTYPE.EQ.'B')THEN
        itype = 11
c
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS) AND DRAW SYMBOLS
        CALL MARKER ( SYM_ID, SYM_H )
        CALL LINETYPE (0)
        ISYM = 1
        CALL SYMBOL(12, itype, ISYM, npoints, X, Y)

c error bars for symbols
C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW ERROR BARS
        CALL LINETYPE( 0 )
c call MARKER to set symbol for top and bottom of error bar

```

```

        CALL MARKER ( ERR_ID, ERR_H )
        call ERRBAR(12,ittype,ISYM,
+           npoints, X, Y, Xrange, Yrange)
C DRAW SPLINE CURVE
        itype = 55
        CALL LINETYPE (2)
        CALL CURVE(15,ittype,.FALSE.,0.,npoints, X, Y)
        ENDIF
c-----
        IF(PTYPE.EQ.'Y')THEN
            itype = 12
C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS) AND DRAW SYMBOLS
            CALL MARKER ( SYM_ID, SYM_H )
            CALL LINETYPE (0)
            ISYM = 1
            CALL SYMBOL(12, itype, ISYM, npoints, X, Y)

c error bars for symbols
C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW ERROR BARS
            CALL LINETYPE( 0 )
c call MARKER to set symbol for top and bottom of error bar
            CALL MARKER ( ERR_ID, ERR_H )
            call ERRBAR(12,ittype,ISYM, npoints, X, Y, Xrange, Yrange)

C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW CURVE
            CALL LINETYPE( 2 )
            CALL CURVE(15,ittype,.FALSE.,0.,npoints,X,Y)
        ENDIF
C
        IF(PTYPE.EQ.'X')THEN
            itype = 21

C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS) AND DRAW SYMBOLS
            CALL MARKER ( SYM_ID, SYM_H )
            CALL LINETYPE (0)
            ISYM = 1
            CALL SYMBOL(12, itype, ISYM, npoints, X, Y)

c error bars for symbols
C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW ERROR BARS
            CALL LINETYPE( 0 )
c call MARKER to set symbol for top and bottom of error bar
            CALL MARKER ( ERR_ID, ERR_H )
            call ERRBAR(12,ittype,ISYM, npoints, X, Y, Xrange, Yrange)

C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW CURVE
            CALL LINETYPE( 2 )
            CALL CURVE(15, 21,.FALSE.,0.,npoints,X,Y)
        ENDIF

        IF(PTYPE.EQ.'L')THEN

```

```

        itype = 22

C SELECT SYMBOL (0-11) AND SYMBOL HEIGHT (PIXELS), DRAW SYMBOLS
      CALL MARKER ( SYM_ID, SYM_H )
      CALL LINETYPE (0)
      ISYM = 1
      CALL SYMBOL(12, itype, ISYM, npoints, X, Y)

c error bars for symbols
C SET LINETYPE 0-solid, 1-dash, 2-dot AND ERROR BARS
      CALL LINETYPE( 0 )
c call MARKER to set symbol for top and bottom of error bar
      CALL MARKER ( ERR_ID, ERR_H )
      call ERRBAR(12,itype,ISYM, npoints, X, Y, Xrange, Yrange)

C SET LINETYPE 0-solid, 1-dash, 2-dot AND DRAW CURVE
      CALL LINETYPE( 2 )
      CALL CURVE(15,itype,.FALSE.,0., npoints,X,Y)
      ENDIF

c-----
      CALL UNREG_FONTS

C USE <ESC> and <CR> TO BREAK AWAY FROM PLOT
C ESC CHARACTER
      KEY(1)=CHAR(27)
C CARRIAGE RETURN CHARACTER
      KEY(2)=CHAR(13)
      BEEP=.TRUE.
      CALL ENDPLOT ( BEEP, 2, KEY )
      goto 100
      RETURN
      END

      SUBROUTINE INPUT2

      INTEGER*4 NDATA, NHEADER
      CHARACTER*78 HEADER

      REAL*4 X(200), Y(200), Xrange(2,200), Yrange(2,200)
      REAL*4 XMIN, XMAX

      COMMON /IDATA2/ NDATA
      COMMON /DATA2/ X,Y,Xrange,Yrange
      COMMON /DATA3/ XMIN, XMAX

      OPEN (UNIT=10,FILE='EXAMPL12.DAT',STATUS='OLD')

      read(unit=10,fmt=*) NHEADER, NDATA
      read(unit=10,fmt=*) XMIN, XMAX
      DO 10 I = 1, NHEADER

```

```
      READ (UNIT=10,FMT=1)HEADER
1      FORMAT(A78)
10     CONTINUE
C
      DO 20 I = 1, NDATA
      READ(UNIT=10,FMT=*) X(I), Xrange(1,I), Xrange(2,I),
+      Y(I), Yrange(1,I), Yrange(2,I)
20     CONTINUE
      CLOSE (UNIT=10)
      RETURN
      END
```

EXAMPL12.LBL - Plot label file for Example 12.

```

4          NUMBER OF LABELS                      EXAMPL12 LABEL FILE
15 0 0 1 0 10      COLOR BACK-COLOR BACK CENTER  XLOC  YLOC
PLOT DATA
15 0 0 1 0 25
SUB-TITLE
14 0 0 1 0 452 (320 for EGA graphics)
X Variable
14 0 0 0 20 37
Y Variable
8 2.5 12 2.0  SYM_ID, SYM_H, ERR_ID, ERR_H

```

EXAMPL12.DAT - Input data for Example 12.

```

3 20      Header  DATA                      EXAMPL12 DATA FILE
0 250  XMIN  XMAX
HEADER 1
HEADER 2
  X      Xmin  Xmax      Y      Ymin  Ymax
10       5    15      10       5    30
20      15    25      20       5    50
30      25    35      40      20   100
40      35    45      60      50   70
50      45    55     100      40   150
60      55    65     200     100   300
70      65    75     400     360   800
80      75    85     800     750   950
90      85    95    1200     800  1350
100     95   105    1500    1350  1800
110    105  115    1200    1000  1500
120    115  125     800     750  1000
130    125  135     400     200   450
140    135  145     200     180   230
150    145  155     100      95   125
160    155  165      60      30    70
170    165  175      40      10    60
180    175  185      20      10    25
190    185  195      10       8    15
200    195  205       5       3    10

```

This selection menu for Example 12 is shown in Fig. 18. The screen output for this example upon selecting spline fit (B) is shown in Fig. 19. The screen output for this example upon selecting log-log (L) plot is shown in Fig. 20.

FASTPLOT EXAMPLE 12 - ERROR BARS

Select type of plot:

- C - Linear Curve
- B - Spline Fit
- X - Y vs log₁₀ X
- Y - log₁₀ Y vs X
- L - log₁₀ Y vs log₁₀ X
- Q - Quit and return to menu

Enter type (or Quit): B

Fig. 18. Selection menu for Example 12.

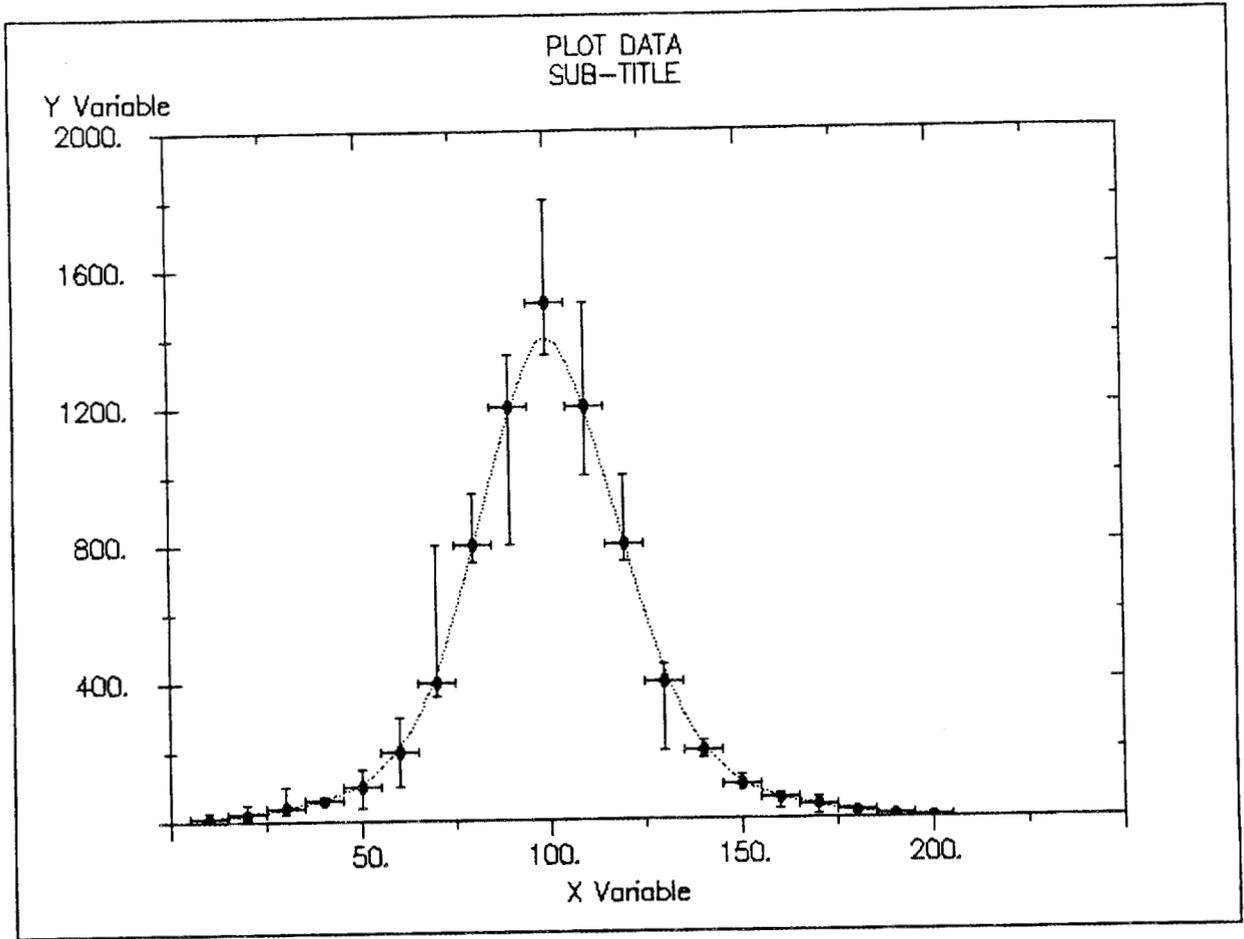


Fig. 19. Screen output for Example 12 upon selecting linear curve (C).

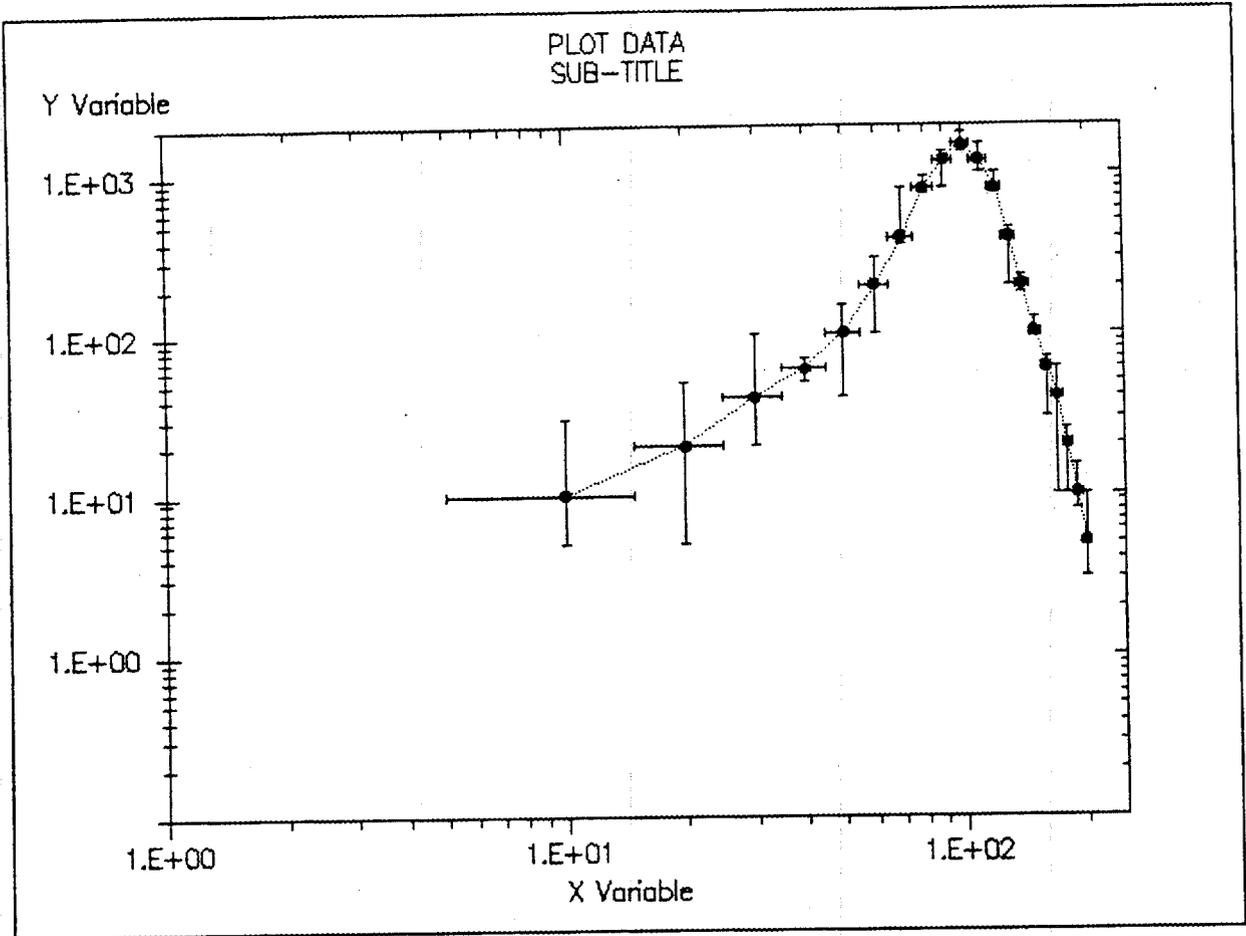


Fig. 20. Screen output for Example 12 upon selecting log-log (L) plot.

Two additional routines (IMAGE and FRACTAL) have been added to FPDEMO to demonstrate FASTPLOT color imaging capabilities. The first example (IMAGE) displays a binary image file (CORK.TOM) using medium resolution graphics (320 × 200) and the 256-color palette. The image file is specified in the input file IMAGE.DAT. The second example (FRACTAL) displays the Sierpinski Gasket using high resolution (VCA) graphics and the default 16-color palette. The input file for the second example is FRACTAL.DAT.

