NBSIR 85-3235

A Device Independent Graphics Kernel

Walter W. Jones and
Alicia B. Fadell*

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Fire Research
Gaithersburg, MD 20899

*Current Address - Department of Mathematics
University of Maryland
College Park, Maryland

May 1985

Issued October 1985

U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NBSIR 85-3235

A DEVICE INDEPENDENT GRAPHICS KERNEL

Walter W. Jones and
Alicia B. Fadell*

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Fire Research
Gaithersburg, MD 20899

*Current Address - Department of Mathematics
University of Maryland
College Park, Maryland

May 1985

Issued October 1985

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

TABLE OF CONTENTS

	Page
List of Figures	iv
List of Tables	v
1. Introduction	1
2. Overview	1
3. Schematic of the transform from the application window to a device dependent world space	7
4. Structure of the Data Files	9
5. Input/Output Devices	11
6. Implementation	12
6.1 BUILD	12
6.2 TITLE	17
6.3 ADDMAP	19
Appendix	
A. Description of Routines and Their Arguments	22
B. Control Sequences for WDDRAW	60
C. Character Sets (1-24)	61
D. Description of BUILD commands	86
E. Description of TITLES commands	100
F. Examples of Usage of Routines	105
G. Listing of Device	139

List of Figures

Page

1. Interface between application program and devices or processors........ 2

2. Schematic of the transform from the application window to a device dependent world space... 6

3. Schematic of the transform from the application window to a viewpoint on the viewing surface.. 7

4. Filling pattern for "FILTYP"...36

5. Default color lookup table for Lexidata 8100............................52

6. Plot geometry used in "SURFAC"; this shows the naming convention used for the skirts and surfaces...53

7. Overlapping spheriods with hidden lines removed........................58

List of Tables

Page

1. Sample of a hardware interface protocol....2

2. General format and grouping of graphics routines....4

3. Device coordinates....8

4. Listing from BUILD - corresponding to data file shown in Table 3....10

5. Coordinate limits of the hardware....10

A DEVICE INDEPENDENT GRAPHICS KERNEL

Walter W. Jones
Alicia B. Fadell

Abstract

This paper describes an interface for programs which allows one to write graphics primitives to several devices without regard for the type of device. The most salient features are that it has low overhead, is transportable and can be expanded as the nature of the input/output devices changes. A conscious effort has been made to include all normal graphics primitives together with the most useful high level routines without compromising the use of special features of custom display units.

Keywords: device independence, display devices, graphics

1. INTRODUCTION

This paper describes a graphics package which is intended to ease the use of input/output devices in acquiring and displaying information graphically. The intent is to reduce the problem to its simplest level by allowing one to describe graphs in much the same way one thinks of them. A further intent is to make the user language truly device independent and allow a programmer or other user to switch devices interactively simply by identifying the desired input and output devices. As much as possible, the similarity in instructions to different devices is maintained. There are some limitations of course. Pen plotters do not normally come with erasers and most storage scopes (e.g., Tektronics) are not erasable on the individual pixel level. Beyond this no function which is available for a particular device, but not supported because there is no commonality amongst the devices, is rendered unavailable. Thus, even for specialized usage, this package will take care of normal initialization and setup.

The devices which are currently supported are Plot-10 emulators, CALCOMP pen plotters, a line printer and the Lexidata 3400/8100 series[1]. These devices encompass most protocols and slots have been left in the package for future expansion. We currently support these devices since those are the ones which are available. Any suggestions for expansion or additional functions are welcome.

2. OVERVIEW

Graphics application programs may be required to send output to or accept input from several devices. Since each output device is supported by different sets of graphics routines, writing application programs is normally

[1]Certain commerical equipment is identified in this paper in order to illustrate adequately certain device specific characteristics. Such idenification does not imply recommendation or indorsement by the National Bureau of Standards, nor does it imply that the equipment is necessarily the best availale for the purpose.

burdensome since applications may reguire several device-dependent versions of each program. Alternatively, each application could include all the device-dependent routines in one version of each program, and call only the routines required for a specific device, e.g., by means of GO TO statements. In either case, when a new device is added to the work environment, application programmers must learn another set of graphics routines and write new codes for each application. This reprogramming can be time-consuming and costly.

A graphics system which supports several input and output devices, while hiding the device-dependent routines from the user, is desirable. DEVICE is a package of FORTRAN subroutines for producing graphics output on several devices. The package consists of one standard routine for each output primitive (line, polygon, character, etc.). Once a device has been identified, a primitive is displayed through a call to the appropriate routine. This routine is also used to generate the same output on other devices. Hence DEVICE is an interface between the application program and the input/output devices since each standard routine is device-independent, as shown in Figure 1.

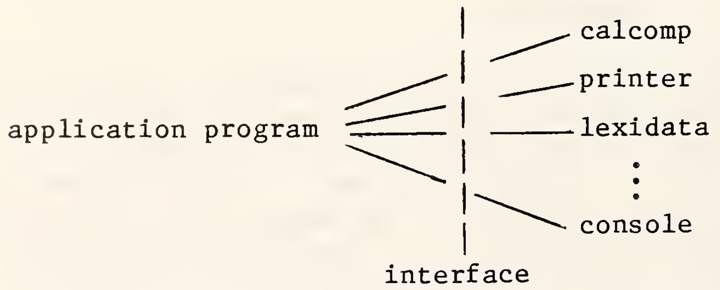


FIGURE 1

Every hardware interface routine is divided into sections. Each of these sections is devoted to a device. Once a primitive is called by a program, command goes to the section designed for the accessed device, as shown in table 1.

Table 1

A Sample of the Hardware Interface Protocol

```
          SUBROUTINE SAMPLE
          :
          COMMON/DEV/IDEV
          GO TO (1,2,...n) IDEV
C         CALCOMP SECTION
1         :    graphics subroutines
          RETURN
C         PRINTER SECTION
2         :    graphics subroutines
          RETURN
          :
          END
```

The coordinates used in labeling a viewing area differ for each device. Therefore, the user's data points must be converted into device coordinates (see section 5). This conversion is done by DEVICE before device-dependent routines are called. The format of a typical graphics subroutine is

```
          SUBROUTINE SAMPLE
          :
          COMMON/DEV/IDEV
          GO TO (1,2,...n) IDEV
C         CALCOMP SECTION
1         convert to CALCOMP coordinates
          call CALCOMP graphics routines
          RETURN
C         PRINTER SECTION
2         convert to PRINTER coordinates
          call PRINTER graphics routines
          RETURN
          :
C         CONSOLE SECTION
n         convert to CONSOLE coordinates
          call CONSOLE graphics routines
          RETURN
          END
```

2.1 Structure of the Package

These routines can be grouped into 6 categories summarized below and defined in table 2. More detail is given in the apprendixes.

1. Device Control - utility routines which include initialization and termination procedures for the device

2. Viewing - specify the part of the user's coordinate system to display and where to place the display on the viewing surface

3. Output Primitives - define objects and display them on the viewing surface

4. Attributes - define the appearance of the output primitives

5. Auxiliary - miscellaneous routines

6. Input - acquire data.

Table 2. Graphics Subroutines - General Format

1. Device Control	Function
DEVICE (n)	initialize a graphics device
NEWFRM	clear the screen and initialize a new frame
HDCOPY	generate a hard copy
FRAME	write out the buffer
ERASE	clear the screen (also done in "NEWFRM")
ENDFRM	close the graphics device

2. Viewing	Function
SCALNG (X1, Y1, X2, Y2, X1H, Y1H, X2H, Y2H, X1S, Y1S, X2S, Y2S)	define the window and viewport
DEFINE (X1, Y1, X2, Y2)	define the window (viewport defaults to entire viewing surface)

3. Output Primitives	Function
LINE (X1, Y1, X2, Y2)	draw a line between 2 points
LINES (X1, Y1, X2, Y2, N)	draw a line between points in an array of length n
LNPLOT (X, Y, I1, I2, I3)	draw lines between selected points of an array
PLYGON (X, Y, N)	draw a closed polygon
BOXPLT (X1, Y1, X2, Y2)	draw a rectangle
CIRCLE (X, Y, R)	draw a circle
SURFAC (Z, NX, NY, MODE)	performs surface plotting
CONTUR (F, TEST, NX, NY, FL)	performs contour plotting
VOLUME (MODE, F, NX, NY, NZ, CLEVEL, NCL, T, NT)	performs volume plotting (3 dimensional contouring)
SYMBOL (X, Y, CHAR)	draws a specified hardware character at a given point
CHPLOT (X, Y, CHAR, I1, I2, I3	draws a specified hardware character at selected points in an array
HHDRAW (X, Y, SX, SXY, ICHAR, NSET, NSET, IERR)	draws a particular character of the specified character set
WDDRAW (X, Y, DX, DY, SX, SXY SY, CHARS)	draws a character string

LABEL (CHARS, X1, Y1, X2, Y2, ANGLE)	draws a character string
ALABEL (CHARS, X1, X2, Y1, Y2, ANGLE)	draws a character string with aspect ratio 4/3
FNUMBR (X, Y, DX, DY, SX, SXY, SY, XNUMBR, WIDTH, DIGITS)	draws a real number in FORTRAN F-type format
ENUMBR (XNUMBR, X1, Y1, X2, Y2)	draws a real number in exponential (E) format (E - type)
GRAFIT (NPLT, X1, X2, X1R, X2R, XX1, XX2, Y1, Y2, Y1R, Y2R, YY1, YY2, XTIT, NDVX, YTIT, NDVY)	sets up a graph (x and y axis) for plotting data
PLYPLT (F, IS)	read in a "BUILD" formatted structure file, F and display according to IS
MAPIN (FW, V, NV, E, ES, NE, P, PS, NP)	read a "BUILD" file and return the vertices, edges and polygons. Files are appended after the initial call.
MAPOUT (W, V, NV, E, ES, NE, P, PS, NP)	display the specified edges as given
VIEWTR (X, W, V, NV, E, ES, NE, P, PS, NP)	display the specified edges and polygons using the transform matrix X.

4. Attributes — Function

COLOR (N)	defines the color for drawing
LINWID (N)	defines the line width
FILTYP (N)	defines the appearance of the interior of polygons and circles
CHRSIZ (CHFRZ, GCHFRZ)	sets the size of the hardware characters
CHRSET (N)	changes the default character set

5. Auxiliary — Function

IOWAIT (N)	puts a pause in the program
SETDEV (N1, N2)	changes logical units
DELAY (N)	delays for hard copy units
SETLUT	sets up the color look-up table

6. Input

DEVINP (I, status, X,Y,Z)	read input coordinates triplet from a device

3. COORDINATE SYSTEMS

The application must define the environment in which it will operate. For instance, temperature may be calculated in degrees celsius for one study, while length is measured in feet for another project. Data can be sent to the DEVICE routines directly from the application; no conversion to a standard unit is required of the user. Within the graphics package, however, the data coordinates must be converted to device-dependent coordinates for the individual hardware.

The user space is a 3-dimensional left-handed coordinate system (positive x-axis to the right, positive y-axis up, and positive z-axis into the viewing surface). Data points are added to the space in "application-dependent" coordinates. Before these points can be displayed, a window and a viewport must be specified.

The window, a rectangle in the x-y plane of the user space, encloses the points to be displayed. This window is mapped into the device's world space. The world space is labeled differently for each device. The world space is determined by the device's fixed points (X1S, Y1S) and (X2S, Y2S). The window is defined by the user who specifies the diagonal endpoints (X1, Y1) and (X2, Y2). The window is mapped to the world space by the scale factors:

$$\begin{aligned} \text{XYCOORD}(1) &= (\text{X2S} - \text{X1S})/(\text{X2} - \text{X1}) \\ \text{XYCOORD}(2) &= \text{X1S} - \text{X1} * \text{XYCOORD}(1) \\ \text{XYCOORD}(3) &= (\text{Y2S} - \text{Y1S})/(\text{Y2} - \text{Y1}) \\ \text{XYCOORD}(4) &= \text{Y1S} - \text{Y1} * \text{XYCOORD}(3) \end{aligned}$$

A point (X,Y) in the user space is converted to the world coordinate (XS,YS) by the transformation:

$$\begin{aligned} \text{XS} &= \text{X} * \text{XYCOORD}(1) + \text{XYCOORD}(2) \\ \text{YS} &= \text{Y} * \text{XYCOORD}(3) + \text{XYCOORD}(4) \end{aligned}$$

Any point within the user's window will be converted into valid world coordinates (i.e., X1S $\leq$ XS $\leq$ X2S and Y1S $\leq$ YS $\leq$ Y25) and can be plotted. Other points will be mapped out of the device's range and should not be displayed on the viewing surface.

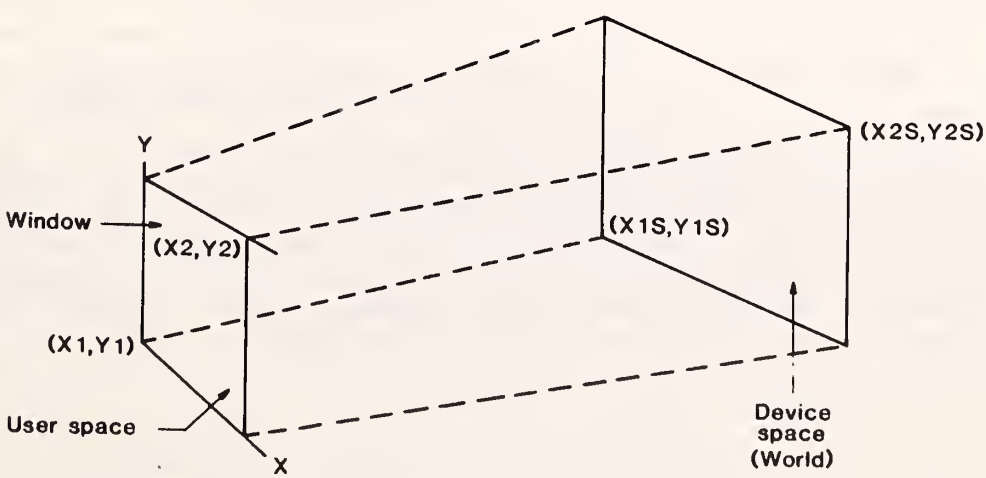


FIGURE 2

The part of the viewing surface which will contain the display is called the viewport. The viewport defaults to the entire viewing surface. In order to place the display on a smaller portion of the viewing surface, the desired rectangular area must be specified. The device coordinates (X1H, Y1H) and (X2H, Y2H) are used to define this viewport. The world space is then mapped onto the viewport. Hence, all data points in the user space are converted to device coordinates through 2 transformations. Only points in the user's window will be transformed into valid device coordinates (i.e., coordinates within the device's viewport).