```

SUBROUTINE IMAGE
c-----*
c This example displays a computer tomography image using      *
c medium resolution graphics (320x200) and 256 colors.         *
c It uses the FP_COLOR.256 color table supplied with FASTPLOT *
c The image is from From "A First Course in Computational Physics" *
c by Paul A. DeVries (1994) John Wiley and Sons               *
c-----*
      character*40      filename
      character*1       key(4)
      integer*1         HEAD(8), IPIX
      integer*4         xpixel, ypixel, indx
      integer*4         width, height, COLOR
c
c initialize keyboard interrupt keys
c
      key(1)='q'
      key(2)='Q'
      key(3)=char(13)
      key(4)=char(27)
c
c get filename of image, control parameters
      open(unit=19,file='image.dat',status='old')
      read(19,*)filename
      read(19,*)COLOR
      read(19,*)xpixel
      read(19,*)ypixel
      close(19)
c
      open (unit=20,file=filename,status='unknown',form='binary')
c
c Read header from binary data file
c
      read(20) (HEAD(i),i=1,8)

c Set up FASTPLOT graphics

c Medium resolution (320x200) graphics with 256 colors
      call BGNPLOTM(0)
c

```

```

        call SCREEN(width, height)
        call PORT(0,0,width,height)
        call WINDOW(0.,1000.,1000.,0.)
        call CLEAR
c
c      0 - DEFAULT
c      1 - greyscale (16)
c      2 - user defined (16)
c      3 - grey scale (256)
c      4 - user defined (256)
        if (COLOR.eq.1) call GREYSCALE (0)
        if (COLOR.eq.2) call DEFINEMAP (0)
        if (COLOR.eq.3) call GREYSCALE (1)
        if (COLOR.eq.4) call DEFINEMAP (1)
c
c STEP 5: Display IMAGE
c
        iy = ypixel
        do j = iy, 1, -1
            do i = 1, xpixel
c
c get pixel value from binary file
c
                read (20) IPIX
                indx = IPIX
c
c Draw image using FASTPLOT call PIXEL
c
                call pixel(indx, i-1, iy - j)
            enddo
        enddo
        call UNREG_FONTS
        call ENDPLOT(.TRUE.,4,KEY)
        close(20)
END

```

IMAGE.DAT - Input data for IMAGE Example.

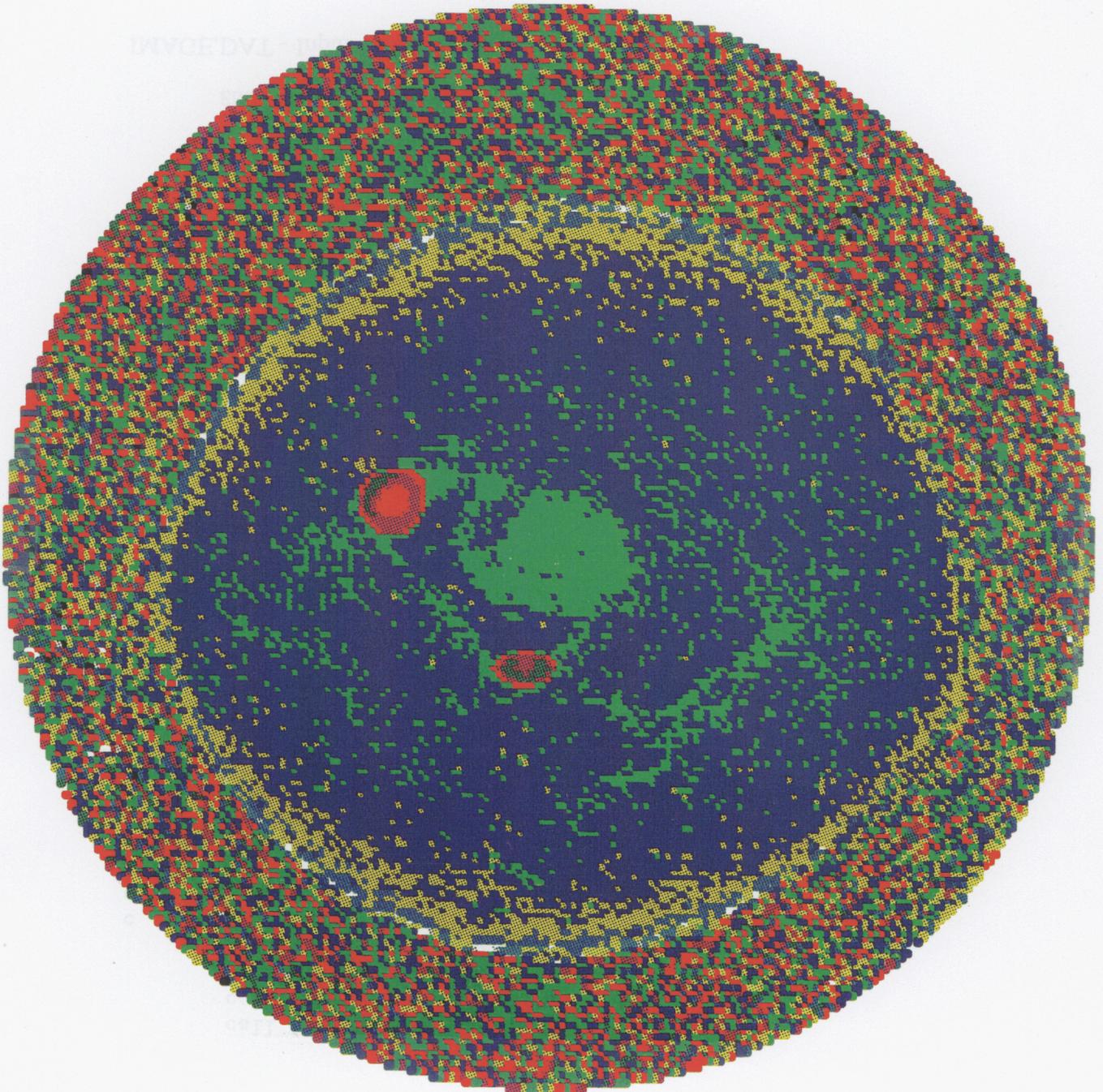
```