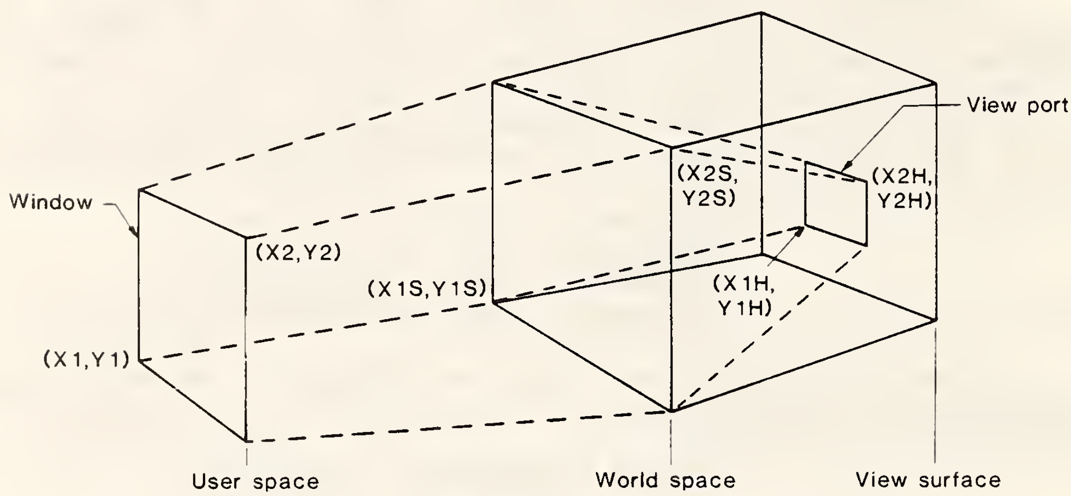


FIGURE 3

Usual usage is to transform the viewport to the entire viewable space (view surface). In this case a call to DEFINE is sufficient. The default values for the device world coordinate space are used, and the defined window is mapped to the maximum normalized device coordinates. Special applications might require other tranformations. Such a facility is provided by SCALNG. This routine specifies the transform from user space to world coordinate space, from world coordinate space to normalized device coordinates and finally to the device (or hardware) address units. Table 3 shows the default values for the current list of output device.

TABLE 3
DEVICE COORDINATES

OUTPUT	WORLD		NORMALIZED		PHYSICAL	
	X	Y	X	Y	X	Y
Lexidata	128000	102300	32700	26160	1279	1023
Tek 40XX	1280	1024	1029	1029	1023	768
CALCOMP	-	-	1029	1023	10.75	8.75
Printer	1279	1023	1279	1023	128	50

The physical (device) coordinates map windows to the full viewing space. Some applications require use of only a portion of this space. An example is the reduction of about 15% of the linear dimensions in order to fit into a 512 line raster scan television frame.

4. STRUCTURE DATA FILES

Several programs create and modify structure files for graphical display. The elements used to compose these structures are vertices, edges, and polygons. The elements are stored in user coordinates in the data file. Attributes for these elements along with the specifications of the user space, are also stored in this file.

Listed at the beginning of the data file are the dimensions of the user space and the window space. The boundaries of the window are put into the WINDOW array: WINDOW (1 → 4) {(left - bottom), and (right - top), respectively}, while the boundaries of the user space are sent to the WORLD array: WORLD (1 → 6) {left, bottom, front, and right, top, back}, respectively. This latter three dimensional space specification is not currently used.

The following two images in the data files specify the number of vertices, elements (defined as polygons + edges) and the number of edges, and a description of the elements by groups. This latter is currently implemented only in ADDMAP, described in Section 6.3.

The data for the elements and attributes comprise the remaining part of the file. The vertex coordinates (x,y,z) are stored next, followed by element pointers. The input routines read the data for the vertices into the array VERTEX (i,j). Here, the row index, i, (possible values = 1,2,3) refers to the i^{th} coordinate (x, y, or z, respectively) and the column vector j refers to the vertex number. For instance VERTEX (2,5) stores the value of the y coordinate of the 5^{th} vertex. In the file, each coordinate triplet is preceded by zero (0) or one (1). A zero indicates that the vertex was deleted from the user space (i.e., the vertex is not used for displaying or for constructing edges and polygons) but was not deleted from the data file. A one (1) specifies that the vertex exists in the user space. An example would be deleting a polygon without removing the corresponding vertices.

The vertices are grouped together to form edges and polygons. Following the list of vertex coordinates is the list of elements. Each element, either an edge or a polygon, is identified by its endpoints which are in

EDGE (i,j) or POLY(i,j),

where i specifies which of the two endpoints and j indicates the element number. For example, if the fourth edge is composed of vertices 6 and 10, we have

EDGE (1,4) = 6 and EDGE (2,4) = 10

Polygons are stored in a similar manner. POLY(i,j) contains the vertex number of the i^{th} vertex of the j^{th} polygon. Polygons have no more than NPVERT (currently 8) vertices.

Data for each edge or polygon fills one row of the data file. The element's vertices are listed first and are followed by the element's attributes. The attributes are read into the arrays ESPEC (i,j) or PSPEC (i,j) for edges and polygons, respectively. Here the i specifies the attribute and j indicates the element number.

i	ESPEC (i,j)
1	line width
2	color
3	line attribute
4	<unused>

i	PSPEC (i,j)
1	line width
2	color
3	fill type
4	polygon (≥0) or polyline (<0)

A sample data file is shown in Table 3. In this file there are 9 valid vertices, 1 "deleted" vertex, 1 edge, and 3 polygons. (Note that the last polygon is actually a polyline.) The same data as listed by the BUILD program is shown in Table 4.

Table 3

```
BUILD 0.00000E+00 0.00000E+00 1.28000E+03 1.02400E+03
-1.00000E+09-1.00000E+09-1.00000E+09 1.00000E+09 1.00000E+09 1.00000E+09
 1           10    4    1
 1            4
 1         1.55608E+02  7.37531E+02 0.00000E+00
 1         4.03576E+02  9.39818E+02 0.00000E+00
 1         5.00956E+02  7.45070E+02 0.00000E+00
 1         2.60015E+02  6.44554E+02 0.00000E+00
 0         4.36470E+02  3.88245E+02 0.00000E+00
 1         9.64768E+02  6.44554E+02 0.00000E+00
 1         5.30060E+02  3.68136E+02 0.00000E+00
 1         5.30070E+02  5.88014E+02 0.00000E+00
 1         8.03136E+02  5.88014E+02 0.00000E+00
 1         8.03136E+02  4.63627E+02 0.00000E+00
           4  6  0  0  0  0  0  0  0  2  1  0  0
           1  2  3  0  0  0  0  0  0  2  1  0  0
           6  7  8  9  0  0  0  0  0  2  8  1  0
           1  4  7  0  0  0  0  0  0  2  1  0 -1
```

The edge, polygon and polyline entries in Table 3 are integer format. In this case there are 13 entries for each element. For polygons and polylines, NPVERT entries, a zero for compatibility with"MOVIE.BYU" and the 4 NSPEC entries. For edges, here are two entries, seven zeros and the 4 ESPEC entries.

Table 4

	WORLD			WINDOW		
LEFT,BOTTOM,FRONT:	-1.0E+09	-1.0E+09	-1.0E+09	LEFT BOTTOM:	0.0	0.00
RIGHT, TOP, BACK:	1.0E+09	1.0E+09	1.0E+09	RIGHT,TOP:	1.3E+03	1.0E+03

	VERTEX(X,Y,Z)		
1)	155.61	737.53	0.00000
2)	403.58	939.82	0.00000
3)	500.96	745.07	0.00000
4)	260.02	644.55	0.00000
6)	964.77	644.55	0.00000
7)	530.07	368.14	0.00000
8)	530.07	588.01	0.00000
9)	803.14	588.01	0.00000
10)	803.14	463.63	0.00000

	EDGES									SPECS		
1)	4	6							2	1	0	0
	POLYGONS									SPECS		
1)	1	2	3	0	0	0	0	0	2	1	0	0
2)	6	7	8	9	0	0	0	0	2	8	1	0
3)	1	4	7	0	0	0	0	0	2	1	0	-1

5. INPUT/OUTPUT DEVICES

Currently four separate input and output devices are supported. These encompass all the common standard interfaces and the flexibility exists to include any future input/output device for which a transformation between the hardware and mathematical space can be given. The devices for output are a CALCOMP pen plotter (1012), Tektronix 40XX or emulator, Lexidata Displays (8100 and 3700) and a line printer. Devices for input are a digitizing tablet, a joystick, segments in the Lexidata picture descriptor lists which can be "picked" and ASCII input from a keyboard or file.

The software maintains three dimensional picture descriptors. Although only the Lexidata 3700 can utilize this capability directly, perspective and visual cuing can be shown on the other devices. The coordinate limits of the hardware are shown in Table 5.

TABLE 5

DEVICE	Xmin/Xmax	Ymin/Ymax	Zmin/Zmax
Tektronix 40XX	0/768	0/1023	-
Printer	1/128	format limited	-
Joystick	0/32767	0/26160	-
Lexidata 8100	0/1279	0/1023	-
Lexidata 3700	0/1279	0/1023	0/4095
Tablet	0/12190	0/9142	-
Segments	0/~ 10^9	0/~ 10^9	
Calcomp plotter	0/11.5	0/8.5	

These are the local address modes for pixels and not the normalized device coordinates (see section 3). This group of devices includes almost all standard protocols for graphics interfaces. These are the CORE standard, escape sequences, direct (DMA) input/output of pixels, line oriented input/output (ASCII) and the very early protocols formulated by Tektronix and CALCOMP. Slots have been left in the software to accomodate new devices. At the time of this writing, the GKS standard has not been formalized. However, if it is similar to the proposed standard then inclusion of devices which follow this protocol is straightforward.

6. IMPLEMENTATION

6.1 BUILD

The BUILD program is used to create, modify, or display files consisting of vertices, edges, and polygons. These files also contain the elements' characteristics: line width, color, and polygon fill type. The first step in creating a file is to specify the user space and the window. The user space defaults to a cube centered at the origin and having sides of length 6×10^9. Prompts are used by the programs when needed. There are no default values for the window space. The x and y coordinates of the window must be specified with a call to

WINDOW

At this point, data for the various structures may be entered and subsequently modified or displayed. The work file, however, need not be a new file. It is possible to alter or display data of an existing file. The command to access an old file is

GET [filename]

where the file defaults to INFILE, the last file accessed. Similarly, a work file may be saved by

SAVE [filename]

In this case, the default file is OUTFILE which is the last file that was saved. A check of the filenames stored in INFILE and OUTFILE is possible by calling

STATUS

This command also indicates the number of vertices, edges, and polygons in the work file along with the maximum number of each element allowed.

The work file is modified in several ways. Elements are added to the file with

ADD { VERTEX / EDGE / POLYGON } (default = VERTEX)

Vertices are input by specifying three (x,y,z) coordinates. Edges are specified by pointing to two vertices and polygons are defined by indicating a minimum of 3 vertices to a maximum of NPVERT vertices.

Elements can be added from any of the input devices. The command

SET { JOYSTICK / TABLET / KEYBOARD } (default = KEYBOARD)

sets the default input device. At the start of program execution, the keyboard is the data input device. To add data from a different device, SET must be used before ADD.

If the attributes of new edges or polygons are to be different than the default characteristics, the command

SELECT

must be used before the ADD command. SELECT allows the specification of the line width, color, polygon fill type and a user specification parameter, the fourth element in ESPEC and NSPEC. These new values then become the default attributes.

Elements are deleted from the file with the command

DELETE { VERTEX / EDGE / POLYGON } (default = VERTEX)

When a vertex is deleted, any edge or polygon containing that vertex is also deleted. It should be noted that the vertex is not deleted from the permanent data file; instead, this point is marked with a flag to indicate deletion from the user space. In order to remove vertices from the data file, the

SQUEEZE

command is used. All "flagged" vertices, along with any vertices in the user space which are not used by an edge or a polygon, are deleted from the data file by SQUEEZE. When the DELETE command is used for polygons and edges, however, these elements are removed from the user space and the data file.

Once added to the user space, the elements can be modified. Elements are moved to new locations through the command

MOVE {VERTEX | EDGE | POLYGON} (default = VERTEX)

A vertex can be moved by specifying its new (x,y,z) coordinates. Alternatively, a single vertex or a group of vertices can be moved a distance given by an (x,y,z) displacement vector. MOVE EDGE allows a single edge or a group of edges to change location by a specified (x,y,z) distance. Polygons are moved in the same manner as edges. "MOVE" makes new verticies when moving edges or polygons. In addition, duplicates of elements can be created with

DUPLICATE {EDGE | POLYGON} (default = VERTEX)

The new elements are positioned at the designated displacement from the original position.

In addition to modifying portions of the model, the entire structure can be transformed. The model is moved to another part of the user space with the command

TRANSLATE

accompanied by an (x,y,z) displacement vector. A rotation of the model about a specified point (default is the body center of gravity for the model) is possible with

ROTATE

The rotations occur about axes parallel to the x-, y-, or z-axes of coordinate system. The axis and angle of rotation are specified. The structure is scaled about its center point by

SCALE

which requires the input of a positive scale factor. If the scale factor is greater than one, the model will be magnified. If the factor is between zero and one, the structure will be reduced. Each scaling, rotation, and translation, along with the order of application, is recorded in a matrix called the transformation matrix. This matrix contains the information needed to directly convert the structure from the original configuration to the final position determined by the series of transformations.

A listing of the elements in the work file or of the transformation matrix can be obtained at the terminal with

$$\text{LIST} \quad \begin{Bmatrix} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \\ \text{MATRIX} \end{Bmatrix} \qquad \text{(default = VERTEX)}$$

Heading the vertex list are the world and window coordinates of the work file. The (x,y,z) coordinates of a designated group of vertices follow. These vertices are listed by numbers; missing numbers correspond to vertices which were removed from the user space with DELETE. The LIST EDGE command also requires the specification of a group of edges. Each edge's number and endpoint vertices are listed on a row. Similarly, the polygon numbers and component vertices are listed for a group of polygons by the LIST POLYGON command. Finally the transformation matrix is displayed with LIST MATRIX.

The lists can be sent to the printer instead of the terminal with the command

$$\text{PRINT} \quad \begin{Bmatrix} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \\ \text{ALL} \\ \text{MATRIX} \end{Bmatrix} \qquad \text{(default = ALL)}$$

The PRINT commands produce lists in the same format as the corresponding LIST commands. However, groups of elements are not specified by the user. The PRINT element command lists the first through the last "element" stored in the work file. The additional command, PRINT ALL, lists the world and window coordinates; all of the vertices, edges, and polygons; and the transformation matrix.

The contents of the work file can be displayed graphically. At the start of program execution, the output device for the display defaults to the Lexidata. A different device is selected with

$$\text{DEVICE} \quad \begin{Bmatrix} \text{CALCOMP} \\ \text{PRINTER} \\ \text{LEXIDATA} \\ \text{CONSOLE} \end{Bmatrix} \qquad \text{(default = LEXIDATA)}$$

The specified device becomes the new default for graphical display.

If the output device has associated hard copies, these copies can be generated with

$$\text{COPY} \quad \begin{Bmatrix} \text{ON} \\ \text{OFF} \end{Bmatrix} \qquad \text{(default = OFF)}$$

After COPY ON has been entered, a specified number of hard copies of each subsequent graphics diplay will be produced. To discontinue this automatic duplication, COPY OFF must be entered. Note that the copy switch is in the OFF position at the start of program execution.

The model is displayed with

$$\text{DISPLAY} \begin{Bmatrix} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \\ \text{ALL} \end{Bmatrix} \qquad \text{(default = ALL)}$$

If the ouptut device is a screen, the screen is cleared before the display is generated. Only the elements within the window are displayed. DISPLAY VERTEX plots the model's vertices and labels each vertex with its number. DISPLAY EDGE or DISPLAY POLYGON draws the models' edges or polygons, respectively without numbering them. All edges and polygons are shown with DISPLAY ALL. A screen can be cleared with

ERASE

Otherwise, the image will remain on the screen until DISPLAY is used again or until termination of the program.

The field of view for a display is altered with

FIELD

The point of observation can be moved relative to the x-y plane of the user's system. A distance of zero places the observer at the origin of the coordinate system. A positive distance, d, positions the observer d units on the negative z-axis. The perspective of the display changes as the viewer's distance is altered. In addition, the angle of view can be changed. This angle, originating at the observer and bisected by the z-axis, determines the scope of vision. For instance, as the angle is decreased, the scope of vision becomes narrower. At the start of program execution, the observer is positioned 10,000 units from the origin and the angle of view is 90°.

At times it is useful to modify a structure throughout a sequence of frames. The number of frames and the changes desired are specified with

ANIMATE

The structure can be translated, scaled about its center, or rotated about a particular point. Edges and polygons may be moved. The observer can be moved along the z-axis. Any combination of these modifications is possible. Note that ANIMATE only receives the parameters for each change. The sequence of frames is viewed with DISPLAY.

Some transformations may cause the model to be moved outside the window. If the location of the structure becomes unknown, there may be difficulties in retrieving the model. The model is found with

AUTO

The window and the observer are automatically moved so that the entire model can be seen. As a result of AUTO, the scale of the model may not be optimum for viewing. This situation is easily modified with SCALE and FIELD.

The commands and the user's reponses to the subsequent prompts do not have to be entered at the console. The input device is changed to a specified file with

INPUT

All input is read from this file until another input file is specified. The

EXIT

command changes the input device to the console.

Finally, the program is terminated with

END

While the program is running, the

HELP

command is used to list all the commands together with a brief description of them.

6.2 TITLES

The TITLES program is used to create and modify a series of colored pictures consisting of character strings, lines, and circles. The strings are positioned on the screen by means of a joystick. Characters from any combination of the 24 character sets (see Appendix C) are used to form a string. Lines, circles, and bullets also may be added to the picture. Elements are added in various colors and sizes. Once the pictures (usually called slides) have been created, hard copies may be generated. TITLES is commonly used to produce slide presentations.

At the start of program execution, there is the option of accessing an existing data file or composing a new file. The desired option is specified by entering the filename, or, to indicate the creation of a new file, by hitting the <RETURN> key.

Once an existing file has been read, the user is asked for a command with the prompt FUNCTION=. Prompts are given when needed. If the file is new, the command defaults to

NEW,

a new slide is created and added to the work file. The default color and character set for the text on this slide is specified. Lines of text are entered and then positioned with the cursor. Colors and character sets may be changed within a character string by means of control sequences. Control sequences are also used to add subscripts and superscripts and to change the justification of the text (see Appendix B). These attributes, however, are returned to the default values for the next line of text.

Once a character string is placed on the screen, modifications may be made. The string may be deleted, moved, centered on the line, and scaled. Note that the character size resulting from a scaling becomes the default size for all characters subsequently entered. This process of adding and modifying text is continued until the <RETURN> key is entered instead of text characters. At this point, the user is informed of the number of slides in the work file.

Slides are modified in several ways. Lines of text can be corrected one line at a time with the command

CORRECT

A character string can be deleted, moved, centered on the line and scaled. When all corrections have been made to one line of text, the process is repeated for the remaining character strings on the slide.

Various elements may be added to a slide. Text is added with the command

ADD

The default color and character set are selected. Lines of text are entered and modified as with the NEW command. Other types of elements are drawn in specified colors and at given locations. Bullets (filled circles) are added with the command

BULLET

A straight line is drawn between two specified points with

LINE

Finally, with a center point and a radial distance designated, a circle is added to the slide with

CIRCLE

Once drawn, each element may be corrected in the usual manner. It may be deleted, moved, centered, or scaled.

Specified slides in the work file may be displayed on the Lexidata with the command

VIEW

Hard copies are generated with the command

PROCESS

Slides are specified along with the desired number of copies of each slide. The delay time for the hard copy device is changed with

DELAY

and the work file may be written and saved as a data file with the command

SAVE

If the work file is new, a filename is specified by the user. Otherwise, the slides are automatically written to the file last accessed. In order to access another data file, the

REREAD

command is used.

The various character sets and colors may be referenced. The command

SET

displays a specified character set. Each set has been arranged to correspond to the 96 displayable ASCII characters. When adding text to a slide, a desired character is placed in the character string by entering the corresponding ASCII character. The command

COLORS (default = 1)

displays all the available colors in rows. The colors are numbered to correspond with the rows. The color in the top row is color number one, the next row is color number two, and so on.

The two remaining commands are HELP and END. The commands are listed (but not described) with

HELP

Finally, the program is terminated with

END

6.3 ADDMAP

This is a structure manipulation program to add pieces of structure file together. The commands are similar to those found in BUILD and TITLE. There are some additional commands to manipulate the individual pieces. The commands are

SET, NEW, GET, WINDOW, ADD, SAVE, DISPLAY, SPECS, DRAG, GROUP, TRANSLATE, ROTATE, SCALE, FIELD, DEVICE, HELP and END.

The only commands described here are those which differ from the explanation in sections 6.1 and 6.2. The implementation of several commands, such as SET and GET, differs in BUILD and TITLES, but accomplishes the same task.

The special commands to deal with adding structures involve those which force actions on only a portion of the total file. These are NEW, DRAG and

GROUP.

In general, all manipulation commands are applied to all extant vertices. However, as pieces are added with repeated GET requests, a table is maintained which points to each of these GROUPS of vertices, polygons, edges and polylines. In order to apply an operation to a single group, the command,

GROUP

is used. The response will be an integer from zero to a number no larger than the number of GETs which have been applied. If 0 is entered, then all groups are affected. If a non-zero number is used, then only that group is affected. An error is returned and the request repeated if a negative integer or a number which exceeds the maximum number of groups is entered.

DRAG

is similar to translate, but is done by a pointing device. When this command is invoked, a point to drag is specified. This is the initial point. Then a final or destination position is requested. The action is to move the appropriate vertices by this change. Essentially, it allows the user to specify a translation without knowing the actual coordinates of a map position on the screen. This program maintains the group designations internally whereas BUILD does not. The grouping is given on the third line (image) of the structures file.

The command

NEW

resets pointers and counters and is equivalent to restarting the program.

ACKNOWLEDGMENTS

As with any good software product, many people have contributed to the development of improvement of the package over several years. Special thanks to Jay Boris and Richard Peacock who wrote routines which appear here, and to the many users who have suffered through initial releases of the software, making sometimes pointed but useful comments on improvements which were needed, and usually made. Also to the numerous readers who caught poor explanations and unjustified assumptions.

APPENDIX A

The following is a description of each routine, together with its arguments. All routines are listed, although emphasis is on the higher level routines. For the "hackers", low level routines are included since special effects are sometimes desirable. The usual FORTRAN convention for integer and real (floating) numbers is maintained. Type is given only for arrays and character variables.

ALABEL

Purpose: to draw a character string with an aspect ratio of 4/3 given the width and an angle of rotation.

Usage: Call ALABEL (CHARS, X1, Y1, X2, Y2, ANGLE)

Description of Parameters:

CHARS - CHARACTER * 1 ARRAY	- character string to be drawn
X1, Y1	- lower left starting point of the string
X2	- X coordinate of the right ending point of the string
Y2	- not used
ANGLE	- angle (radians) by which to rotate the string. [A positive (negative) angle causes a counter-clockwise (clockwise) rotation about the lower left starting point.

Control Sequences: See Appendix B

Method: This routine calls the subroutine WDCOUNT to determine if the string contains characters. If characters exist, LABEL calculates the space size of the characters in order to create a 4/3 aspect ratio. Finally, the subroutine WDDRAW is called to plot the text.

BOXPLT

Purpose: to draw a rectangle given the endpoints of one of the diagonals.

Usage: Call BOXPLT (X1, Y1, X2, Y2)

Description of Parameters:

X1, Y1 - coordinates of lower left corner of the box

X2, Y2 - coordinates of the upper right corner of the box

Method: The routine determines the endpoints of each side of the rectangle. These endpoints are placed in an array and sent to the subroutine LINES which plots the four sides.

CHPLOT

Purpose: to draw a specified hardware character centered at selected points in an array.

Usage: Call CHPLOT (X, Y, CHAR, I1, I2, I3)

Description of Parameters:

X, Y -	ARRAY	- points at which to plot the character
CHAR -	CHARACTER * 1	- character to be plotted
I1		- index of first point to plot
I2		- increment at which points are to be selected to plot
I3		- index of last point to plot

Method: The routine selects the points at which to plot the character. The subroutine SYMBOL is called to draw the character at each of these chosen points.

CHRSET

Purpose: to change the default character set

Usage: Call CHRSET (N)

Description of Parameters:

N - number (1-24) corresponding to a character set

CHRSIZ

Purpose: to set the size of the hardware character

Usage: Call CHRSIZ (CHFRZ, GCHFRZ)

Description of Parameters:

CHFRZ - a fraction specifying the size of the characters relative to the screen (default = .03)

GCHFRZ - a fraction specifying the size of the characters relative to the variables Y1R and Y2R in subroutine GRAFIT (default = .04)

CIRCLE

Purpose: to draw a circle of a given radius about a specified center point.

Usage: Call CIRCLE (X,Y,R)

Description of Parameters:

X,Y - center of the circle

R - radius of the circle

Method: The routine generates commands to transform the points from the user space to the raster space. It then generates commands to plot a circle centered at the point (X,Y) with a radius of R.

COLOR

Purpose: to define the color for drawing

Usage: Call COLOR(N)

Description of Parameters:

N - the number corresponding to a desired color

Method: This command is ignored for the printer and Tektronix. For the other devices, an integer is normalized between 1 and NUMCOLOR, where NUMCOLOR is the number of colors the device can display. The Calcomp and Lexidata possess 4 and 16 colors, respectively. The numbers are normalized by modular arithmetic and numbers less than 1 are set equal to 1:

```
COLOR = MOD(N-1, NUMCOLOR) + 1
COLOR = MAXØ(COLOR, 1)
```

CONTUR

Purpose: to perform surface plotting

Usage: Call CONTUR (F, TEST, NX, NY, FL)

Description of Parameters:

F - ARRAY (NX, NY) - real values of the function to be contoured

TEST - ARRAY (NX, NY) - user supplied scratch array having the same dimension as F

NX - range and dimension of i in F(i,j)

NY - range and dimension of j in F(i,j)

FL - value of F(i,j) for contouring

Method: Contours are plotted by looking at a projection of the function F(I,J) and determining if the function crosses the contouring interval within their box. Six interpolations are done. First I to I+1, then J to J+1 parallel crossings are considered. Finally, the four cases of diagonal crossings are considered by triangular interpolation of each tesselation of the box by the corner points with the center. The corner points are considered in pairs moving counter-clockwise around the projected box.

DEFINE

Purpose: to define the correspondence between the user's window and the device's world space

Usage: Call DEFINE (X1, Y1, X2, Y2)

Description of Parameters:

X1, Y1 - coordinates of the window's lower left vertex

X2, Y2 - coordinates of the window's upper right vertex

Method: The differences X2-X1 and Y2-Y1 are checked. If either of the differences is equal to zero, the difference is set equal to one. The routine then calculates the transformation which maps objects from the user's window to the device's world space. The viewport defaults to the entire viewing surface.

DELAY

Purpose: to delay for hard copy units

Usage: Call DELAY(N)

Description of Parameters:

N - the number of seconds to delay

Method: This routine calls the system routing WAIT for the Tektronix and the Matrix camera.

DEVICE

Purpose: to initialize a device

Usage: Call DEVICE(N)

Description of Parameters:

N - defines a device

= 1 - Calcomp

= 2 - Printer

= 3 - Lexidata

= 4 - Textronix Console

= 5 - Empty Slot

Method: If a device is already open, execution of the program is terminated. Otherwise, the desired (device) file is connected to a unit. The device defaults to the printer if the specified number "n" does not correspond to an existing device.

DEVINP

Purpose: to get a coordinate triplet from an input device.

Usage: CALL DEVINP (INPUT, MASK, STATUS, X, Y, Z)

Description of Parameters:

INPUT - input device: 1 = joystick; 2 = tabet; 3 = keyboard.

MASK - button select - by power of 2.

1 = button 1

2 = button 2

4 = button 3

8 = button 4

STATUS - returns a value corresponding to MASK for the button pushed.
X,Y,Z - cordinate values returned. For two-dimensional devices, Z is always 0.

ENDFRM

Purpose: to close the graphics device

Usage: Call ENDFRM

Method: Closes logical units associate with DEVICE. Normally these are units 7, 8, and 9. This routine does not empty the I/O buffer but does send disconnect sequences and, for screen devices, erases the screen.

ENUMBR

Purpose: to draw a real decimal number expressed with powers of 10 such that the mantissa is between 1 and 10, i.e., $y*10^n$ where $1 < |y| < 10$

Usage: Call ENUMBR (XNUMBR, X1, Y1, X2, Y2)

Description of Parameters:

XNUMBR - number to be plotted in exponential format

X1, Y1 - starting point (lower left corner) of the first character drawn

X2, Y2 - terminating point (upper right corner) of the last character drawn

Method: A real number, not equal to zero, is normalized between 0 and 10. Then FNUMBR plots this portion as scaled by the starting and terminating points (X1, Y1) and (X2, Y2). The characters "10*" are then plotted, followed by a superscript containing the normalizing power of 10.

ERASE

Purpose: to erase the screen

Usage: Call ERASE

Method: This command is ignored for the Calcomp. For display devices a simple erase sequence is sent.

FILTYP

Purpose: to define the appearance of the interior of polygons and circles

Usage: Call FILTYP(N)

Description of Parameters:

N - the number corresponding to a desired fill type

Method: There are 9 fill types (0-8) for devices supporting filling patterns. The current devices, except for the Calcomp, support these patterns. Figure (4) shows the fill pattern numbered from 0 on the lower left to 8 at the upper right of the figure.