'cork.tom'  filename      *** FASTPLOT IMAGE EXAMPLE DATA ***
4                                                    c o l o r      m a p
(0-default,1-GS(16),2-USER(16),3-GS(256),4-USER(256)]
320          number of xpixels in image
200          number of ypixels in image

```

The screen output for this example is shown in Fig. 21.

Fig. 21. X-ray tomography reconstruction of a 1-cm cork containing three objects: a 2-mm round wooden shaft, a 1-mm steel shaft (paper clip), and a 0.5 mm by 1-mm rectangular steel saw shaft. This image can be displayed with the routine IMAGE using a 256-color map such as the thermal map (Appendix E). To create this color print of the image, an HPGL file was converted to a color postscript file using the PRINTGL program. The postscript file was printed on a Tektronix Phaser 200e wax-transfer color printer. To enhance the color contrast between the cork (blue), the wooden shaft (green), and the metal objects (red), a banded 16-color map was used for the color print.



SUBROUTINE FRACTAL

```

c-----*
c Creates Sierpinski Gasket using iterated function system method *
c developed by Richard C. Ward, Oak Ridge National Laboratory *
c date: 5/5/92, modified 12-30-93 *
c This FASTPLOT example uses call to routine PIXEL *
c-----*
c
      character*200 string,NAME
      character*40 ifsfile
      character*1 key(2)
      integer*1 s(300,300), t(300,300)
      real*4 a(10),b(10),c(10),d(10),e(10),f(10)
      real*4 xwin,ywin
c
      key(1) = char(13)
      key(2) = char(27)

c register and set font
      call REG_FONTS
      call SET_FONT(4,15,8)

c get IFS file
      ifsfile='fractal.dat'
      open(unit=10,file=ifsfile,status='old')
      read(10,*)icolor
      read(10,*)iback
      read(10,*)num
      read(10,*)initial
      read(10,*)n, NAME
c
      do i=1,n
        read(10,*)a(i),b(i),c(i),d(i),e(i),f(i)
      enddo
c
c initialize s array
      do i=1,num
        do j=1,num
          s(i,j)=0
        enddo
      enddo
c
      if (initial.eq.1) then
c initial configuration is a square
        do i=1,num
          s(i,num) = 1
          s(i,1) = 1
          s(1,i) = 1
          s(num,i) = 1
        enddo

```

```

endif

if (initial.eq.2)then
c initial configuration is a triangle
do 160 i=1,num
do 161 j = i,num
s(i,j) = 1
161 continue
160 continue
endif

c
call BGNPLOT (iback)
call SCREEN (IX, IY)
call PORT (0, 0, IX, IY)
xwin = num
ywin = num
call WINDOW (0., xwin, ywin, 0.)
call LINETYPE(0)

c
count = 0
1000 continue

c
count = count + 1
if (count.eq.8) THEN
string = 'Press <Enter> to Quit.'
call LABEL(15,0,0,1,0,240,string)
call UNREG FONTS
call ENDPLOT(.TRUE.,2,KEY)
RETURN
endif

string = NAME
call LABEL(15,0,0,1,0,20,string)
do i=1,num

c
do j=1,num

c
t(i,j)=s(i,j)
if ( t(i,j) .eq. 1 ) call pixel (icolor, i+180, j+70)
c zero out s array for next pass
s(i,j)=0

c
enddo

c
enddo

c
do i=1,num

c
do j=1,num

c
if (t(i,j) .eq. 1) then

```

```

do 40 k=1,n
  l = ABS(a(k)*i+b(k)*j+e(k))
  m = ABS(c(k)*i+d(k)*j+f(k))
  if (l .gt. 0 .and. m .gt. 0) s( l, m ) = 1
40  continue
endif
c
  enddo
c
  enddo
c
string = 'Press <Enter> to Continue.'
call LABEL(15,0,0,1,0,400,string)
call HALTCLR (.TRUE., 2, KEY)
c
goto 1000
c
END

```

FRACTAL.DAT - Input data for FRACTAL Example.

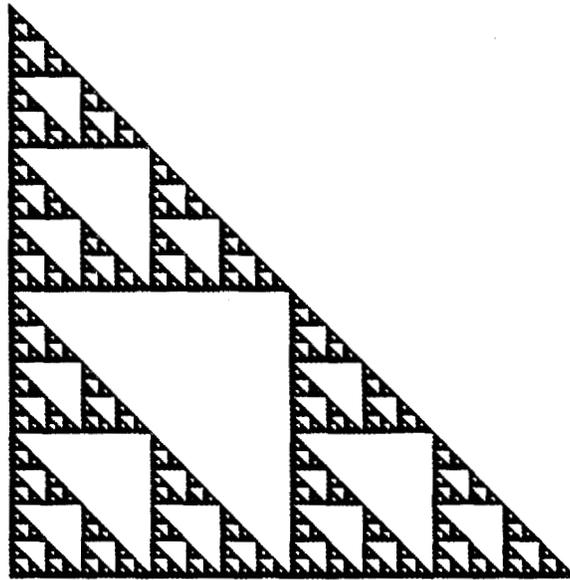
```

14  color          *** FASTPLOT FRACTAL EXAMPLE DATA ***
1   background color (blue)
300 size
2   initial condition (1-open square, 2-triangle)
3   'SIERPINSKI GASKET'
0.5 0 0 0.5 1 1
0.5 0 0 0.5 1 150
0.5 0 0 0.5 150 150

```

The screen output for this example is shown in Fig. 22.

SIERPINSKI GASKET



Press <Enter> to Continue.

Fig. 22. Screen output for FRACTAL example.

Finally, we list the FASTPLOT help facility accessible from FPDEMO. This listing demonstrates the use of a two-column menu and utilizes a routine (TEXT) to read and display a TEXT (help) screen.

SUBROUTINE HELP

```

C*****
C FASTPLOT HELP *
C*****
c
  EXTERNAL MENU
  character*80 string(20)
  integer*4 ROW(20), COL(20)
  integer*4 tcol(20), tbc col(20), hcol(20), hbc col(20)
  integer*4 icen(20), entry
  integer*4 uparrow, dnc arrow, ltarrow, rtarrow, accept, quit
  integer*4 mfirst, mborder, mtitle, mentry

c define menu
c
  do 10 istr = 1,11
    tbc col(istr) = 0
    hcol(istr) = 0
    hbc col(istr) = 0
    COL(istr) = 0
  c to center string, set icen(istr) = 1, COL(istr) is ignored
    icen(istr) = 1
  10 continue

c specify titles - mtitle is number of titles and help lines
  mtitle = 4

c specify menu rentires - mentry is number of menu entries
  mentry = 8

c specify maximum number of characters in menu entry
  mchars = 18

c number of spaces in border
  mborder = 3

c number of entries in each column for multicolumn menus
c or zero for single column menus
  mcolumn = 4

  tcol(1) = 15
  ROW(1) = 5
  string(1) = 'FASTPLOT HELP'

  tcol(2) = 14

```

```

ROW(2)      = 6
string(2)   = 'Ver. 1.0 Date: 12/30/93'

tcol(3)     = 14
ROW(3)      = 21
string(3)   = 'Use Arrow Keys To Make Selection'

tcol(4)     = 0
tbc(4)      = 15
ROW(4)      = 22
string(4)   = 'Press <Enter> To Execute, <Esc> To Exit'

do 20 i = 5, 12
c set text color
    tcol(i) = 12
    tbc(i) = 0
c set highlight color
    hcol(i) = 0
    hbc(i) = 15
c do not center text
    icen(i) = 0
c two column menu
    if (i.le.8) then
        ROW(i) = i+5
        COL(i) = 19
    else
        ROW(i) = i+1
        COL(i) = 42
    endif
20 continue

string(5) = 'Usage'
string(6) = 'Basic Definitions'
string(7) = 'Commands (A-M)'
string(8) = 'Commands (M-Z)'
string(9) = 'Color Imaging'
string(10) = 'Fonts'
string(11) = 'Symbols'
string(12) = 'Return to Top Menu'

c
c MENU actions - add 256 to scan code to get extended key codes
uparrow = 72+256
dnarrow = 80+256
rtarrow = 77+256
ltarrow = 75+256
c accept = <Enter>, quit = <Esc>
accept = 13
quit = 27

mfirst = 1
30 continue

```

```

c put up menu frame - two columns
  istr = 5
  call MFRAME( 14, 2, 0,
+ ROW(istr)-1, COL(istr) - mborder -1,
+ ROW(istr+3)+1, COL(istr+mentry-1)+mborder+mchars)
c put up menu entries
  entry = MENU(tcol, tbcot, hcol, hbcot, icen,
+ ROW,COL,
+ uparrow, dntarrow, ltarrow, rtarrow, accept, quit,
+ mfirst, mcolumn, mtitle, mentry, string)
c
c      call TCLEAR
      call CLEAR
      IF (entry.eq.0) then
c reveal cursor
      CALL cursor (2)
      call ENDTEXT
      return
      ENDIF

      SELECT CASE (entry)
      CASE (1)
        CALL TEXT ('FPHELP.1')
      CASE (2)
        CALL TEXT ('FPHELP.2')
      CASE (3)
        CALL TEXT ('FPHELP.3')
      CASE (4)
        CALL TEXT ('FPHELP.4')
      CASE (5)
        CALL TEXT ('FPHELP.5')
      CASE (6)
        CALL TEXT ('FPHELP.6')
      CASE (7)
        CALL TEXT ('FPHELP.7')
      CASE (8)
c reveal cursor
        CALL cursor (2)
        call ENDTEXT
        return
      END SELECT
      mfirst = entry
      GOTO 30
      RETURN
      END

      SUBROUTINE TEXT (helpfile)
        character*80 info(24),title
        character*8 helpfile
c
        open (UNIT=96,file=helpfile,status='old')

```