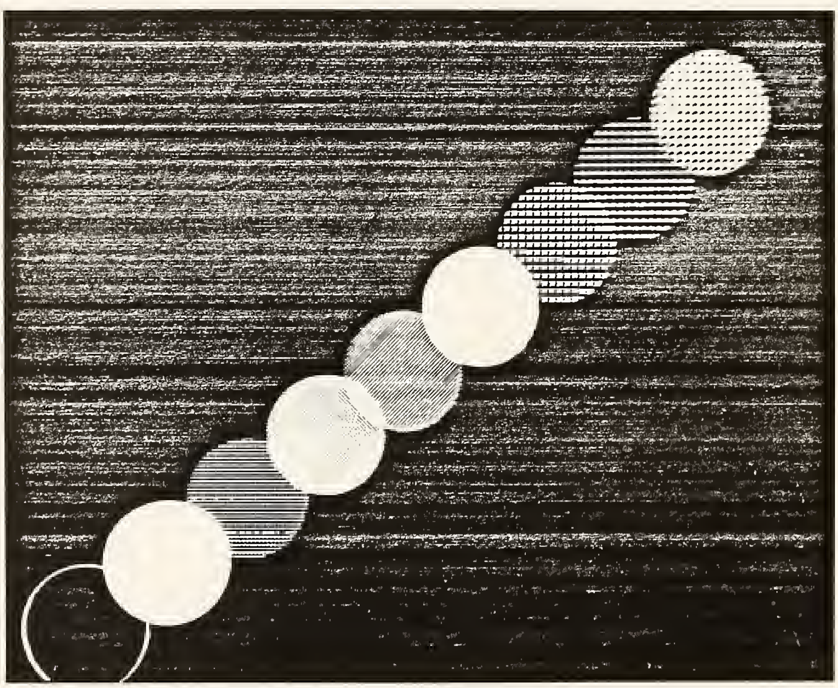


FIGURE 4

FNUMBR

Purpose: to draw a real number in FORTRAN F format

Usage: Call FNUMBR (X, Y, DX, DY, SX, SXY, SY, XNUMBR, IWIDTH, NDIGIT)

Description of Parameters:

X, Y - starting point (lower left corner) of the first character to be drawn

DX - increment added to the X-coordinate for each character drawn

DY - increment added to the Y-coordinate for each character drawn

SX - X space size for the characters

SXY - slant modifier for characters

SY - Y space size for the characters

XNUMBR - number to be plotted in F format

IWIDTH - total width of field including decimal point

NDIGIT - number of places to the right of the decimal point to be drawn

Method: The input "XNUMBR" is scaled to include the specified number of places to the right of the decimal point. Each digit in the number is converted to a literal character. Leading zeros are converted to blanks and the decimal point is inserted if NDIGIT is greater than zero. The subroutine WDDRAW is called to plot the literal data string. The parameters X, Y, DX, DY, SX, SXY, and SY have the same meaning as in WDDRAW.

FRAME

Purpose: to force out the contents of the buffer

Usage: Call FRAME

Method: This call should be used to terminate each series of graphics sequences. It is used to insure that the I/O buffers are empty and to synchronize the timing of the I/O channel.

GRAFIT

Purpose: to set up a graph (X and Y axis) for plotting data

Usage: Call GRAFIT (NPLT, X1, X2, X1R, X2R, XX1, XX2, Y1, Y2, Y1R, Y2R, YY1, YY2, XTIT, NDVX, YTIT, NDVY)

Description of Parameters:

NPLT	- number of the plot referenced (up to four graphs may be placed on one graph)
X1,X2	- labels of X-axis minimum and maximum, respectively. (The literal string 'NONE' produces no label.)
X1R,X2R	- user space value of X-axis minimum and maximum, respectively
XX1,XX2	- minimum and maximum values, respectively, of X in data to be plotted
Y1, Y2	- labels of Y-axis minimum and maximum, respectively. (The literal string 'NONE' produces no label.)
Y1R,Y2R	- user space value of y-axis minimum and maximum, respectively
YY1,YY2	- minimum and maximum values, respectively, of Y in data to be plotted.
XTIT - char	- title for X-axis (less than 30 characters and terminated by '|.')
NDVX	- number of intervals to be drawn on X-Axis
YTIT - char	- title for Y-axis (less than 30 characters and terminated by '|.')
NDVY	- number of intervals to be drawn on Y-axis

There are three additional entry points: PLOTCH, PLOTLN and GRISET.

PLOTCH (NPLT, X, Y, NP CHAR)	- where NPLT is the corresponding plot number
PLOTLN (NPLT, X, Y, NP)	- (See above), X and Y are coordinate arrays, NP is the number of points to plot (or connect) and CHAR is a character in CHARACTER *1 format
GRISET (XL, YL, XR, YT)	- see DEFINE

HDCOPY

Purpose: to generate a hard copy

Usage: Call HDCOPY

Device	Associated Hard Copy
Calcomp	-
Printer	-
Lexidata	Camera
Tektronix	Activates screen copy unit

Method: This command is ignored for the Calcomp and the printer.

HHDRAW

Purpose: to draw a particular character of a specified character set

Usage: Call HHDRAW (X, Y, SX, SXY, SY, ICHAR, NSET, IERR)

Description of Parameters:

- X, Y - starting point (lower left) of the character to be plotted
- SX - X space size for character (user coordinate system)
- SXY - slant modifier for character
- SY - Y space size for character (user coordinate system)
- ICHAR - index to specify characters within the chosen set
- NSET - number specifying a particular character set
- IERR - error flag

  = 0 - no errors
  = 1 - error in accessing set or character

Method: If the desired character is from the hardware set, then the subroutine SYMBOL is called to plot the character. For characters from other sets, HSETS is called to place the character's coordinates and pen values in arrays and then LINES is called to plot the character as a set of connected strokes.

IOWAIT

Purpose: to put a pause in the program

Usage: Call IOWAIT (N)

Description of Parameters:

N - number of milliseconds to pause

Method: This routine calls the system routine WAIT.

LABEL

Purpose: to draw a character string of a specified width and height (i.e., a rectangular area) at a specified location

Usage: Call LABEL (CHARS, X1, Y1, X2, Y2, ANGLE)

Description of Parameters:

CHARS - CHARACTER * 1 ARRAY	-	character string to be drawn
X1, Y1	-	lower left starting point of string
X2, Y2	-	upper right ending point of string
ANGLE	-	angle (radians) by which to rotate the string. [A positive (negative) angle causes a counter-clockwise (clockwise) rotation about the point (X1, Y1)].

Control Sequences: See Appendix B

Method: This routine calls the subroutine WDCOUNT to determine if the string contains characters. If characters exist, LABEL calculates the space size of the characters and calls the subroutine WDDRAW to plot the text.

LINE

Purpose: to draw a line between two given points

Usage: Call LINE (X1, Y1, X2, Y2)

Description of Parameters:

X1, Y1 - coordinates of first point

X2, Y2 - coordinates of second point

Method: The routine transforms the points from the user space to the raster space. Subroutine PUTLNV is then called to generate a line segment to connect the points.

LINES

Purpose: to draw lines between points in an array

Usage: Call LINES (X1, Y1, X2, Y2,N)

Description of Parameters:

X1, Y1-array - starting coordinates of the lines

X2, Y2-array - ending coordinates of the lines

N - number of line pairs: [X1(i), Y1(i)] → [X2(i), Y2(i)]

Method: The routine transforms the points from the user space to the raster space. Subroutine PUTLNV is then called to generate line segments to connect corresponding points.

LINWID

Purpose: to define the line width for plotting

Usage: Call LINWID(N)

Description of Parameters:

N - the number of strokes when drawing a line - not implemented for CALCOMP

Method: This routine sets the number of strokes for each line. (The strokes are drawn one pixel apart.)

LNPLOT

Purpose: to draw lines between selected points of an array

Usage: Call LNPLOT (X, Y, I1, I2, I3)

Description of Parameters:

X - array - X coordinates of data

Y - array - Y coordinates of data

I1 - index of first point to be connected

I2 - increment at which points are to be selected to plot

I3 - index of the last point in the data array

Method: The selected points are stored in new arrays and the subroutine LINES is called to generate each of the lines.

NEWFRM

Purpose: to erase the screen and initialize a new frame

Usage: Call NEWFRM

Method: Device dependent but in general clears buffers and initializes pointers.

PLYGON

Purpose: to draw a closed polygon

Usage: Call PLYGON (X,Y,N)

Description of Parameters:

X - array - X coordinates of the data
Y - array - Y coordinates of the data
N - number of vertices in the polygon

Method: The routine generates commands to transform coordinates from the user space to the raster space. It then generates commands to plot the polygon starting and ending at vertex (X(1), Y(1)).

SCALNG

Purpose: to define the window and the viewport

Usage: Call SCALNG (X1, Y1, X2, Y2, X1H, Y1H, X2H, Y2H, X1S, Y1S, X2S, Y2S)

Description of Parameters:

- X1, Y1 - lower left corner of the window (user coordinates)
- X2, Y2 - upper right corner of the window (user coordinates)
- X1H, Y1H - lower left corner of viewport (device coordinates)
- X2H, Y2H - upper right corner of viewport (device coordinates)
- X1S, Y1S - lower left corner of world (world coordinates)
- X2S, Y2S - upper right corner of world (world coordinates)

Method: Calculates the transformations to map objects from the user's window into the device's world and then into the viewport.

SETDEV

Purpose: to change logical units

Usage: Call SETDEV (N1, N2)

Description of Parameters:

N1 - logical unit for graphics output (0- ; default = 7)

N2 - logical unit for diagnostic output (1- ; default = 0)

Default Device Assignments:

Type	LU	Result
Diagnostics	0	no output
Graphics	7	normal - assigned
Camera*	8	assigned with Lexidata
Character set	0	unit zero-assigned
Calcomp	L7:	RS232 line
Printer	PR:	whatever
Lexidata	LEX:	DMA (L34DVR in system)
Matrix camera	L14:	RS232
Tektronix	C:	console
Open slot	NULL:	bit bucket
Console input	5	
Character set	9 (closed then opened)	
Tablet input*	8 (closed then opened)	

*Note: Camera and tablet cannot be active simultaneously

SETLUT

Purpose: to set color look-up table

Usage: Call SETLUT

Method: The default color look-up table is shown in Fig. (5) for the Lexidata 8100 . SETLUT is not currently used. It exists for users who must define look-up tables on other systems.

FIGURE 5

Purpose: construct and plot data surface

Usage: Call SURFAC (Z, NX, NY, MODE)

Description of Parameters:

Z - ARRAY (NX, NY)	- data to be plotted as a surface
NX	- dimension and range of i in Z(i,j)
NY	- dimension and range of j in Z(i,j)
MODE	- specifies the surface to plot MODE = 1: upper surface MODE = -1: lower surface

This subroutine and its entries construct and display plots of a surface function (two dimensional) with the hidden lines removed. The appearance of the plot is quite flexible and can include skirts around unobservable portions. It is possible to show just the upper or just the lower surface, or both together. The surface must be approximately horizontal.

SFRAME (MODE)	- shows rectangular parallelpiped plotting region.
SSKIRT (Z,NX,NY,MODE)	- draw skirts around the plotting region (See figure 6) - uses hidden surface algorithm, so should be called last
SRFSET (X,Y,Z MIN, Z MAX, NX, NY)	- initiates the plotting region

THE GEOMETRY FOR "SURFAC" PLOTS

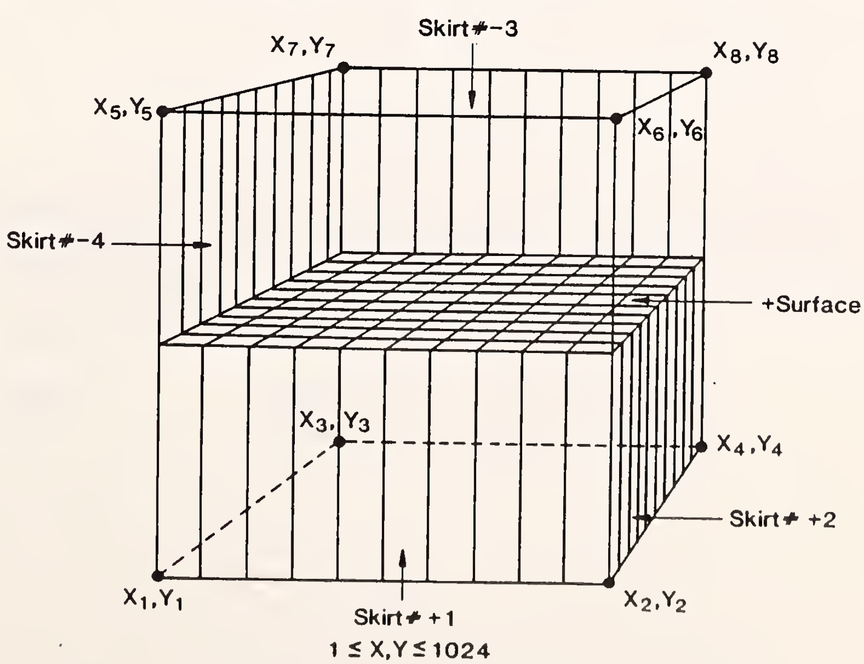


FIGURE 6

SYMBOL

Purpose: to draw a specified hardware character centred at a given point

Usage: Call SYMBOL (X, Y, CHAR)

Description of Parameters:

X, Y - location at which to plot the character

CHAR - CHARACTER*1 - character to be plotted

Method: The routine transforms the center point from the user space to the raster space. Subroutine PUTCH is then called to generate the commands to plot the character.

VOLUME

Purpose: To draw a two-dimension projection of a 3-dimensional array; simulates effect of "3D"

Usage: Call VOLUME (MODE, F, NX, NY, NZ, CLEVE, NCL, T, NT)

Description of Parameters:

F - ARRAY (NX, NY, NZ)	- three-dimensional figure to be plotted
NX	- range and dimension of i in F(i,j,k)
NY	- range and dimension of j in F(i,j,k)
NZ	- range and dimension of k in F(i,j,k)
T - ARRAY (NT, NT)	- plotting array (boolean) to eliminate hidden lines
NT	- dimension of the array T: T(NT,NT)
CLEVE	- contour surface level for plotting
MODE	- specifies the surface to be plotted
ABS(MODE) = 1:	plot contour level "CLEVE" in each plane; fix hidden line matrix, T
ABS(MODE) = 2:	find hidden line matrix, T, without plotting
MODE < 0:	"outside" is less than "CLEVE"
MODE > 0:	used when the value of the function on the "outside" is greater than "CLEVE"
NCL	- number of pairs of (MODE, CLEVE) to be plotted

Method: Contours a Function F(See above) in three dimensions and projects the resultant plots onto a two dimensional plotting surface. Auxiliary entries are

VOLSET (X,Y,T,NT)	- initialize the plot
VOLFRM (MODE)	- MODE = 0 = corner vertices - Mode = 1 = surrounding box

A three dimensional interpolation by triangular tesselation is used to find the contour crossing points. The technqiue is similar to that used in CONTUR. Once again, the hidden pixel (frame buffer) is filled as the figure is scanned back to front. A sample showing two overlapping spheroids and a cylinder is shown in Fig. 7. The functions plotted are

Cylinders: $F = (X-10.5)^2 + (Y-3.5)^2$

Sphere:

$$F_1 = \frac{10}{(X-8)^2 + (Y-10.5)^2 + (Z-10.5)^2}$$

$$F_2 = \frac{20}{(X-13)^2 + (Y-10.5)^2 + (Z-10.5)^2}$$

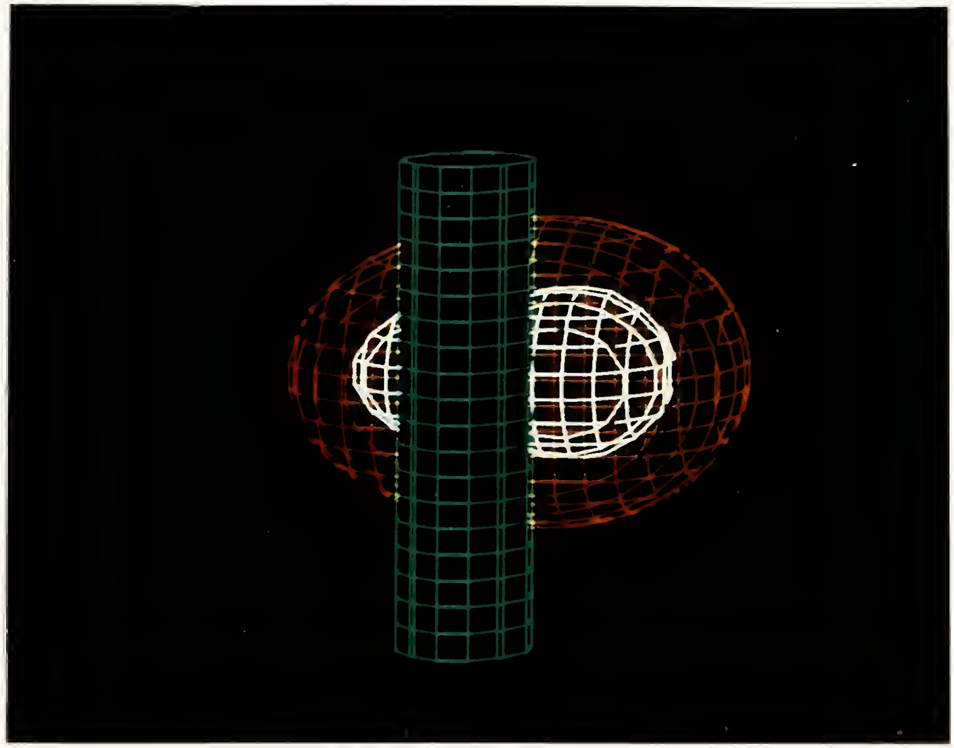


FIGURE 7

WDDRAW

Purpose: to draw a character string

Usage: Call WDDRAW (X, Y, DX, DY, SX, SXY, SY, CHARS)

Description of Parameters:

X, Y	- starting point of the character string (the string is plotted to the right of (X,Y) unless otherwise indicated by a control character)
DX	- increment to be added to the X-coordinate for each character drawn
DY	- increment to be added to the Y-coordinate for each character drawn
SX	- X space size for characters (user coordinate system)
SXY	- slant modifier for characters
SY	- Y space size for characters (user coordinate system)

CHARS - CHARACTER * 1 ARRAY - character string to be drawn

Control Sequences - See Appendix B

Method: This routine scans the string for control characters and text characters. When a text character is found, the corresponding character number is obtained by the function IDCHAR. The set number assumes the default value unless the number was changed by a control character within the string. The routine also executes scaling, shifting, and rotating transformations in order to determine the starting positions of the characters. Finally, the subroutine HHDRAW is called to plot the text characters.

APPENDIX B

Control Sequences for WDDRAW, LABEL, and ALABEL

The control sequence is "|" followed by an editing character which is one of the following:

L - justify the text to the left

M - center the text

R - justify the text to the right (default)

Hnn - change from default character set to set nn

U - draw the following characters in superscript

D - draw the following characters in subscript

O - reset character size and placement from superscript or subscript to original size

B - backspace over last character drawn (works only for one character, multiple backspaces will produce unpredictable results.)

Cnn - change from the default color to color nn

. - end of character string

APPENDIX C

Set Number	Alphabet	Style	Size
1	Roman (Hardware characters)		
2	Roman		Cartographic
3	Greek		Cartographic
4	Roman	Simplex	Print
5	Greek	Simplex	Print
6	Script	Simplex	Print
7	Roman	Complex	Index
8	Greek	Complex	Index
9	Italic	Complex	Index
10	Roman	Complex	Print
11	Greek	Complex	Print
12	Italic	Complex	Print
13	Script	Complex	Print
14	Roman	Duplex	Print
15	Roman	Triplex	Print
16	Italic	Triplex	Print
17	German	Gothic	Print
18	English	Gothic	Print
19	Italian	Gothic	Print
20	Cyrillic	Complex	Print
21	Miscellaneous		
22	Miscellaneous		
23	Miscellaneous		
24	Miscellaneous		

Set Number 1

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n
o	p	q	r	s	t	u	v	w	x
y	z		%	&	@	$	#	.	,
:	;	!	?	-	+	=	*	/	(
)	[	]	{	}	¦	"	<	>	°
'		¬	.	_					

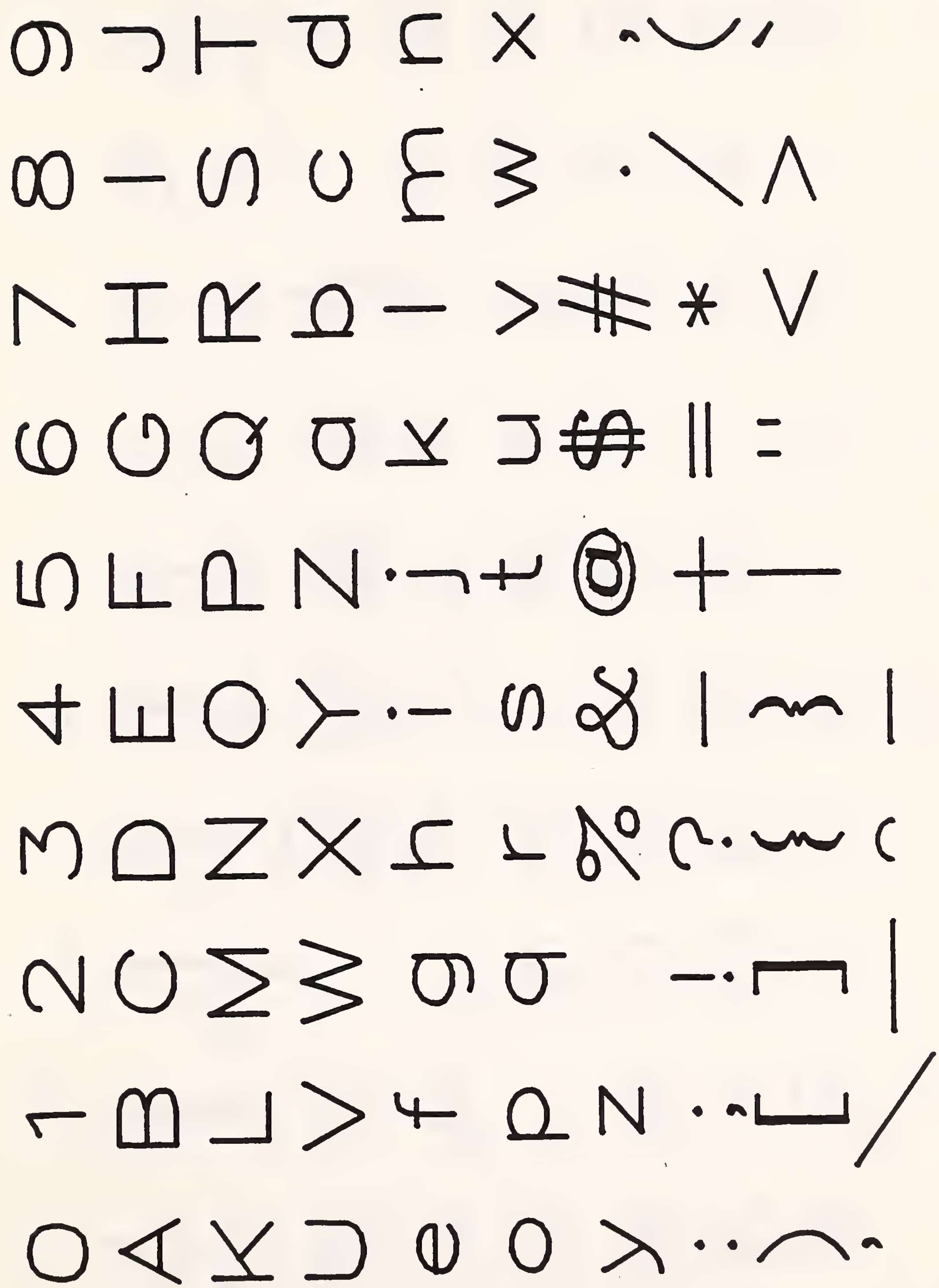
Set Number 2

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z			&	
$	#	.	,	:	;	!	?	—	+
=	*	/	(	)						
"									'
'	→	'	°		⌑		×	·	

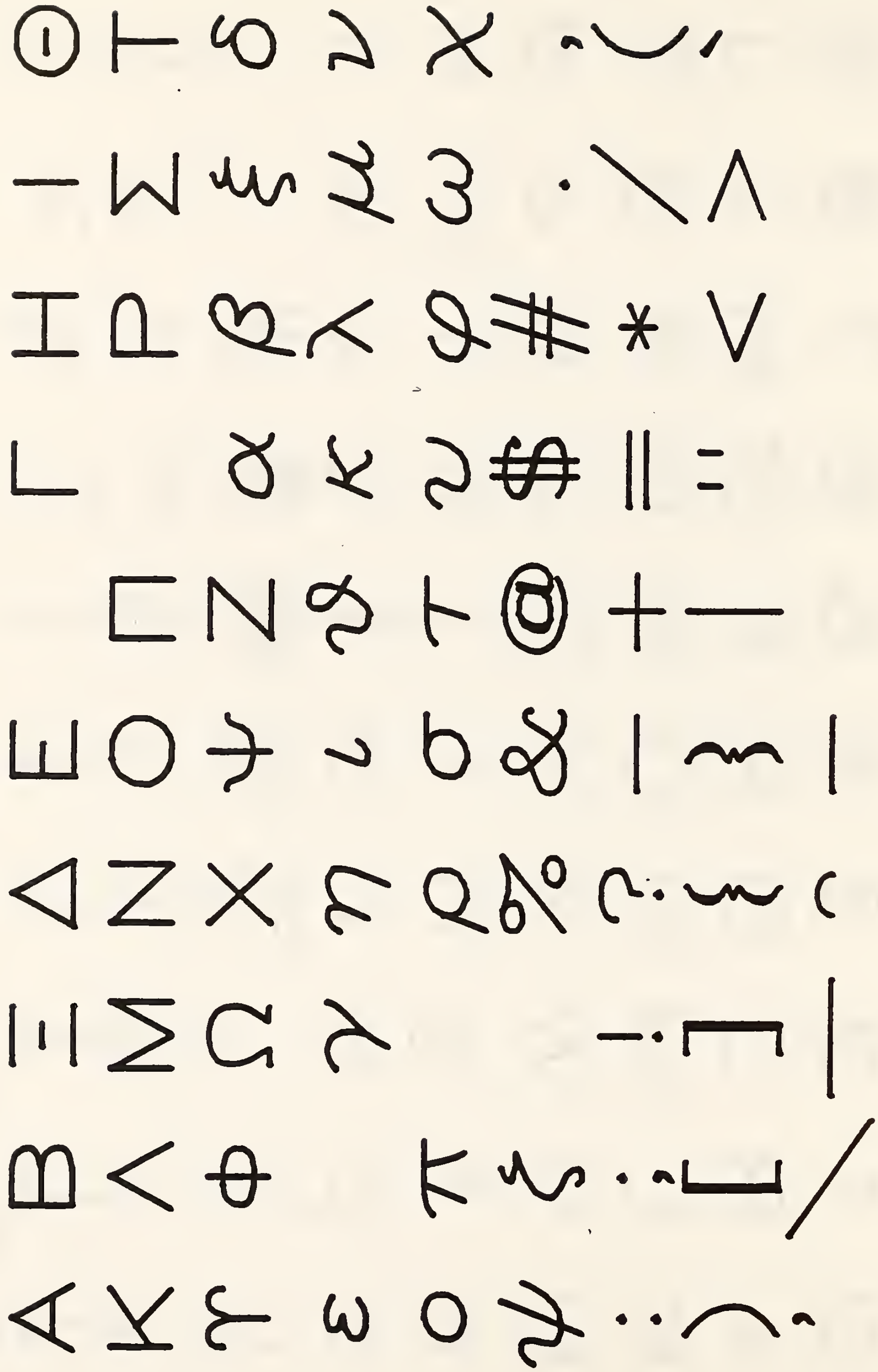
Set Number 3

Α	Β	Ξ	Δ	Ε		Γ	Η	Ι	Θ
Κ	Λ	Μ	Ν	Ο	Π		Ρ	Σ	Τ
ϒ	Φ	Ω	Χ	Ψ	Ζ				

Set Number 4



Set Number 5




A B C D E F G H I J
K L M N O P Q R S T
U V W X Y Z a b c d
e f g h i j k l m n
o p q r s t u v w x
y z % & @ $ # . ,
: ; ! ? — + = * / (
) [ ] { } | " < > `
, \ _ ^ —

Set Number 7

0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? — + = * / (

) [ ] { } | " < > `

’ \ ___ ⌒ –

Set Number 8

Α	Β	Ξ	Δ	Ε		Γ	Η	Ι	Θ
Κ	Λ	Μ	Ν	Ο	Π		Ρ	Σ	Τ
Υ	Φ	Ω	Χ	Ψ	Ζ	α	β	ξ	δ
ε		γ	η	ι	ϑ	κ	λ	μ	ν
ο	π		ρ	σ	τ	υ	φ	ω	χ
ψ	ζ								