```

        read (UNIT=96,FMT=1) title
1      format(a80)
        read (UNIT=96,FMT=*) nlines
c
        do 10 i=1,nlines
          read (UNIT=96,FMT=1) info(i)
10     continue
c
c place help on screen
        call MFRAME(12,1,0,2,2,24,78)
        call TLABEL(0,15,1,1,4,0,title)
c
        do 20 i=1,nlines
          call TLABEL(14,0,0,0,4+i,4,info(i))
20     continue
c
        ikey = 0
30     continue
        call KEYINT(ikey)
c
        if (ikey.eq.13.or.ikey.eq.27)then
c          call TCLEAR
          call CLEAR
          close (96)
          return
        else
          goto 30
        endif
c
    end

```

An example help screen is shown in Fig. 23.

FASTPLOT SYMBOLS

To utilize FASTPLOT symbols, they must first be loaded into memory by calling SET_SYMBOL. The subroutine MARKER is used to select a particular symbol to use in a plot. The parameters specified are the symbol id and the symbol height (roughly in pixels).

id	SYMBOL	id	SYMBOL
0	open square	7	filled square
1	open circle	8	filled circle
2	open triangle	9	filled triangle
3	vertical cross (+)	10	filled diamond
4	lazy cross (x)	11	filled inverted triangle
5	open diamond	12	horizontal bar - (error bar)
6	open inverted triangle	13	vertical bar (error bar)

Call subroutine LINETYPE to specify the linetype prior to calling MARKER. To select another symbol, subroutine MARKER must be called again. The symbols are defined in the symbol table: FP_SYMBL.TAB. The user may specify the location of the symbol table in FP_FONT.LOC.

Fig. 23. Example of FPDEMO help screen (Symbols).

4. LIST OF FASTPLOT SUBROUTINE AND FUNCTION CALLS (February 1994)

SUBROUTINE AXIS (*icolor, itype*)
 SUBROUTINE BGNPLOT (*icolor*)
 SUBROUTINE BGNPLOTM (*icolor*)
 SUBROUTINE BGNTEXT (*icolor*)
 FUNCTION BSPLIN (*T, A, B, C, D*)
 SUBROUTINE CAPITAL (*ch*)
 SUBROUTINE CLEAR
 SUBROUTINE COLORMAP (*indx, ired, igreen, iblue*)
 SUBROUTINE CURSOR (*itype*)
 SUBROUTINE CURVE (*icolor, itype, fill, xstep, npoints, X, Y*)
 SUBROUTINE DEFINEMAP (*itype*)
 SUBROUTINE ENDPLOT (*beep, numkey, key*)
 SUBROUTINE ENDTEXT
 SUBROUTINE ERRBAR (*icolor, itype, ierr, npoints, X, Y, Xrange, Yrange*)
 SUBROUTINE FRAME (*icolor*)
 SUBROUTINE GET_DISPLAY (*display*)
 SUBROUTINE GREYSCALE (*itype*)
 SUBROUTINE GRID (*icolor, itype, step, minor*)
 SUBROUTINE HALTCLR (*beep, numkey, key*)
 SUBROUTINE KEYINT (*ikey*)
 SUBROUTINE LABEL (*icolor, ibackcol, iback, icenter, xloc, yloc, string*)
 SUBROUTINE LINETYPE (*itype*)
 SUBROUTINE MARKER (*id, height*)
 FUNCTION MENU (*tcol, tbc, hcol, hbcol, icen, ROW, COL, uparrow, dnarrow, larrow, rarrow, accept, quit, mfirst, mcolumn, mtitle, mentry, string*)
 SUBROUTINE MFRAME (*icolor, itype, icen, ROWTL, COLTL, ROWBR, COLBR*)
 SUBROUTINE NUMBERS (*icolor, itype, form, step, skip*)
 SUBROUTINE PIXEL (*icolor, ix, iy*)
 SUBROUTINE PORT (*iul_x, iul_y, ilr_x, ilr_y*)
 SUBROUTINE QUERY (*mcol, mbcol, ROW, COL, message, answer, mbeep, ecol, ebc, icen, ROWE, COLE, errmess, ebeep, ngood, good*)
 SUBROUTINE REG_FONTS
 INTEGER*4 FUNCTION RGB (*r, g, b*)
 SUBROUTINE SCREEN (*width, height*)
 SUBROUTINE SET_FONT (*ifont, ifonth, ifontw*)
 SUBROUTINE SET_SYMBOL
 SUBROUTINE SYMBOL (*icolor, itype, isym, npoints, X, Y*)
 SUBROUTINE TCLEAR
 SUBROUTINE TICKS (*icolor, itype, tpix_h, tpix_w, step, minor*)
 SUBROUTINE TLABEL (*icolor, ibackcol, iback, icenter, ROW, COL, string*)
 SUBROUTINE WINDOW (*ul_x, ul_y, lr_x, lr_y*)
 SUBROUTINE UNREG_FONTS

5. FASTPLOT LIBRARY REFERENCE

This section describes the use of the subroutines in the FASTPLOT library. The FORTRAN source for the FASTPLOT library and the assembler source for subroutine KEYINT are included in the diskette.

SUBROUTINE AXIS (*icolor, itype*)

This subroutine draws an axis line on the edge of the existing graphics window (virtual coordinates), as specified by parameter *itype*. The position of the axis is as follows:

	<i>itype</i>
X-axis along the bottom	10
Y-axis along the left side	20
X-axis along the top	30
Y-axis along the right side	40

The color of the axis is determined by the parameter *icolor*, which can range from 0 to 15. The default colors for the IBM PC are listed in Appendix D.

Type for variables passed to subroutine AXIS:

INTEGER*4 *icolor, itype*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor**, **moveto_w**, **lineto_w**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE BGNPLOT (*icolor*)

This subroutine sets the video mode to the maximum resolution allowed by the graphics adapter. The background color is set to *icolor*.

Type for variables passed to subroutine BGNPLOT:

INTEGER*4 *icolor*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setvideomode**

Uses FASTPLOT subroutines and/or functions: **GET_DISPLAY**

SUBROUTINE BGNPLOTM (*icolor*)

This subroutine sets the video mode to medium resolution (320 × 200) graphics. The background color is set to *icolor*.

Type for variables passed to subroutine BGNPLOTM:

INTEGER*4 *icolor*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setvideomode**

Uses FASTPLOT subroutines and/or functions: **GET_DISPLAY**

SUBROUTINE BGNTTEXT (*icolor*)

Used in TEXT mode, this subroutine saves the original foreground and background colors, clears the screen, and sets the background color to *icolor*. It serves a similar purpose as BGNPLOT for graphics mode.

Type for variables passed to subroutine BGNTTEXT:

INTEGER*4 *icolor*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **getttextcolor**, **getbkcolor**, **setbkcolor**, **clearscreen**

Uses FASTPLOT subroutines and/or functions: **NONE**

FUNCTION BSPLIN (*T, A, B, C, D*)

This function returns the B-spline fit given four consecutive values along the curve (*A,B,C,D*). The parameter *T* is the sub-interval. This function is used by the CURVE routine to plot a linear spline fit.

Type for variables passed to function BSPLIN:

REAL*4 *T, A, B, C, D*

Returns: REAL*4 *BSPLIN*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE CAPITAL (*ch*)

This subroutine converts lower-case character *ch* to upper case.

Type for variables passed to subroutine CAPITAL:

CHARACTER*1 *ch*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE CLEAR

This subroutine clears the screen. The graphics routine **clearscreen** is called with the parameter **\$GCLEARSCREEN**, which clears the entire graphics screen.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **clearscreen**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE COLORMAP (*indx, ired, igreen, iblue*)

This subroutine remaps the color palette using the Microsoft® FORTRAN graphics call **remappalette**. The value *indx* is assigned to the intensities *ired, igreen, iblue*. The *indx* varies from 0 to 15 for 16 color map and from 0 to 255 for the 256 color map. The intensity values vary from 0 to 63.

Type for variables passed to subroutine COLORMAP:

INTEGER*4 *indx, ired, igreen, iblue*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **remappalette**

Uses FASTPLOT subroutines and/or functions: **RGB**

SUBROUTINE CURSOR (*itype*)

This subroutine sets the cursor to type *itype* specified. If *itype* is 0, no cursor is drawn; if *itype* is 1 (2), the cursor is single (double) underline; if *itype* is 3, one obtains a block cursor.

Type for variables passed to subroutine CURSOR:

INTEGER*4 *itype*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **getvideoconfig**, **settextcursor**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE CURVE (*icolor, itype, fill, xstep, npoints, X, Y*)

Depending on curve type, *itype*, this subroutine plots a curve through the points by the arrays *X* and *Y*, specified in the existing graphics window (virtual coordinates). The number of points in the arrays is passed as the parameter *npoints*. If *itype* is 11, CURVE plots a line between each point of the arrays. If *itype* is 12, CURVE plots a line through the points given by *X* and $\log_{10} Y$. If *itype* is 21, CURVE plots a line through the points given by $\log_{10} X$ and *Y*. If *itype* is 22, CURVE plots a line through the points given by $\log_{10} X$ and $\log_{10} Y$. If *itype* is 31, CURVE plots a histogram versus linear *x*. Finally, if *itype* is 55, CURVE plots a B-spline through the points given by *X* and *Y*. The spline fit is restricted to 2000 points.

The type of curve is specified by *itype*. The following types are allowed:

<i>itype</i>		Curve type
11	linear	<i>X</i> - linear <i>Y</i>
12	linear	<i>X</i> - $\log_{10} Y$
21	\log_{10}	<i>X</i> - linear <i>Y</i>
22	\log_{10}	<i>X</i> - $\log_{10} Y$
31	histogram	<i>X</i> - linear <i>Y</i>
55	linear	<i>X</i> - linear <i>Y</i> (spline fit)

If *itype* is 31, the histogram spacing *xstep* must be provided. In addition, the user must specify *fill* as either **.TRUE.** for filled histogram or **.FALSE.** for unfilled histogram.

The color of the curve is determined by the parameter *icolor* which can range from 0 to 15. This subroutine plots a line between each point in the arrays no spline fit is made.

Type for variables passed to subroutine CURVE:

INTEGER*4 *icolor, itype, npoints*
REAL*4 *X(npoints), Y(npoints), xstep, xmin, ymin*
LOGICAL *fill*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor**,
moveto_w, **lineto_w**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE DEFINEMAP (*itype*)

This subroutine calls **COLORMAP** to remap the color palette using either a 16 color map (**FP_COLOR.16**) or a 256 color map (**FP_COLOR.256**). A value of *itype* = 0 specifies the 16 color map; a value of *itype* = 1 specifies the 256 color map. The routine looks first in the local directory for the color map files; it then assumes the files are in the directory specified by **FP_FONT.LOC**. If files are not found, an error message is printed.

Type for variables passed to subroutine **DEFINEMAP**:

INTEGER*4 *itype*

Uses Microsoft®FORTRAN graphics subroutines and/or functions: **NONE**

Uses FASTPLOT subroutines and/or functions: **COLORMAP**

SUBROUTINE ENDPLOT (*beep, numkey, key*)

This subroutine is called at the conclusion of the plot. This routine pauses and waits for the user to type one of the characters in the array *key*. A beep is heard if the parameter *beep* is set to **TRUE**, no beep if set to **FALSE**. The character array *key* is dimensioned to the number of keys *numkey*. Examples for the values of the variable *key* are: 'Q', 'q', the escape key [**CHAR(27)**] and the return key [**CHAR(13)**]. When one of the specified keys is pressed, the subroutine restores the default screen mode (text mode). This routine functions exactly like subroutine **HALTCLR** except that it restores the default screen mode rather than just clearing the screen.

Type for variables passed to subroutine **ENDPLOT**:

LOGICAL *beep*
CHARACTER*1 *key (numkey)*
INTEGER*4 *numkey*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setvideomode**

Uses FASTPLOT subroutines and/or functions: **KEYINT**

SUBROUTINE ENDTEXT

Used in TEXT mode, this subroutine is called at the conclusion of all text screens. This subroutine restores the original foreground and background colors saved using **BGNTEXT**.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setttextcolor**, **setbkcolor**, **clearscreen**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE ERRBAR (*icolor, itype, ierr, npoints, X, Y, Xrange, Yrange*)

Type for variables passed to subroutine ERRBAR:

```
INTEGER*4  icolor, itype, ierr, npoints
REAL*4     X(npoints), Y(npoints), xmin, ymin
REAL*4     Xrange(2,npoints), Yrange(2,npoints)
```

Depending on curve type *itype*, this subroutine plots an error bar at every *ierr* value of the *X* and *Y* arrays specified in the existing graphics window (virtual coordinates). The number of points in the arrays is passed as the parameter *npoints*. If *itype* is 11, ERRBAR plots an error bar at every *ierr* value of the arrays *X* and *Y*. If *itype* is 12, ERRBAR plots an error bar at every *ierr* value of the arrays *X* and $\log_{10}Y$. If *itype* is 21, ERRBAR plots an error bar at every *ierr* value of the arrays $\log_{10}X$ and *Y*. Finally, if *itype* is 22, ERRBAR plots an error bar at every *ierr* value of the arrays $\log_{10}X$ and $\log_{10}Y$.

X(1,npoints) and *X(2,npoints)* determines the minimum and maximum range for the *X* array. *Y(1,npoints)* and *Y(2,npoints)* determines the minimum and maximum range for the *Y* array.

The type of curve is specified by *itype*. The following types are presently allowed:

<i>itype</i>	Curve type		
11	linear	<i>X</i>	- linear <i>Y</i>
12	linear	<i>X</i>	- \log_{10} <i>Y</i>
21	\log_{10}	<i>X</i>	- linear <i>Y</i>
22	\log_{10}	<i>X</i>	- \log_{10} <i>Y</i>

The type of symbol plotted at the top/bottom and left/right of the error bar is determined by the previous call to subroutine MARKER. Although any symbol in the symbol table may be used with the error bars, two symbols have been added specifically for error bars: a horizontal bar (symbol id = 12) and a vertical bar (symbol id = 13). The user can select either symbol id = 12 or 13 since ERRBAR knows which symbol to place on which error bar. The color of the error bar is determined by the parameter *icolor* which can range from 0 to 15. Filled symbols used for error bars are noted in the symbol table (FP_SYMBL.TAB) and are filled using the graphics call `floodfill_w`. When using filled symbols on top/bottom and left/right of error bar, call the ERRBAR routine prior to the call to subroutine CURVE.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: SYMBOL, CURVE

SUBROUTINE FRAME (*icolor*)

This subroutine draws a frame around the existing graphics window (virtual coordinates) as specified by the last call to SUBROUTINE WINDOW. The color of the frame is determined by the parameter *icolor*, which can range from 0 to 15. The graphics routine `rectangle_w` is called with parameter `$GBORDER` to yield an unfilled rectangle for the plot.

Type for variables passed to subroutine FRAME:

INTEGER*4 *icolor*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: `setcolor`, `rectangle_w`

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE GET_DISPLAY (*display*)

This subroutine returns the display type. The return parameter *display* will be set to CGA (for CGA graphics adapter and display, or EGA graphics adapter and CGA display), EGA (for EGA graphics adapter and EGA display), VGA (for VGA graphics adapter and display) or TXT (if no graphics adapter detected). This routine can be called prior to subroutine BGNPLOT. It is generally used to determine the presence of a graphics adapter before executing code which will utilize the graphics capabilities. The routine returns EGA if a HERCULES graphics adapter is detected.

Type for variables passed to subroutine GET_DISPLAY:

CHARACTER*3 *display*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **getvideoconfig**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE GREYSCALE (*itype*)

This subroutine calls RGB to remap the color pallette, establishing either a 16 level greyscale (*itype* = 0) or a 256 level greyscale (*itype* = 1). The routine looks first in the local directory for the color map files; it then assumes the files are in the directory specified by FP_FONT.LOC. If files are not found, an error message is printed. Greys are obtained by having equal components of red, green, and blue. Since the maximum intensity for any one color is 63, while there are 256 grey scales for (*itype* = 1), each group of four levels (0-3, 4-7, 8-11, etc.) is set to the same grey scale.

Type for variables passed to subroutine GREYSCALE:

INTEGER*4 *itype*

Uses Microsoft®FORTRAN graphics subroutines and/or functions: **NONE**

Uses FASTPLOT subroutines and/or functions: **COLORMAP**

SUBROUTINE GRID (*icolor, itype, step, minor*)

This subroutine puts a grid on the plot as specified by the parameter *itype*. A vertical grid is drawn if *itype* is 1 and a horizontal grid if *itype* is 2.

The color of the grid is determined by the parameter *icolor*, which can range from 0 to 15. The grid spacing is determined by the parameter *step*. No minor grid is placed if *minor* is zero. For a linear plot, a grid is placed every *step/minor* if *minor* is positive. For a log plot, specify *minor* as -1 to obtain minor log grid. Call GRID prior to calling TICKS; otherwise, the tick marks will not show.

Type for variables passed to subroutine GRID:

INTEGER*4 *icolor, itype, minor*
REAL*4 *step*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor, moveto_w, lineto_w**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE HALTCLR (*beep, numkey, key*)

This subroutine is called when the user wishes to halt and clear the screen. The routine pauses and waits for the user to type one of the characters in the array *key*. A beep is heard if the parameter *beep* is set to TRUE, no beep if set to FALSE. The character array *key* is dimensioned to the number of keys *numkey*. Examples for the values of the array *key* are: 'Q', 'q', the escape key [CHAR(27)] and the return key [CHAR(13)]. When one of the specified keys is pressed, the subroutine clears the screen. This routine functions exactly like subroutine ENDPLOT except that it clears the screen rather than restoring the default screen mode.

Type for variables passed to subroutine HALTCLR:

LOGICAL	<i>beep</i>
CHARACTER*1	<i>key (numkey)</i>
INTEGER*4	<i>numkey</i>

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **clearscreen**

Uses FASTPLOT subroutines and/or functions: **KEYINT**

ASSEMBLER ROUTINE KEYINT (*key*)

This routine returns the ASCII value for the key pressed. If the key returns an extended key code, KEYINT returns the extended key code + 256. The extended key codes and corresponding FASTPLOT codes are shown in Appendix B. This routine is called by subroutine ENDPLOT and subroutine HALTCLR to test all keys that the user has selected to return the screen to default mode. Subroutine KEYINT can be called from the user's FORTRAN code as well.

Type for variables passed to subroutine KEYINT:

INTEGER*4	<i>key</i>
-----------	------------

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **NONE**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE LABEL (*icolor, ibackcolor, iback, icenter, xloc, yloc, string*)

This subroutine puts a label *string* at SCREEN COORDINATE location *xloc, yloc* in graphics mode. The color of the text is set by *icolor* which can range from 0 to 15. If *icenter* is 1, the string is centered; if *icenter* is 0, the string is placed at position *xloc*. If *iback* is 1, the background will be colored with the color *ibackcolor*. This allows labels to be printed on top of a plotted curve. The labels must be printed after the curve is drawn.

Type for variables passed to subroutine LABEL:

CHARACTER*200 *string*
INTEGER*4 *icolor, icenter, xloc, yloc, ibackcolor, iback*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **getvideoconfig, gettextextent, moveto, setcolor, outgtext**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE LINETYPE (*itype*)

This subroutine sets the line style depending on *itype*:

<i>itype</i>	linetype	
0	SOLID	(calls setlinestyle with parameter #FFFF)
1	DASH	(calls setlinestyle with parameter #CCCC)
2	DOT	(calls setlinestyle with parameter #5555)

Type for variables passed to subroutine LINETYPE:

INTEGER*4 *itype*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setlinestyle**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE MARKER (*id, height*)

This subroutine is used to select a particular symbol to use in a plot. The parameters specified are the symbol *id* (see the symbol table, file FP_SYMBL.TAB, further on) and the symbol height (roughly in pixels). To select another symbol, subroutine MARKER must be called again. The default symbols available with FASTPLOT are the following:

<i>id</i>	<u>SYMBOL</u>
0	open square
1	open circle
2	open triangle
3	vertical cross (+)
4	lazy cross (x)
5	open diamond
6	open inverted triangle
7	filled square
8	filled circle
9	filled triangle
10	filled diamond
11	filled inverted triangle
12	horizontal bar (error bar)
13	vertical bar (error bar)

The user can modify these symbols or add to the symbol table by editing the symbol table (file FP_SYMBL.TAB). See APPENDIX 1 for a listing of the FASTPLOT symbol table.

Type for variables passed to subroutine MARKER:

```
INTEGER*4 id
REAL*4    height
```

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: NONE

INTEGER*4 FUNCTION MENU (*tcol, bcol, hcol, hbcoll, icen, ROW, COL, uparrow, dnarrow, ltarrow, rtarrow, accept, quit, mfirst, mcolumn, mtitle, mentry, string*)

This function is used to construct a text menu. The number of titles or help lines for the menu is *mtitle*, and the number of entries in the menu is *mentry*. The number of text spaces between the left and right sides of the menu frame and the menu entries is *mborder*. The menu item to be highlighted on first entering menu is *mfirst*. The number of entries in a menu column if more than one column is *mcolumn*. For a single column menu, *mcolumn* should be set to zero.

A number of variables are passed as arrays. The first *mtitle* elements of each array are for the title lines and/or help lines, and the last *mentry* elements of each array are for the menu entries.

The foreground color of text strings is *tcol*; the background color of text strings is *bcol*. The text highlight color of strings is *hcol*; the text-highlight background color of strings is *hbcol*. The text is placed at the row and column *ROW*, *COL*, with the upper-left corner of the first menu item placed at the text position *ROW*, *COL* element specified for the menu entries. The text strings passed are in the array *string*. The text strings will be centered if *icen* is specified as 1, or placed at the *COL* position if *icen* is specified as 0.

Finally, the menu action must be passed through the MENU function call. The variables *uparrow*, *dnarrow*, *larrow*, and *rarrow* are passed to the KEYINT routine. The values are determined by adding 256 to the ASCII extended key codes for the cursor up, cursor down, cursor left, and cursor right keys. The ASCII integer value for the <Enter> key, for example, is passed as *accept* and the ASCII integer value for the <Esc> key is passed as *quit*. The user could pass any key characters for these six action parameters.

The user should call TCLEAR prior to executing the MENU function. If a menu frame is desired, the user must call MFRAME prior to executing the MENU function, placing the top-left and bottom-right corners of the menu frame just outside the menu entries.

The function MENU returns a number corresponding to the menu entry selected or 0 if *quit* key is pressed.

Type for variables passed to function MENU:

INTEGER*4 *mfirst*, *mcolumn*, *mtitle*, *mentry*
INTEGER*4 *uparrow*, *dnarrow*, *larrow*, *rarrow*, *accept*, *quit*

ARRAYS of size *mtitle* + *mentry*:

INTEGER*4 *tcol*, *bcol*, *hcol*, *hbcol*, *icen*, *ROW*, *COL*, *string*

Returns: INTEGER*4 *MENU*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: MFRAME, TLABEL, KEYINT

SUBROUTINE MFRAME (*icolor*, *itype*, *icen*, *ROWTL*, *COLTL*, *ROWBR*, *COLBR*)

This subroutine draws a frame around text, such as entries in a menu. The row and column of the top-left corner is specified by (*ROWTL*, *COLTL*); the row and column of the bottom-right corner is specified by (*ROWBR*, *COLBR*). The color of the frame is determined by the parameter *icolor* which can range from 0 to 15. The type of frame (single - 1) or (double - 2) is specified by *itype*. The frame will be centered horizontally if *icen* is 1, in which case *COLTL* and *COLBR* are ignored. If *icen* is 0, the frame is placed using the top-left and bottom-right (*ROW*, *COL*) values.

Type for variables passed to subroutine MFRAME:

INTEGER*4 *icolor, itype, icen, ROWTL, COLTL, ROWBR, COLBR*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setttextcolor, setttextposition, outtext**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE NUMBERS (*icolor, itype, form, step, skip*)

This subroutine puts numbers on the axis specified by the parameter *itype*. The numbers are placed on the plot, depending on the value of the parameter *itype*, as follows:

	<i>itype</i>
linear X-axis	11
log X-axis	12
linear Y-axis	21
log Y-axis	22

The numbers are placed along the axis at the interval specified by the parameter *step*. The color of the numbers is determined by the parameter *icolor*, which can range from 0 to 15. The parameter *form* is the format for the numbers (e.g., *form* = 'F5.1'). If the parameter *skip* is set to .TRUE., the first number on the axis is not displayed.

To place the numbers on the graph, the viewport SCREEN COORDINATES are shifted by the parameters BOTTOM, LEFT, TOP, RIGHT computed internally using values passed from routines SET_FONTS and TICKS. The bottom of the viewport is lowered by the amount BOTTOM, and the top is raised by the amount TOP. The left side of the viewport is moved further left by amount LEFT, and the right side of the viewport is moved further right by the amount RIGHT.

Type for variables passed to subroutine NUMBERS:

INTEGER*4 *icolor, itype*
LOGICAL *skip*
CHARACTER*6 *form*
REAL*4 *step*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor, setviewport, moveto_w, outtext**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE PIXEL (*icolor, ix, iy*)

This subroutine draws a pixel of color *icolor* at SCREEN COORDINATES (*ix, iy*). The color of the pixel can range from 0 to 15 for the 16-color map or 0 to 255 for the 256-color map.

Type for variables passed to SUBROUTINE PIXEL:

INTEGER*4 *icolor, ix, iy*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor, setpixel.**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE PORT (*iul_x, iul_y, ilr_x, ilr_y*)

This subroutine sets the viewport (screen coordinates). The upper-left corner of the viewport is *iul_x, iul_y* and the lower-right corner of the viewport is *ilr_x, ilr_y*. This is the convention is the same as the one used in the Microsoft® FORTRAN graphics subroutine **setviewport** called by this routine.

Type for variables passed to subroutine PORT:

INTEGER*4 *iul_x, iul_y, ilr_x, ilr_y*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setviewport**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE QUERY (*mcol, mbc, ROW, COL, message, answer, mbeep, ecol, ebc, icen, ROWE, COLE, errmess, ebeep, ngood, good*)

This subroutine is called to query the user. The routine passes the query as *message* and, a default answer as *answer*. The color of these text strings is *mcol* and the background color is *mbc*. The *message* string is placed at position (*ROW, COL*) and the *answer* string is placed one character width from the end of the *message* string. If parameter *mbeep* is TRUE, a beep is heard when the message is written. The user presses <Enter> to accept the default

answer or types in another answer and presses <Enter>. The user's answer can be compared to a set of acceptable *good* answers. The number of acceptable answers is *ngood* and the answers, in character form, are the elements of the array *good*. If the user's answer does not correspond to any of the acceptable answers, an error message *errmess* is printed at position (*ROWE,COLE*) if *icen* is 0. The error message is centered in row *ROWE* if *icen* is 1. The color of the error message is *ecol*, and the background color is *ebcol*. If parameter *ebeep* is TRUE, a beep is heard when the error message is written.

If the query pertains to a filename, the value of *ngood* must be negative (-1,-2,-3,or-4). In each case *answer* is concatenated with the first element of *good*, which should be either blank or an extension (e.g., '.DAT') to produce a filename. If *ngood* is -1, an error is indicated if the file does not exist. If *ngood* = -2, an error is indicated if the file does exist, the user is asked for a new filename. If *ngood* = -3, an error is indicated if the file does exist, control is returned to main program with *answer* equal to the filename and *ngood* = -999. The main program can then call QUERY again with message **Overwrite? (y/[n])**: to overwrite the existing file. The existence of the file is tested using an OPEN statement. If *ngood* is -4, the existence of the file is not checked.

If *ngood* is zero or positive, subroutine QUERY returns the user's response as *answer*; if *ngood* is negative, QUERY returns the full filename as *answer*.

Type for variables passed to subroutine QUERY:

INTEGER*4	<i>mcol,mbcol,ROW,COL,ecol,ebcol,icen,ROWE,COLE,ngood</i>
CHARACTER*80	<i>message, answer, errmess</i>
CHARACTER*40	<i>good (ngood)</i>
LOGICAL	<i>mbeep, ebeep</i>

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setttextcolor**, **setbkcolor**, **setttextposition**, **outtext**

Uses FASTPLOT subroutines and/or functions: TLABEL

SUBROUTINE REG_FONTS

This subroutine registers (initializes) the fonts used by the graphics routines. Microsoft® FORTRAN Versions 5.0 and 5.1 provide six typefaces to use with graphics: Courier, Helv, Tmns Roman, Modern, Script and Roman. The font typeface files all have the extension FON. The subroutine assumes they are in the current directory unless the user supplies the location (path) in the file FP_FONT.LOC. The first three (Courier, Helv, Tmns Roman) are bit-map fonts and the last three (Modern, Script and Roman) are scalable fonts. The sizes (roughly in pixels) available for the bit-mapped fonts are: Courier (10 × 8, 12 × 9, 15 × 12), Helv (10 × 5, 12 × 7, 15 × 8, 18 × 9, 22 × 12, 28 × 16), Tmns Roman (10 × 5, 12 × 6, 15 × 8, 16 × 9, 20 × 12, 28 × 16). The fonts are specified using the SET_FONT subroutine (see below) which will select the best fit if the height and width specified are not exactly those shown above for bit-mapped fonts.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **registerfonts**

Uses FASTPLOT subroutines and/or functions: **NONE**

INTEGER*4 FUNCTION RGB (*r, g, b*)

This function mixes the red (*r*), green (*g*), and blue (*b*) components to produce the correct *RGB* value. The intensities can vary from 0 - 63. This routine is called by the routines *COLORMAP* and *GREYSCALE* to compute the desired color map. Returns mixed color value, *RGB*, which is a 32-bit integer of the form: 00000000 00BBBBBB 00GGGGGG 00RRRRRR, where *r*, *G*, and *B* represent bit values for red, green, and blue intensities. For example, the *RGB* value corresponding to pink is: 00000000 00100000 00100000 00111111. The function is obtained from ref. 1 (page 300).

Type for variables passed to subroutine *GREYSCALE*.

INTEGER*4 *r, g, b*

Returns: INTEGER*4 *RGB*

Uses Microsoft® FORTRAN graphics subroutine and/or functions: **NONE**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE SCREEN (*width, height*)

This subroutine returns the pixel width and height, in *SCREEN COORDINATES*, of the screen.

Type for variables passed to subroutine *SCREEN*:

INTEGER*4 *width, height*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **getvideoconfig**

Uses FASTPLOT subroutines and/or functions: **NONE**

SUBROUTINE SET_FONT (*ifont, ifonth, ifontw*)

This subroutine selects the desired font *ifont* and specifies the font height *ifonth* and font width *ifontw*. The fonts available for a given typeface are described under SUBROUTINE REG_FONTS. The SET_FONT subroutine will select the best fit if the height and width specified are not exactly those shown for bit-mapped fonts.

Type for variables passed to subroutine SET_FONT:

INTEGER*4 *ifont, ifonth, ifontw*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **options, outtext, setfont**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE SET_SYMBOL

This subroutine loads the symbol table (file FP_SYMBL.TAB) into memory. This subroutine must be called before subroutine MARKER, which selects symbol, is called. The subroutine assumes that the symbol table is in the current directory unless the user supplies the location (path) in the file FP_FONT.LOC.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: NONE

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE SYMBOL (*icolor, itype, isym, npoints, X, Y*)

Depending on curve type, *itype*, this subroutine plots a symbol at every *isym* value of the points given by the *X* and *Y* arrays, specified in the existing graphics window (virtual coordinates). The number of points in the arrays is passed as the parameter *npoints*. If *itype* is 11, SYMBOL plots a symbol at every *isym* value of the arrays *X* and *Y*. If *itype* is 12, SYMBOL plots a symbol at every *isym* value of the arrays *X* and $\log_{10}Y$. If *itype* is 21, SYMBOL plots a symbol at every *isym* value of the arrays $\log_{10}X$ and *Y*. Finally, if *itype* is 22, SYMBOL plots a symbol at every *isym* value of the arrays $\log_{10}X$ and $\log_{10}Y$.

The type of curve is specified by *itype*. The following types are presently allowed:

<i>itype</i>	Curve type	
11	linear <i>X</i>	linear <i>Y</i>
12	linear <i>X</i>	\log_{10} <i>Y</i>
21	\log_{10} <i>X</i>	linear <i>Y</i>
22	\log_{10} <i>X</i>	\log_{10} <i>Y</i>

The type of symbol is determined by the previous call to subroutine **MARKER**. The color of the symbol is determined by the parameter *icolor*, which can range from 0 to 15. Filled symbols are noted in the symbol table (**FP_SYMBL.TAB**) and are filled using the graphics call **floodfill_w**. When using filled symbols on top of a curve, call **SYMBOL** prior to the call to subroutine **CURVE**.

Type for variables passed to subroutine **SYMBOL**:

```
INTEGER*4  icolor, itype, isym , npoints
REAL*4     X(npoints), Y(npoints), xmin, ymin
```

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setcolor**, **moveto_w**, **lineto_w**, **floodfill_w**

Uses **FASTPLOT** subroutines and/or functions: **NONE**

SUBROUTINE **TCLEAR**

This subroutine clears the text screen by sending extended screen control characters to the **ANSI.SYS** driver. The driver file **ANSI.SYS** must be in **CONFIG.SYS**.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **NONE**

Uses **FASTPLOT** subroutines and/or functions: **NONE**

SUBROUTINE **TICKS** (*icolor, itype, tpix_h, tpix_w, step, minor*)

This subroutine puts tick marks on the axis specified by the parameter *itype*. The tick marks are placed, depending on the value of *itype* as follows:

	<i>itype</i>
X-axis, bottom of graph	
Inside axis	10
Outside axis	11
Both sides of axis	12
Y-axis, left side of graph	
Inside axis	20
Outside axis	21
Both sides of axis	22
X-axis, top of graph	
Inside axis	30
Outside axis	31
Both sides of axis	32
Y-axis, right side of graph	
Inside axis	40
Outside axis	41
Both sides of axis	42

The color of the tick marks is determined by the parameter *icolor*, which can range from 0 to 15. The placing of the tick marks is determined by the parameter *step*. The height (roughly in pixels) of the X-axis tick marks is determined by the parameter *tpix_h*. The width (roughly in pixels) of the Y-axis tick marks is determined by the parameter *tpix_w*.

No minor tick marks are placed if *minor* is zero. For a linear plot, minor tick marks are placed every *step/minor*. For a log plot, specify *minor* as -1 to obtain minor tick marks.

Type for variables passed to subroutine TICKS:

```

INTEGER*4  icolor, itype, tpix_h, tpix_w, minor
REAL*4     step

```

Uses Microsoft® FORTRAN graphics subroutines and/or functions: *setcolor*, *setviewport*, *moveto_w*, *lineto_w*

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE TLABEL (*icolor, ibackcol, iback, icenter, ROW, COL, string*)

This subroutine puts a label (text string) at location *ROW, COL* in text mode. The color of the text is set by *icolor*, which can range from 0 to 15. If *icenter* is 1, the string is centered; if *icenter* is 0, the value *COL* is used for horizontal placement. If *iback* is 1, the background will be colored with the color *ibackcol*. This routine is called by function MENU.

Type for variables passed to subroutine TLABEL:

CHARACTER*80 *string*
INTEGER*4 *icolor, icenter, ROW, COL, ibackcol, iback*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setttextcolor, setbkcolor, setttextposition, setttextcolor, outtext**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE WINDOW (*ul_x, ul_y, lr_x, lr_y*)

This subroutine sets the graphics window (virtual coordinates). The upper-left corner of the graphics window is (*ul_x, ul_y*) and (*lr_x, lr_y*) is the lower-right corner of the graphics window. This convention is the same as the one used in the Microsoft® FORTRAN graphics subroutine **setwindow** called by this routine. The parameter *finvert* passed to the graphics function **setwindow** is set to TRUE so that the y-axis increases from the screen bottom to the screen top.

Type for variables passed to subroutine WINDOW:

REAL*4 *ul_x, ul_y, lr_x, lr_y*

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **setwindow**

Uses FASTPLOT subroutines and/or functions: NONE

SUBROUTINE UNREG_FONTS

This subroutine frees memory allocated to graphics fonts.

Uses Microsoft® FORTRAN graphics subroutines and/or functions: **unregisterfonts**

Uses FASTPLOT subroutines and/or functions: NONE

APPENDIX A

SYMBOL TABLE FOR FASTPLOT

The FASTPLOT library routine SYMBOL can be used to plot symbols. Before the routine is called, the routine SET_SYMBOL must be called to load the symbol table file FP_SYMBL.TAB into memory, and the routine MARKER must be called to select the symbol to be used. There are 14 symbols in the symbol table:

<u>ID</u>	<u>SYMBOL</u>	<u>NUMBER of NODES</u>	<u>FILL</u>
0	Open square	5	.FALSE.
1	Open circle	9	.FALSE.
2	Open triangle	4	.FALSE.
3	Vertical cross (+)	5	.FALSE.
4	Lazy cross (x)	5	.FALSE.
5	Open diamond	5	.FALSE.
6	Open inverted triangle	4	.FALSE.
7	Filled square	5	.TRUE.
8	Filled circle	9	.TRUE.
9	Filled triangle	4	.TRUE.
10	Filled diamond	5	.TRUE.
11	Filled inverted triangle	4	.TRUE.
12	Horizontal bar (error bar)	2	.FALSE.
13	Vertical bar (error bar)	2	.FALSE.

The FASTPLOT symbol table file, FP_SYMBL.TAB, is shown below. The user can add five additional symbols to this table or modify the existing symbols.

```

-----
SYMBOLS FOR FASTPLOT LIBRARY - 03/26/93 (From Mack Patterson, CAD, ORNL)
14  NUMBER OF SYMBOLS IN SYMBOL TABLE
0   SYMBOL - SQUARE
.FALSE.  FILL
5   NUMBER OF COORDINATES
-2 -2
 2 -2
 2  2
-2  2
-2 -2
1   SYMBOL - CIRCLE (OCTAGON)
.FALSE.  FILL
9   NUMBER OF COORDINATES
 1  2
-1  2
-2  1
-2 -1
-1 -2
 1 -2
 2 -1

```

```

    2 1
    1 2
2     SYMBOL - TRIANGLE
.FALSE.  FILL
4     NUMBER OF COORDINATES
    -2 -1
     2 -1
     0 2
    -2 -1
3     SYMBOL - VERTICAL CROSS
.FALSE.  FILL
5     NUMBER OF COORDINATES
    -2 0
     2 0
     0 0
     0 2
     0 -2
4     SYMBOL - LAZY CROSS
.FALSE.  FILL
5     NUMBER OF COORDINATES
    -2 -2
     2 2
     0 0
    -2 2
     2 -2
5     SYMBOL - DIAMOND
.FALSE.  FILL
5     NUMBER OF COORDINATES
     2 0
     0 -2
    -2 0
     0 2
     2 0
6     SYMBOL - UP-SIDE DOWN TRIANGLE
.FALSE.  FILL
4     NUMBER OF COORDINATES
     2 1
    -2 1
     0 -2
     2 1
7     SYMBOL - FILLED SQUARE
.TRUE.   FILL
5     NUMBER OF COORDINATES
    -2 -2
     2 -2
     2 2
    -2 2
    -2 -2
8     SYMBOL - FILLED CIRCLE (OCTAGON)
.TRUE.   FILL
9     NUMBER OF COORDINATES

```