Set Number 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? — + = * / (

) [ ] { } | " < > `

' \ ___ ⌒ –

Set Number 10

0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? - + = * / (

) [ ] { } | " < > `

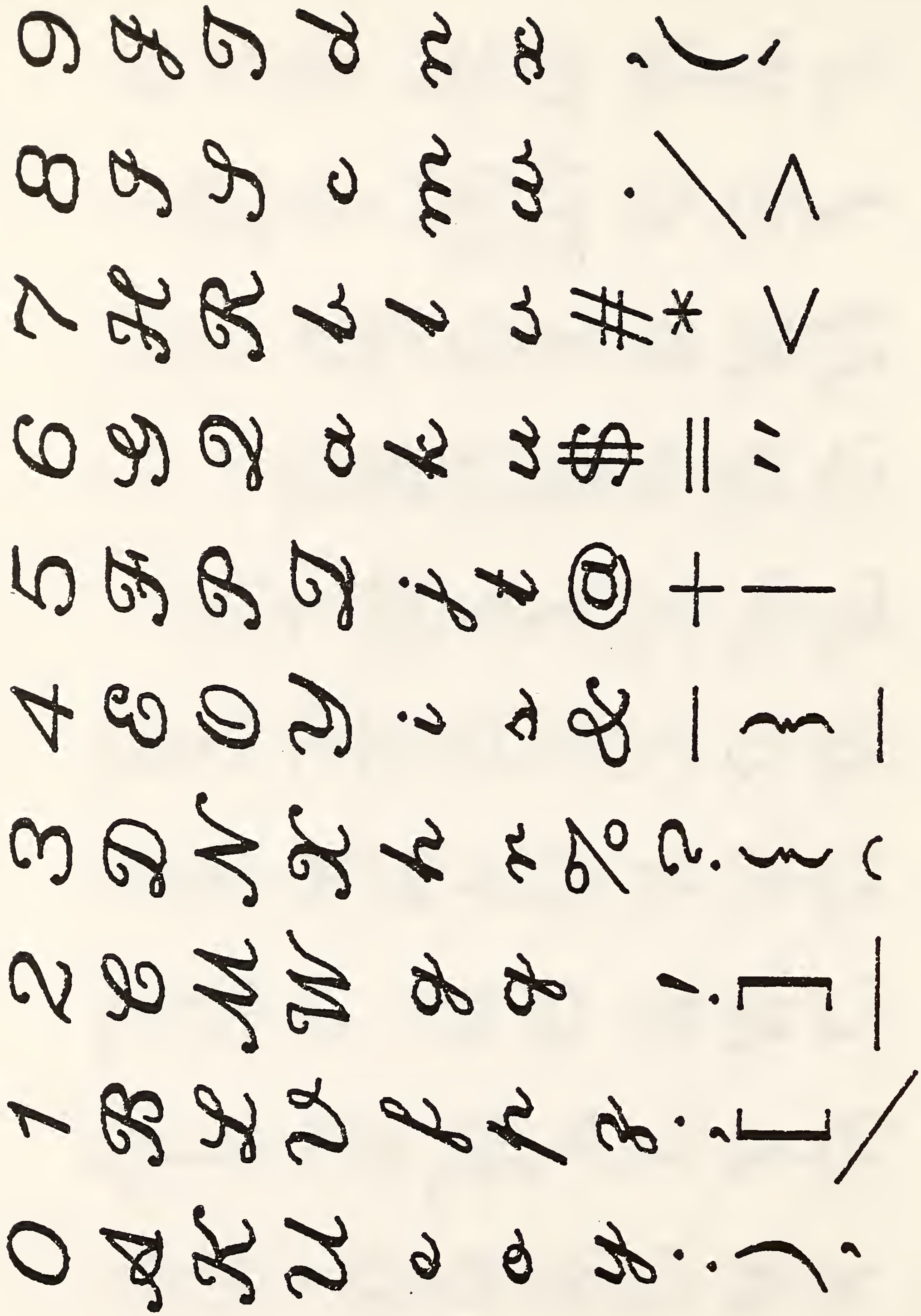
, \ _ ⌢ —

Α	Β	Ξ	Δ	Ε		Γ	Η	Ι	Θ
Κ	Λ	Μ	Ν	Ο	Π		Ρ	Σ	Τ
ϒ	Φ	Ω	Χ	Ψ	Ζ	α	β	ξ	δ
ε		γ	η	ι	ϑ	κ	λ	μ	ν
ο	π		ρ	σ	τ	υ	φ	ω	χ
ψ	ζ		%	&	@	$	#	.	,
:	;	!	?	—	+	=	*	/	(
)	[	]	{	}	|	"	<	>	`
,	\	___	⌢	—					

Set Number 12

A B C D E F G H I J
K L M N O P Q R S T
U V W X Y Z a b c d
e f g h i j k l m n
o p q r s t u v w x
y z % & @ $ # . ,
: ; ! ? — + = * / (
) [ ] { } | " < > `
, \ _ ⌒ —

Set Number 13



0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? – + = * / (

) [ ] { } | " < > `

, \ _ ⌒ _

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n
o	p	q	r	s	t	u	v	w	x
y	z		%	&	@	$	#	.	,
:	;	!	?	–	+	=	*	/	(
)	[	]	{	}	|	"	<	>	`
,	\	_	⌢	—					

0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? — + = * / (

) [ ] { } | ″ < >

, \ _ ⁀ _

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n
o	p	q	r	ſ	t	u	v	w	x
y	z		%	&	@	$	#	.	,
:	;	!	?	—	+	=	*	/	(
)	[	]	{	}	|	"	<	>	`
,	\	s	ß	tʒ					

0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

: ; ! ? - + = * / (

) [ ] { } | " < > `

' \ _ ^ _

Set Number 19

0 1 2 3 4 5 6 7 8 9

A B C D E F G H I J

K L M N O P Q R S T

U V W X Y Z a b c d

e f g h i j k l m n

o p q r s t u v w x

y z % & @ $ # . ,

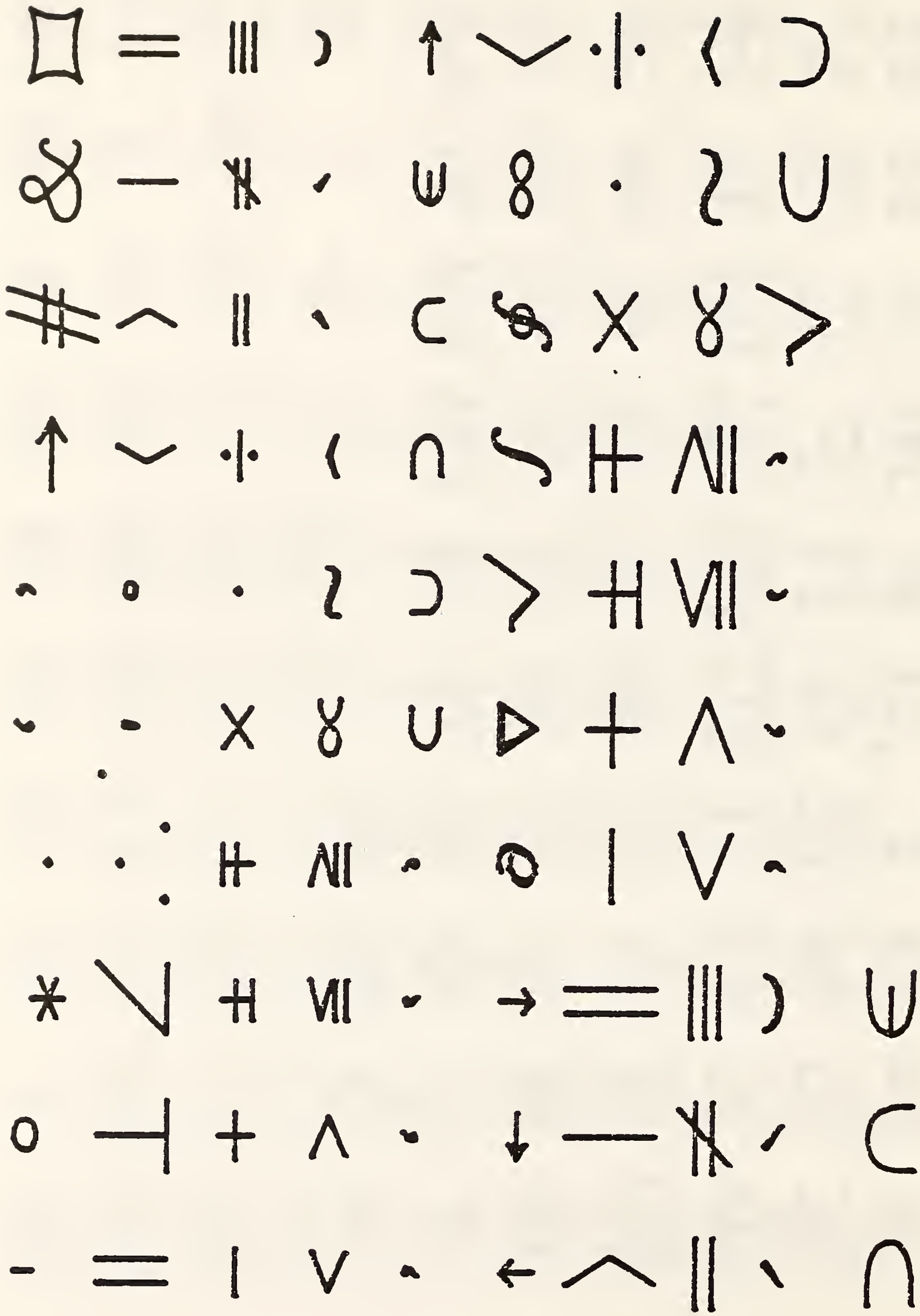
: ; ! ? - + = * / (

) [ ] { } | " < > `

' \ _ ^ -

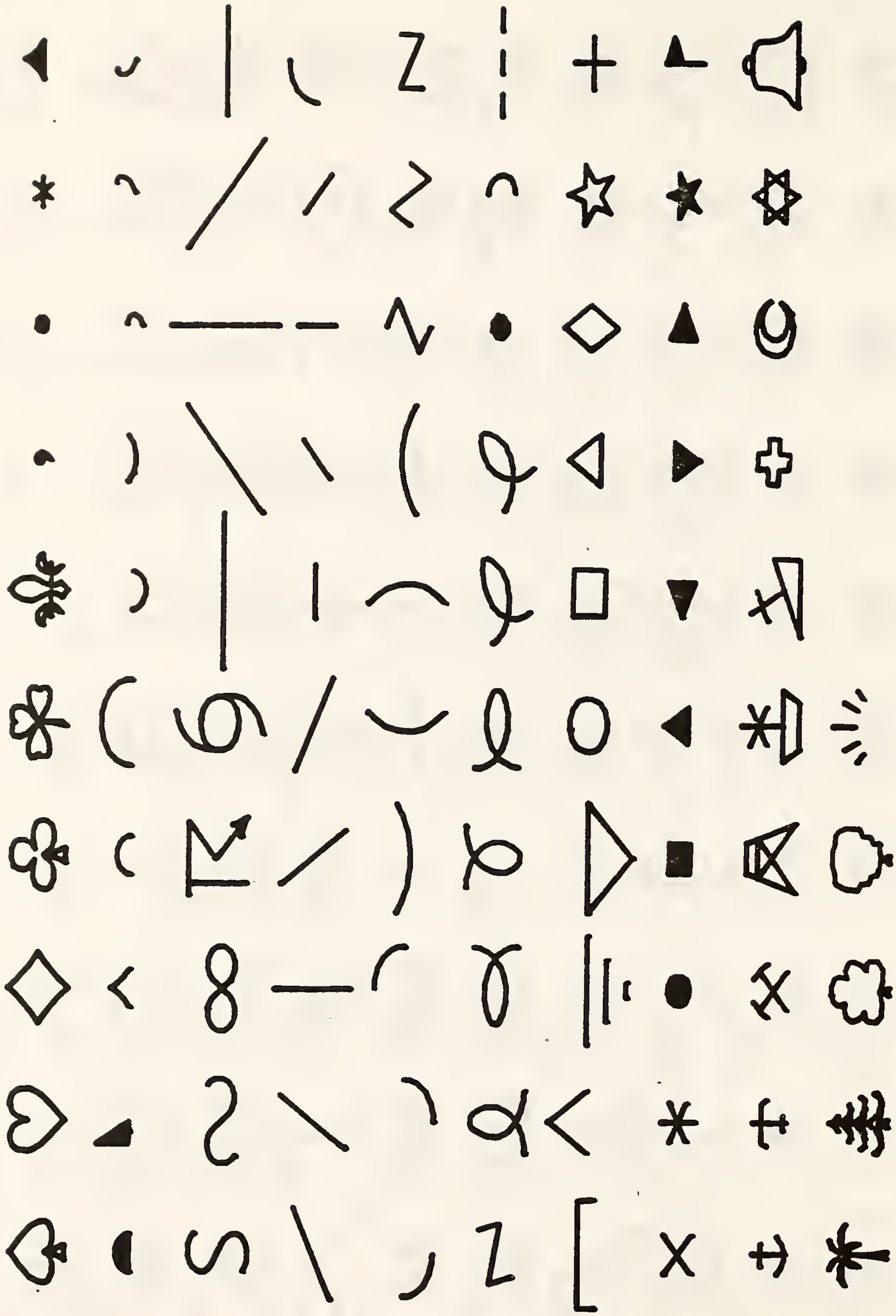
А Б В Г Д Е Ж З И Й
К Л М Н О П Р С Т У
Ф Х Ц Ч· Ш Щ Ъ Ы Ь Э
Ю Я а б в г д е ж з
и й к л м н о п р с
т у ф х ц ч ш щ ъ ы
ь э ю я

Set Number 21

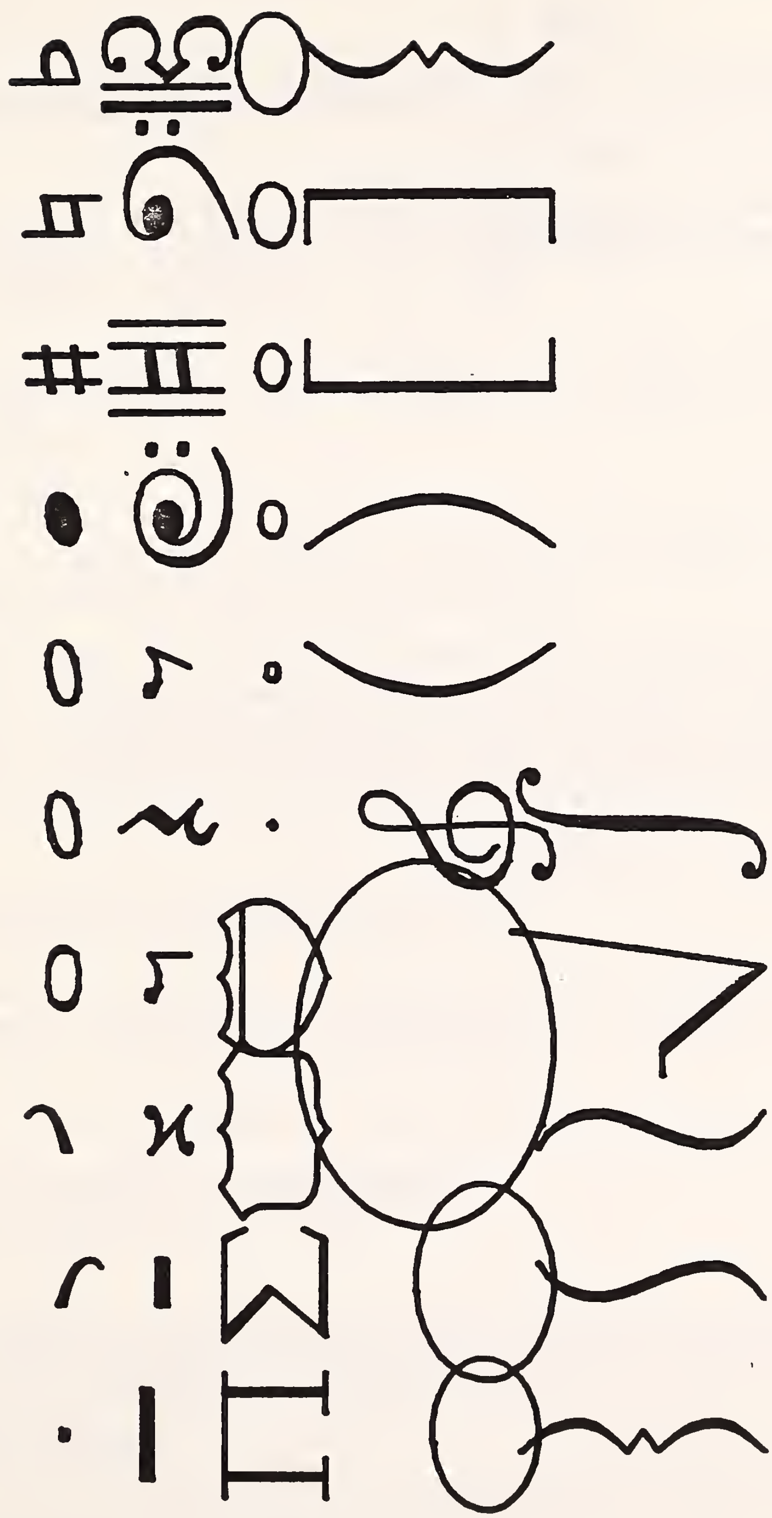


Set Number 22

Set Number 23



Set Number 24



APPENDIX D

The following is a description of each BUILD command. The prompts associated with each command and the format of the user's response are indicated.

Note: When specifying a command, at least two letters of each word must be entered (e.g. AD PO means ADD POLYGON).

1. ADD

Elements are added to the arrays.

1.1 ADD VERTEX

```
ENTER (X,Y,Z) COORDINATES AND SELECT BUTTON
     BUTTON 1  (KEYBOARD = '1')-- ENTER VERTEX
     BUTTON 2  (KEYBOARD = '2')-- QUIT
```

If the input device is the joystick or the tablet, the vertex is selected with the picking implement and then button 1 is pressed. If the keyboard is used, the prompt

```
X,Y,Z,IDV=>
```

appears. Three real numbers, followed by the number 1, are entered. These numbers are separated by commas or spaces.

In both cases, the X,Y,Z coordinates of the vertex are stored in the VERTEX array. Vertices are added until the VERTEX array is filled or the user wants to quit. In order to quit, button 2 is pressed. At the keyboard, any three numbers, followed by the number 2, will terminate the process. (Alternatively, three commas followed by the number 2 can be entered: ,,,2)

1.2 ADD EDGE

```
BY VERTEX NO. (1) OR COORDINATES (2)?
```

An integer is entered. The number 1 is entered if the edges are to be composed of vertices in the VERTEX array. The number 2 is entered if the edges will be created from new vertices.

If the first method is chosen, there will be the prompt

```
ENTER ENDPOINTS -- 2 VERTEX NUMBERS
TO QUIT HIT <RETURN> OR ENTER 'QUIT'
VETEX NUMBERS:
```

Two integers, separated by a comma or space, ar entered. These numbers, indicating the vertices which define the edge, are stored in the EDGE array. The user is prompted for another edge by

```
VERTEX NUMBERS:
```

The process continues until the EDGE array is filled or the user has finished adding edges.

If the second method is selected (i.e. new vertices will be chosen), the user is prompted

```
ENTER 2 ENDPOINT COORDINATES:
     BUTTON 1 (KEYBOARD = '1') -- ENTER VERTEX FOR EDGE
     BUTTON 2 (KEYBOARD = '2') -- LAST VERTEX FOR EDGE
     BUTTON 3 (KEYBOARD = '4') -- QUIT
```

The vertices are specified as in ADD VERTEX. The first vertex of an edge is followed by button 1 ('1' on the keyboard) and the second vertex is followed by button 2 ('2' on the keyboard). The new vertices are added to the VERTEX array and the new edge is added to the EDGE array. Additional edges can be created until the VERTEX or EDGE array is filled or the user wants to quit. Button 3 ('4' on the keyboard) is used to terminate the process.

Note: In both cases, if a newly formed edge exists in the EDGE array, this element will not be stored again. A message will appear to inform the user of the condition.

1.3 ADD POLYGON

Polygons are added in the same manner as edges except that at least three vertices are needed to define a polygon. Also, there is a limit of NPVERT vertices for a polygon. Polygons can be created from existing vertices or from new vertices. When entering new vertices, button 1 ('1' on the keyboard) is pressed after each vertex except the last one in a polygon. This last vertex is followed by button 2 ('2' on the keyboard). Finally, button 3 ('4' on the keyboard) terminates the process.

New polygons are added to the POLY array provided that they don't already exist in the array. (Messages will appear to indicate existing polygons.) Any new vertices will be added to the VERTEX array.

2. ANIMATE

A sequence of frames is specified. This sequence is viewed with DISPLAY.

NUMBER OF FRAMES

An integer is entered to indicate the number of frames.

TOTAL TRANSLATION CHANGE (DX,DY,DZ)

Three real numbers separated by commas, are entered. These numbers indicate the total displacement of the model in the X,Y,Z directions.

TOTAL ROTATION CHANGE (DX,DY,DZ)

Three real numbers, separated by commas, are specified. The model can be rotated about a point (specified in the next step) with respect to X',Y',Z' axes. These axes pass through the specified point and are parallel to the X,Y,Z axes of the coordinate system. The three numbers entered indicate the angles (in degress) of rotation about the X',Y',Z' axes. Note that the rotations will be made in he X',Y',Z' order.

RELATIVE ORIGIN

The point about which rotations occur is specified. The center for the model is chosen by hitting the <RETURN> key. A different point is chosen by entering the appropriate X,Y,Z coordinates. The coordinates are entered as three real numbers separated by commas.

DISTANCE TO ORIGIN (DELTA)

A real number indicating the total change in the observer's position, is entered. A positive number moves the observer along the Z-axis in the negative direction (i.e. away from the viewing surface). A negative number causes the observer to move in the positive direction along the Z-axis.

SCALE FACTOR

A positive real number is entered. If the number is greater than one, the model is magnified. The model is reduced if the number is less than one.

MOVE EDGE(S)

There are several acceptable formats for the response:

```
n
n-m
n-
n BY X,Y,Z
n-m BY X,Y,Z
n- BY X,Y,Z
```

One edge (n) or a group of edges (n-m) may be moved. Note n- represents all the edges from n through the last element in the EDGE array. n and m are entered as positive integers. The relative displacement vector is specified by the real numbers X,Y,Z. If this displacement vector is not specified, the user is prompted for it.

MOVE POLYGON(S)

One polygon (n) or a group of polygons (n-m) may be moved by a distance X,Y,Z. The specifications are entered in the same format as for MOVE EDGE(S).

Note: Any of these transformations are omitted from the animated sequence by hitting <RETURN> after a particular prompt.

3. AUTO

The window and observer are moved so that the entire model appears in the field of view. This command is particularly useful if the location of the model outside the window becomes unknown to the user. New parameters are calculated by AUTO. A display does not automatically follow. The DISPLAY command must be used to view the model.

Note: The scaling of the model may be altered as a result of using AUTO. It may be necessary to compensate for this change by using the SCALE command.

4. COPY

The switch for hard copy of the graphics display is set.

4.1 COPY ON

NO. OF COPIES

An integer is entered to indicate the number of hard copies of each display to produce. This number defaults to one (1) if the <RETURN> key is pressed. Following each graphics display on the output device, hard copies, associated with this device, are generated.

4.2 COPY OFF

The automatic generation of hard copies after each graphics display is discontinued.

Note: At the start of program execution, the copy switch is set off.

5. DELETE

Elements are deleted from the arrays.

5.1 DELETE VERTEX

Vertices are not actually deleted from the VERTEX array. Instead, these "deleted" vertices are made unavailable for displaying and for constructing edges or polygons. Vertices are deleted from the array with the SQUEEZE command.

```
ENTER VERTEX LINE NUMBERS
HIT <RETURN> OR ENTER 'QUIT' WHEN DONE
VERTEX n=
```

There are three formats for entering vertices:

```
n
n-m
n-
```

where n and m are positive integers. One vertex (n) or a group of vertices (n-m) may be deleted. n- repesents all the vertices from n through the last element in the VERTEX array. Once these vertices are deleted, the user is prompted for other vertices to delete with

```
VERTEX n=
```

This process is repeated until all vertices are deleted or until the user has finished.

5.2 DELETE EDGE

```
ENTER LINE NUMBERS.
HIT <RETURN> OR ENTER  QUIT' WHEN DONE
LINE NUMBER(S):
```

One edge (n) or a group of edges (n-m) may be deleted. These specifications are entered in the same format as the DELETE VERTEX. The user is prompted for LINE NUMBER(S): until the EDGE array is empty or the user is done.

5.3 DELETE POLYGON

```
ENTER LINE NUMBERS.
HIT <RETURN> OR ENTER 'QUIT' WHEN DONE
LINE NUMBER(S):
```

One polygon (n) or a group of polygons (n-m) may be deleted. These specifications are entered in the same format as the DELETE VERTEX. The user is prompted for LINE NUMBERS: until the POLY array is empty or the user is finished.

6. DEVICE

At the begining of program execution, the output device for graphics display defaults to the Lexidata. DEVICE is used to select a new default device. The possible selections are

```
DEVICE    { CALCOMP  }        (default = LEXIDATA)
          { PRINTER  }
          { LEXIDATA }
          { CONSOLE  }
```

7. DISPLAY

Array elements are displayed in graphics mode.

7.1 DISPLAY VERTEX

Each vertex in the model is displayed and labelled with its vertex number.

7.2 DISPLAY EDGE

The model's edges are displayed.

7.3 DISPLAY POLYGON

The model's polygons are displayed.

7.4 DISPLAY [ALL]

All the edges and polygons in the model are displayed.

8. DUPLICATE

Elements are duplicated at other locations. The new elements are added to the array.

8.1 DUPLICATE EDGE

WHICH EDGE(S)?

The specifications for duplicating edges are entered in one of several ways:

```
n
n-m
n-
n BY X,Y,Z
n-m BY X,Y,Z
n- BY X,Y,Z
```

where n and m are positive integers and X,Y,Z are real numbers. One edge (n) or a group of edges (n-m) may be duplicated. n- represents all the edges from n through the last element in the EDGE array. The displacement vector is specified by X,Y,Z. If this vector is not specified, the user is prompted for it. Edges can be duplicated until <RETURN> is pressed or QUIT is entered.

8.2 DUPLICATE POLYGON

WHICH POLYGON(S)?

One polygon (n) or a group of polygons (n-m) may be duplicated. The specifications are entered in the same format as for DUPLICATE EDGE. The process of duplicating polygons can be terminated by hitting <RETURN> or entering QUIT.

Note: In DUPLICATE, the displacment vector can be entered using a type of shorthand. A zero may be represented by a blank or by no character at all.

e.g.	long form	short form
	10,8,0	10,8
	0,7,0	,7
	0,0,6	,,6
	1,0,3	1,,3

9. END

The graphics display screen is erased. Execution of BUILD is terminated.

10. ERASE

The graphics display screen is erased. Note: this command is ignored for the CALCOMP.

11. EXIT

The command source is changed to the console.

12. FIELD

The field of view is altered by moving the observer and changing the angle of view.

CURRENT DISTANCE TO ORGIN: x.xx
NEW DISTANCE TO ORIGIN?

A positive real number, X, is entered. The observer is positioned at the point (0,-X); i.e. X units in front of the viewing surface.

CURRENT ANGLE OF VIEW: x.xx
NEW ANGLE OF VIEW?

The angle is measured in degrees. A positive real number in entered.

13. GET

A structure file is opened. Elements are read into variables and arrays. This file becomes the new INFILE.

GET FILENAME

The specified file is opened and read.

GET

The INFILE is opened and read. If there is no INFILE, the user is prompted for a filename.

14. HELP

The BUILD commands and their functions are listed at the console. Only a portion of the list appears followed by

CONTINUE? Y/N

Either the letter Y (for yes) or the letter N (for no) is entered. The remaining commands are listed if Y is entered. Nothing else is listed if N is entered.

15. INPUT

The command source is changed to a specified file. Possible formats are

INPUT filename
or
INPUT

The user is prompted for a filename in the latter case.

16. LIST

The array elements and/or the transformation matrix are listed at the console.

16.1 LIST VERTEX

The world and window coordinates are listed. The user is prompted for the vertices to list:

WHICH VERTICES?

There are three formats for entering vertices:

n
n-m
n-

where n and m are positive integers. One vertex (n) or a group of vertices (n-m) may be listed. n- represents all the vertices from n through the last element in the VERTEX array. The user is prompted for more vertices with

WHICH VERTICES?

This process is repeated until the user hits <RETURN> or types QUIT.

16.2 LIST EDGE

Only the model's edges are listed.

WHICH ELEMENTS?

One edge (n) or a group of edges (n-m) may be listed. These specifications are entered in the same format as in LIST VERTEX. The user is prompted for edges until <RETURN> is pressed or QUIT is entered.

16.3 LIST POLYGON

The model's polygons are listed.

WHICH ELEMENTS?

One polygon (n) or a group of polygons (n-m) may be listed. These specifications are entered in the same format as in LIST VERTEX. The user is prompted for polygons until <RETURN> is pressed or QUIT is entered.

16.4 LIST MATRIX

The transformation matrix is listed.

17. MOVE

Elements are moved to new locations.

17.1 MOVE VERTEX

VERTEX n =

There are several possible formats for response:

```
n
n-m
n-
n TO X,Y,Z
n BY X,Y,Z
n- BY X,Y,Z
```

One vertex (n) or a group of vertices (n-m) may be moved. Note: n- represents all the vertices from n through the last element in the VERTEX array. n and m are entered as positive integers. Vertices may be moved by a distance relative to the origin; the displacement vector is given by (X,Y,Z). In addition, a single vertex can be moved to a specific point (X,Y,Z). In both cases, X,Y,Z are entered as real numbers. If the input consists of only the vertex number(s), the user will be prompted for a <u>relative</u> displacement vector. This process of moving vertices will continue until <RETURN> is pressed or QUIT is entered.

17.2 MOVE EDGE

MOVE EDGE(S)

The specifications for moving the edges are entered in one of several ways:

```
n
n-m
n-
n BY X,Y,Z
n-m BY X,Y,Z
n- BY X,Y,Z
```

where n and m are positive integers and X,Y,Z are real numbers. One edge (n) or a group of edges (n-m) may be moved. n- represents all the edges from n through the last element in the EDGE array. The displacement vector is specified by X,Y,Z. If this vector is not specified, the user is prompted for it. Edges can be moved until <RETURN> is pressed or QUIT is entered.

17.3 MOVE POLYGON

MOVE POLYGON(S)

One polygon (n) or a group of polygons (n-m) may be moved. The specifications are entered in the same format as for MOVE EDGE. The process of moving polygons can be terminated by hitting <RETURN> or entering QUIT.

Note: In MOVE, the displacement vector can be entered using a type of shorthand. A zero may be represented by a blank character or by no character at all.

e.g.	long form	short form
	10,8,0	10,8
	0,7,0	,7
	0,0,6	,,6
	1,0,3	1,,3

18. PRINT

The array elements and/or the transformation matrix are listed at the printer.

18.1 PRINT VERTEX

The world and window coordinates are printed. Then all the vertices are listed.

18.2 PRINT EDGE

All of the model's edges are listed.

18.3 PRINT POLYGON

All of the polygons are printed.

18.4 PRINT MATRIX

The transformation matrix is printed.

18.5 PRINT ALL

The world and window coordinates are printed. All of the vertices, edges, and polygons are listed. The transformation matrix is printed.

19. ROTATE

The model is rotated about a specified point. Rotations take place about X',Y',Z' axes. These axes passing through the given point, are parallel to the X,Y,Z axes, respectively, of the coordinate system.

ANGLES? (X,Y,Z)

Three real numbers, separated by commas, are entered. These numbers specify the angles (in degrees) of rotation about the X', Y', Z' axes, respectively. Note that the rotations will be made in the X', Y', Z' order.

RELATIVE ORIGIN? (X,Y,Z)

The point about which rotations occur is specified. The center of the model is chosen by hitting the <RETURN> key. A different point is chosen by entering the appropriate X,Y,Z coordinates. These coordinates are entered as three real numbers separated by commas.