```

1  2
-1 2
-2 1
-2 -1
-1 -2
1 -2
2 -1
2 1
1 2
9  SYMBOL - FILLED TRIANGLE
.TRUE.  FILL
4  NUMBER OF COORDINATES
-2 -1
2 -1
0 2
-2 -1
10 SYMBOL - FILLED DIAMOND
.TRUE.  FILL
5  NUMBER OF COORDINATES
2 0
0 -2
-2 0
0 2
2 0
11 SYMBOL - FILLED UP-SIDE DOWN TRIANGLE
.TRUE.  FILL
4  NUMBER OF COORDINATES
2 1
-2 1
0 -2
2 1
12 SYMBOL - VERBAR (Use for top/bottom of vertical error bar)
.FALSE. FILL
2  NUMBER OF COORDINATES
-2 0
2 0
13 SYMBOL - HORBAR (Use for left/right of horizontal error bar)
.FALSE. FILL
2  NUMBER OF COORDINATES
0 -2
0 2

```


APPENDIX B

FASTPLOT EXTENDED KEY CODES

The extended key codes (arrow keys, page-up, page-down, home, end, function keys, etc.) are returned by KEYINT as the extended key code + 256. The extended key codes and the corresponding code used by FASTPLOT are shown below. Thus to set up an action using the up-arrow key, one must loop indefinitely until KEYINT returns 72+256 or 328.

Key	Extended Key Code	FASTPLOT code
Shift Tab	15	271
Alt (Q,W,E,R,T,Y,U,I,O,P)	16-25	272-281
Alt (A,S,D,F,G,H,J,K,L)	30-38	286-294
Alt (Z,X,C,V,B,N,M)	44-50	300-306
F1-F10	59-68	315-324
Home	71	327
Cursor Up	72	328
PgUp	73	329
Cursor Left	75	331
Cursor Right	77	333
End	79	335
Cursor Down	80	336
PgDn	81	337
Ins	82	338
Del	83	339
Shift F1-F10	84-93	340-349
Ctrl F1-F10	94-103	350-359
Alt F1-F10	104-113	360-369
Ctrl PtrSc	114	370
Ctrl Left Cursor	115	371
Ctrl Right Cursor	116	372
Ctrl End	117	373
Ctrl PgDn	118	374
Ctrl Home	119	375
Ctrl Alt (1,2,3,...,0,-,-)	120-131	380-387
Ctrl PgUp	132	388

APPENDIX C

FASTPLOT RESERVED UNITS, COMMON BLOCKS, AND VARIABLE NAMES

All reserved COMMON block and variable names have the prefix *FP_*. This list gives the reserved I/O unit numbers, COMMON blocks, and variable names.

Reserved I/O unit numbers: 96, 97, 98 and 99

COMMON BLOCK NAMES: VARIABLE NAMES:

FP_INFO *FP_DISPY, FP_ASPCT*

FP_DISPY

Variable type: CHARACTER*3

Description: Display type (TXT, CGA, EGA, VGA)

FP_ASPCT

Variable type: REAL*4

Description: Aspect ratio correction relative to VGA
(VGA 1:1, EGA 350:480, CGA 200:480)

FP_PORT *FP_ULX, FP_ULY, FP_LRX, FP_LRY, FP_PORTX, FP_PORTY*

FP_ULX, FP_ULY

Variable type: INTEGER*2

Description: x, y position (in pixels) of upper-left corner of viewport

FP_LRX, FP_LRY

Variable type: INTEGER*2

Description: x, y position (in pixels) of lower-right corner of
viewport

FP_PORTX, FP_PORTY

Variable type: INTEGER*2

Description: width and height of viewport

FP_WIND *FP_WULX, FP_WULY, FP_WLRX, FP_WLRY, FP_WINDX, FP_WINDY*

FP_WULX, FP_WULY

Variable type: REAL*4

Description: x, y position of upper-left corner of graphics window

FP_WLRX, FP_WLRY

Variable type: REAL*4

Description: x, y position of lower-right corner of graphics window

FP_WINDX, FP_WINDY
Variable type: REAL*4
Description: width and height of graphics window

FP_SYMB *FP_NUMB, FP_SYMN, FP_SYMBX, FP_SYMBY*

FP_NUMB
Variable type: INTEGER*4
Description: number of symbols in symbol table

P_SYMN(id)
Variable type: INTEGER*4
Description: number of nodes for symbol *id*

FP_SYMBX(id,j), FP_SYMBY(id,j)
Variable type: INTEGER*4
Description: x, y position of *j*th node of symbol *id*

FP_SFILL *FP_SYMFL(id)*
Variable type: LOGICAL
Description: Symbol *id* is filled if true, unfilled if false

FP_MARK *FP_MARKN, FP_MARKX, FP_MARKY, FP_MARKFL*

FP_MARKN
Variable type: INTEGER*4
Description: Number of nodes for selected symbol

FP_MARKX(j), FP_MARKY(j)
Variable type: REAL*4
Description: x, y screen position for *j*th node of selected symbol

FP_MARKFL
Variable type: LOGICAL
Description: Selected symbol is filled if true, unfilled if false

FP_MARK2 *FP_MARKID, FP_MARKH*

FP_MARKID
Variable type: INTEGER*4
Description: Symbol *id* for selected symbol

FP_MARKH
Variable type: REAL*4
Description: *height* of selected symbol

FP_FONTS *FP_FONTH, FP_FONTW*
Variable type: INTEGER*4
Description: Font height and width in pixels

FP_TICKS *FP_TICKH, FP_TICKW, FP_SIDE*

FP_TICKH, FP_TICKW
Variable type: INTEGER*4
Description: Tick height and width in pixels

FP_SIDE
Variable type: INTEGER*4
Description: -1, ticks inside, 0 ticks outside, 1 both

FP_TEXT *FP_OLDFGD, FP_OLDBGD*

FP_OLDFGD
Variable type: INTEGER*4
Description: Screen foreground color prior to running application

FP_OLDBGD
Variable type: INTEGER*2
Description: Screen background color prior to running application

FP_FRAME *FP_ROWTL, FP_COLTL, FP_ROWBR, FP_COLBR, FP_CHEK, FP_ERR*

FP_ROWTL, FP_COLTL
Variable type: INTEGER*4
Description: top-left corner of text menu frame

FP_ROWBR, FP_COLBR
Variable type: INTEGER*4
Description: bottom-right corner of text menu frame

FP_CHEK
Variable type: LOGICAL
Description: If true, menu frame is used. If false, no menu frame is used.

FP_ERR
Variable type: LOGICAL
Description: Error has occurred in previous call to QUERY, QUERY must remove error text.

APPENDIX D

DEFAULT COLORS FOR THE IBM PC

<u>icolor</u>	<u>Description</u>
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown (or dark yellow)
7	Light grey (or ordinary white)
8	Dark grey (black on many screens)
9	Light blue
10	Light green
11	Light cyan
12	Light red
13	Light magenta
14	Yellow (or light yellow)
15	Bright white

APPENDIX E

FASTPLOT COLOR MAPS

Each of these is a "thermal" color map, characteristic of a heated object (see H. Levkowitz and G. Herman, "Color Scales for Image Data," *IEEE Computer Graphics and Applications*, Vol. 12, No. 1, January 1992, pp. 72-80).

FP_COLOR.16

USER DEFINED COLOR MAP
16 COLORS

0	0	0	0
1	16	8	0
2	32	16	0
3	40	18	0
4	42	20	0
5	44	22	0
6	46	25	0
7	48	28	0
8	52	30	0
9	56	34	0
10	60	36	4
11	63	42	8
12	63	48	16
13	63	56	32
14	63	63	48
15	63	63	63

FP_COLOR.256

USER DEFINED COLOR MAP
256 COLORS

0	1	0	0
1	2	1	0
2	3	1	0
3	4	2	0
4	5	2	0
5	6	3	0
6	7	3	0
7	8	4	0
8	9	4	0
9	10	5	0
10	11	5	0
11	12	6	0
12	13	6	0

13	14	7	0
14	15	7	0
15	16	8	0
16	17	8	0
17	18	8	0
18	19	9	0
19	20	9	0
20	21	9	0
21	22	10	0
22	23	10	0
23	24	11	0
24	25	11	0
25	26	11	0
26	27	12	0
27	28	12	0
28	29	12	0
29	30	13	0
30	31	13	0
31	32	14	0
32	32	14	0
33	32	14	0
34	32	14	0
35	33	14	0
36	33	14	0
37	33	14	0
38	33	14	0
39	34	15	0
40	34	15	0
41	34	15	0
42	34	15	0
43	35	15	0
44	35	15	0
45	35	15	0
46	35	15	0
47	36	16	0
48	36	16	0
49	36	16	0
50	36	16	0
51	37	16	0
52	37	16	0
53	37	16	0
54	37	16	0
55	38	17	0
56	38	17	0
57	38	17	0
58	38	17	0
59	39	17	0
60	39	17	0
61	39	17	0
62	39	17	0
63	40	18	0

64	40	18	0
65	40	18	0
66	40	18	0
67	40	18	0
68	40	18	0
69	40	18	0
70	40	18	0
71	41	19	0
72	41	19	0
73	41	19	0
74	41	19	0
75	41	19	0
76	41	19	0
77	41	19	0
78	41	19	0
79	42	20	0
80	42	20	0
81	42	20	0
82	42	20	0
83	42	20	0
84	42	20	0
85	42	20	0
86	42	20	0
87	43	21	0
88	43	21	0
89	43	21	0
90	43	21	0
91	43	21	0
92	43	21	0
93	43	21	0
94	43	21	0
95	44	22	0
96	44	22	0
97	44	22	0
98	44	22	0
99	44	22	0
100	44	22	0
101	44	23	0
102	44	23	0
103	45	23	0
104	45	23	0
105	45	23	0
106	45	24	0
107	45	24	0
108	45	24	0
109	45	24	0
110	45	24	0
111	46	25	0
112	46	25	0
113	46	25	0
114	46	25	0

115	46	25	0
116	46	25	0
117	46	26	0
118	46	26	0
119	47	26	0
120	47	26	0
121	47	26	0
122	47	27	0
123	47	27	0
124	47	27	0
125	47	27	0
126	47	27	0
127	48	28	0
128	48	28	0
129	48	28	0
130	48	28	0
131	49	28	0
132	49	28	0
133	49	28	0
134	49	28	0
135	50	29	0
136	50	29	0
137	50	29	0
138	50	29	0
139	51	29	0
140	51	29	0
141	51	29	0
142	51	29	0
143	52	30	0
144	52	30	0
145	52	30	0
146	52	30	0
147	53	31	0
148	53	31	0
149	53	31	0
150	53	31	0
151	54	32	0
152	54	32	0
153	54	32	0
154	54	32	0
155	55	33	0
156	55	33	0
157	55	33	0
158	55	33	0
159	56	34	0
160	56	34	0
161	56	34	0
162	56	34	0
163	57	34	1
164	57	34	1
165	57	34	1

166	57	34	1
167	58	35	2
168	58	35	2
169	58	35	2
170	58	35	2
171	59	35	3
172	59	35	3
173	59	35	3
174	59	35	3
175	60	36	4
176	60	36	4
177	60	36	4
178	60	37	4
179	61	37	5
180	61	37	5
181	61	38	5
182	61	38	5
183	62	39	6
184	62	39	6
185	62	39	6
186	62	40	6
187	63	40	7
188	63	40	7
189	63	41	7
190	63	41	7
191	63	42	8
192	63	42	8
193	63	42	9
194	63	43	9
195	63	43	10
196	63	43	10
197	63	44	11
198	63	44	11
199	63	45	12
200	63	45	12
201	63	45	13
202	63	46	13
203	63	46	14
204	63	46	14
205	63	47	15
206	63	47	15
207	63	48	16
208	63	48	17
209	63	49	18
210	63	49	19
211	63	50	20
212	63	50	21
213	63	51	22
214	63	51	23
215	63	52	24
216	63	52	25

217	63	53	26
218	63	53	27
219	63	54	28
220	63	54	29
221	63	55	30
222	63	55	31
223	63	56	32
224	63	56	33
225	63	57	34
226	63	57	35
227	63	58	36
228	63	58	37
229	63	59	38
230	63	59	39
231	63	60	40
232	63	60	41
233	63	61	42
234	63	61	43
235	63	62	44
236	63	62	45
237	63	63	46
238	63	63	47
239	63	63	48
240	63	63	49
241	63	63	50
242	63	63	51
243	63	63	52
244	63	63	53
245	63	63	54
246	63	63	55
247	63	63	56
248	63	63	57
249	63	63	58
250	63	63	59
251	63	63	60
252	63	63	61
253	63	63	62
254	63	63	63
255	63	63	63

INTERNAL DISTRIBUTION

- | | |
|---------------------|--|
| 1. B. A. Bervan | 15. J. C. Ryman |
| 2. W. D. Brosey | 16-18. C. H. Shappert |
| 3. D. A. Carpenter | 19. A. L. Sjoreen |
| 4. M. Cristy | 20. R. E. Swaja |
| 5. K. F. Eckerman | 21. M. A. Taylor |
| 6. T. K. Evers | 22-24. R. C. Ward |
| 7. R. H. Fowler | 25. G. E. Whitesides |
| 8. G. D. Kerr | 26. Central Research Library |
| 9. D. H. Hetrick | 27. ORNL Y-12 Research Library
Document Reference Section |
| 10. J. T. Holdeman | 28. Laboratory Records Department |
| 11. W. L. Jackson | 29. Laboratory Records, ORNL (RC) |
| 12. D. C. Kocher | 30. ORNL Patent Office |
| 13. R. W. Leggett | |
| 14. M. R. Patterson | |

EXTERNAL DISTRIBUTION

31. F. M. Fota, USAEHA, ATTN: HSHB-MR-HM, Aberdeen Proving Ground, MD 21010-5422
32. J. C. Johnson, HQAMC, ATTN: AMCSG-R, 5001 Eisenhower Avenue, Alexandria, VA 22333-0001
33. C. B. Nelson, U.S. Environmental Protection Agency, Office of Radiation Programs, ANR-461, 401 M Street, S.W. Washington, DC, 20460
34. M. T. Stabin, Radiopharmaceutical Internal Dose Center, Oak Ridge Associated Universities, MERT, P.O. Box 117, Oak Ridge, TN 37831
35. Office of the Deputy Assistant Manager for Energy Research and Development, Department of Energy Oak Ridge Operations (DOE-ORO), P.O. Box 2008, Oak Ridge, TN 37831
- 36-38. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831