Note: After both prompts, the response can be entered using a type of shorthand. A zero may be represented by a blank character or by no character at all.

e.g.	long form	short form
	10,8,0	10,8
	0,7,0	,7
	0,0,6	,,6
	1,0,3	1,,3

20. SAVE

Variables and array elements are written to a file. This structure file is closed and becomes the new OUTFILE.

SAVE filename

The current work file is saved in the specified file.

SAVE

The work file is saved in the OUTFILE. If there is no OUTFILE, the user is prompted for a filename.

21. SCALE

All the vertices are scaled about the center point of the structure.

SCALE FACTOR?

A positive real number is entered. If the number is greater than one, the model is magnified. The model is reduced if the number is less than one.

22. SELECT

The defaults for line width, color, and polygon filling are chosen.

LINE WIDTH (0-15)

An integer is entered. As the numbers increase, the lines become wider. At the start of program execution, the default value is 2.

COLOR (1-15)

An integer, corresponding to the desired color, is entered. Initially, the default value is 1.

FILL TYPE (0-4)

An integer, indicating a particular polygon filling, is entered. At the start, the default value is 0. (See FILTYP in Appendix A).

Note: If the default value for an attribute is not to be changed, the <RETURN> key is pressed after the prompt.

23. SET

At the beginning of program execution, the input device for data (vertices, edges, and polygons) defaults to the keyboard. SET is used to select a new default input device. The possible selections are

SET { KEYBOARD / JOYSTICK / TABLET } (default = KEYBOARD)

24. SQUEEZE

Vertices which were deleted with the DELETE command and vertices which are not used in any polygon or edge are removed from the VERTEX array. The remaining vertices are shifted up in the array in order to fill empty slots. The EDGE and POLY arrays are adjusted to account for the new vertex numbers.

25. STATUS

Some general information concerning the work file appears at the console. The filenames of the INFILE and OUTFILE are listed. The number of elements in the VERTEX, EDGE, and POLY arrays are indicated along with the size of each array.

26. TRANSLATE

The vertices of the model are translated.

TRANSLATE VERTICES BY (DX,DY,DZ)

Three real numbers, separated by commas, are entered. These numbers indicate the total displacement of the model in the X,Y,Z directions. The numbers can be entered using a type of shorthand. A zero may be represented by a blank character or by no character at all.

e.g.	long form	short form
	10,8,0	10,8
	0,7,0	,7
	0,0,6	,,6
	1,0,3	1,,3

27. WINDOW

The boundaries of the window are specified.

ENTER THE LOWER LEFT VERTEX:

If the input device is the joystick or the tablet, the picking implement is used to point to the lower left corner of the window and then button 1 is pressed. If the keyboard is the input device, then the user in prompted

X,Y,Z,IDV=

Three real numbers, followed by the number 1, are entered. These numbers are separated by commas or spaces. The real numbers specify the X,Y,Z, coordinates, respectively, of the lower left corner of the window. Note: The Z-coordinate is assigned the value zero, regardless of the value entered by the user.

ENTER THE UPPER RIGHT VERTEX:

This vertex is entered in the same manner as the lower left vertex.

28. WORLD

The boundaries of the world space are specified.

ENTER THE LOWER LEFT FRONT VERTEX:

If the input device is the joystick or the tablet, the picking implement is used to point to the lower left front corner of the box which will be the world space. Button 1 is then pressed. If the keyboard is the input device, then the user is prompted

X,Y,Z,IDV=

Three real numbers, followed by the number 1, are entered. These numbers are separated by commas or spaces. The real numbers specify the X,Y,Z coordinates, respectively, of the lower left front corner of the box.

ENTER THE UPPER RIGHT BACK VERTEX:

This vertex is entered in the same manner as the lower left front vertex.

APPENDIX E

The following is a description of each TITLES command. The prompts corresponding to each command, together with the formats of the user's responses, are indicated.

1. ADD

Character strings are added to a specified slide.

FRAME=

An integer is entered to specify a slide

COLOR=

An integer (1-15) is entered to specify a default color for the new character strings.

SET=

An integer (1-24) is entered to specify a default character set for the new text.

INPUT THE STRING

The character string is entered. The character sets and colors used within the string are modified with control sequences. In addition, control sequences are used to place subscripts and superscripts in the string and to change the justification of the text. See Appendix B for a list of these sequences.

TOGGLE BUTTON FOR LOWER LEFT

The cursor is moved to select the location of the lower left corner of the string. Once the cursor is set in place, the text is displayed.

DEL(1), MOV(2), CENTER(3), SCALE(4)=

An integer (1-4) is entered:

1 - The string is deleted.

2 - The text will be moved. TOGGLE BUTTON TO MOVE LL CORNER. The cursor is positioned at the desired location of the lower left corner of the text. Once the cursor is set in place, the text is moved.

3 - The string is centered on the line.

4 - The text is scaled by a SCALE FACTOR which is specified as a positive real number.

<RETURN> - The string is not modified.

Only one correction can be made at a time. The prompt for corrections will reappear until the string is deleted or the <RETURN> key is entered.

INPUT THE STRING

Another string may be added to the slide. When no other strings are to be added, the <RETURN> key is entered.

2. BULLET

A bullet is drawn on a specified slide.

FRAME=

An integer is entered to specify the slide on which the bullet will be drawn.

COLOR=

An integer (1-15) is entered to select the color of the bullet.

TOGGLE FOR CENTER

The cursor is positioned at the desired location of the bullet. Once the cursor is set, the bullet is displayed.

3. CIRCLE

A circle is drawn on a specified slide.

FRAME=

An integer is entered to specify the slide on which the circle will be drawn.

COLOR=

An integer (1-15) is entered to select the color of the bullet.

TOGGLE FOR CENTER

The cursor is positioned at the location of the circle's center.

TOGGLE FOR RADIUS

The cursor is positioned at the location of any point on the circle. Once the cursor is set, the circle is drawn.

4. COLORS

All of the available colors are displayed.

5. CORRECT

The text on a slide is corrected one string at a time.

FRAME=

An integer is entered to specifiy the slide which will be corrected. A line of text is printed at the console, followed by

DEL(1), MOV(2), CENTER(3), SCALE(4)=

An integer (1-4) is entered:

1 - The string is deleted.

2 - The text will be moved. TOGGLE BUTTON TO MOVE LL CORNER. The cursor is positioned at the desired location of the lower left corner of the text. Once the cursor is set in place, the text is moved.

3 - The string is centered on the line.

4 - The text is scaled by a SCALE FACTOR which is specified as a positive real number.

<RETURN> - The string is not modified

Only one correction can be made at a time. This prompt for corrections will reappear until the string is deleted or the <RETURN> key is entered.

The next line of text, along with the prompt for corrections, is displayed on the console. This process is repeated for each line of the text on the slide.

6. DELAY

The delay time for the hard copy device is specified. An integer is entered for this delay time.

7. END

The program is terminated.

8. HELP

The TITLES commands are listed.

9. NEW

A new slide is created. The frame number is incremented. New character strings are added to this slide as with the ADD command.

10. LINE

A line is drawn on a specified slide.

FRAME=

An integer is entered to specify the slide on which the line will be drawn.

COLOR=

An integer (1-15) is entered to select the color of the line.

TOGGLE FOR START OF LINE

The cursor is positioned at the location of one of the line's endpoints.

TOGGLE FOR END OF LINE

The cursor is positioned at the location of the other endpoint of the line. Once the cursor is set, the line is drawn.

11. PROCESS

Hard copies of specified slides are generated.

INPUT SLIDE # (0 TO PROCESS.) AND FRAME COUNT =

Two integers are entered to indicate the slide number and the number of hard copies, respectively.

INPUT SLIDE# (0 TO PROCESS,) AND FRAME COUNT=

Another slide may be specified for copying. When all the desired slides have been specified, a zero is entered. The hard copies are then processed and the program is terminated.

12. REREAD

A file is opened and data is read into variables.

RESTART FILE

The name of a data file is entered. A new file may be started by just hitting

<RETURN>.

n SLIDES
FUNCTION=

The user is informed of the number of slides in the the data file. A command is then specified by the user.

13. SAVE

The slides in the work file are written to a data file. If a data file has been opened, the slides are written to this file. Otherwise, the user is prompted for

FILE NAME.

The name of the new data file is entered.

14. SET

A character set is displayed.

SET NO.=

A integer (1-24) is entered to select a character set.

15. VIEW

A slide is displayed.

NUMBER=

An integer is entered. This number indicates the slide to be displayed. Once the image has been generated, the process is repeated with the NUMBER= prompt. When no other slides are to be displayed, a zero or the <RETURN> key is entered.

Appendix F

PROGRAM G2TEST . . . . page 1

SUBROUTINE ANGTST . . . . page 2

SUBROUTINE CIRC . . . . page 4

SUBROUTINE CUBE . . . . page 5

SUBROUTINE MM . . . . page 7

SUBROUTINE MM3 . . . . page 8

SUBROUTINE SETMAT . . . . page 9

SUBROUTINE TRANS . . . . page 10

MAIN PROGRAM . . . . page 11

SUBROUTINE GRAFIT1 . . . . page 12

SUBROUTINE GRAFIT2 . . . . page 13

SUBROUTINE RDCNL . . . . page 15

SUBROUTINE GRAFIT3 . . . . page 16

SUBROUTINE GRAFIT4 . . . . page 18

SUBROUTINE PLT . . . . page 19

SUBROUTINE PHASE . . . . page 20

SUBROUTINE RUMTST . . . . page 21

SUBROUTINE VENTV . . . . page 22

SUBROUTINE LAYER . . . . page 23

SUBROUTINE FIRE . . . . page 24

SUBROUTINE PLUME . . . . page 25

SUBROUTINE ROOM . . . . page 26

SUBROUTINE SADDLE . . . . page 27

SUBROUTINE SETCOL . . . . page 29

SUBROUTINE SYMTST . . . . page 30

SUBROUTINE VOLTST . . . . page 31

SUBROUTINE FILNGT . . . . page 33

```
      PROGRAM G2TEST
      INTEGER PB(6)
C
C     TEST EACH FUNCTION OF THE DEVICE INDEPENDENT PCKAGE
C
C
      WRITE(5,1) ' '
      WRITE(5,1) 'ANGLE  1'
      WRITE(5,1) 'CIRCLE  2'
      WRITE(5,1) 'FLYING CUBE  3'
      WRITE(5,1) 'GRAFIT(1)  4'
      WRITE(5,1) 'GRAFIT(2)  5'
      WRITE(5,1) 'GRAFIT(3)  6'
      WRITE(5,1) 'GRAFIT(4)  7'
      WRITE(5,1) 'PHASE SPACE PLOT (CONTOUR AND SURFACE PLOT)  8'
      WRITE(5,1) 'SINGLE ROOM FIRE  9'
      WRITE(5,1) 'SADDLE SURFACE 10'
      WRITE(5,1) 'HARDWARE/HERSHEY SYMBOLS 11'
      WRITE(5,1) '3D CONTOURING 12'
      WRITE(5,1) 'POLYGON FILLING 13'
      CALL SYSIO(PB,41,5,' TEST =',7,0)
      READ(5,3,END=4,ERR=4) I
C
C     GET THE DEVICE
C
      CALL SYSIO(PB,33,5,'DEVICE=',7,0)
      READ(5,2) ID
    2 FORMAT(I)
      CALL DEVICE(ID)
      GO TO (201,202,203,204,205,206,207,208,209,210,211,212,
     .                                213),I
      STOP 'NO SELECTION'
  201 CALL ANGTST
  202 CALL CIRC(ID)
  203 CALL CUBE
  204 CALL GRAFIT1(ID)
  205 CALL GRAFIT2(ID)
  206 CALL GRAFIT3(ID)
  207 CALL GRAFIT4(ID)
  208 CALL PHASE
  209 CALL RUMTST
  210 CALL SADDLE(ID)
  211 CALL SYMTST
  212 CALL VOLTST(ID)
  213 CALL FILNGT(ID)
    1 FORMAT(1X,A50)
    3 FORMAT(I)
    4 STOP
      END
C
```

```
      SUBROUTINE ANGTST
C
C     A SIMPLE TEST OF THE CHARACTER STUFF
C
      CHARACTER*1 TITLE(80)
      INTEGER PB(6)
      CHARACTER*80 STRING
C
C     FIRST A SIMPLE DEMONSTRATION
C
      CALL GRISET (0.,0.,100.,100.)
      CALL NEWFRM
      CALL WDDRAW (50.,90.,6.,0.,8.,0.,8.,'¶LL¶C02EFT¶.')
      CALL WDDRAW (50.,75.,6.,0.,8.,0.,8.,'¶M¶C01C¶C02ENTERED¶.')
      CALL WDDRAW (50.,60.,6.,0.,8.,0.,8.,'¶R¶C01R¶C02IGHT¶.')
      STRING='¶M¶C011¶C022¶C033¶C044¶C055¶C066¶C077¶C088¶C139¶C140¶.
     .'
      CALL WDDRAW (50.,45.,6.,0.,8.,0.,8.,STRING)
      STRING='¶M¶C01ALSO¶U¶C05SUPERSCRIPTS¶O ¶C01AND¶C05¶DSUBSCRIPTS
     .¶.'
      CALL WDDRAW (50.,30.,6.,0.,4.,0.,5.,STRING)
      STRING = '¶MABC¶U1.234¶.'
      CALL LABEL(STRING,60.,10.,90.,20.,.50)
C     STRING='¶M¶C01¶H04AND ¶C03YOU ¶C07¶H18CAN ¶H04CHANGE CHARACTER
C    . ¶C03¶H18SETS¶.'
C     CALL WDDRAW (50.,20.,6.,0.,4.,0.,5.,STRING)
      STRING = '¶C02¶H04S¶US¶US¶US¶.'
      CALL WDDRAW (10.,10.,6.,0.,4.,0.,5.,STRING)
      STRING = '¶C03¶OD¶DD¶DD¶DD¶.'
      CALL WDDRAW(20., 10., 6., 0., 4., 0., 5., STRING)
      STRING = '¶C08NOW TRY ONE YOURSELF!¶.'
      CALL WDDRAW(10.,1.,6.,0.,4.,0.,5.,STRING)
      CALL FRAME
      PAUSE
C
C     THE RESTART POSITION - SCART WITH THE ANGLE OF THE TEXT
C
    4 CALL SYSIO(PB,33,5,'ANGLE=',6,0)
      READ(5,1) ANGLE
    1 FORMAT(2F)
      IF(ANGLE.LT.0.0) GO TO 5
      THETA = ANGLE/57.295
      DO 7 I = 1, 80
    7 TITLE(I) = ' '
C
C     ERASE THE SCREEN AND PREPARE TO DRAW
C
      CALL NEWFRM
      CALL SYSIO(PB,33,5,'TEXT=',5,0)
      READ(5,3) TITLE
    3 FORMAT(80A1)
      DO 11 I = 1, 80
      II = 81 - I
      IF(TITLE(II).NE.' ') GO TO 12
   11 CONTINUE
```

```
      GO TO 4
12    TITLE(II+1) = '¶'
      TITLE(II+2) = '.'
C     CALL SYSIO(PB,33,5,'LINWID=',7,0)
C     READ(5,2) LW
      CALL DEFINE (0., 0., 13., 13.)
      CALL COLOR(4)
C     CALL LINWID(LW)
      CALL BOXPLT (0.5, 0.5, 12.5, 12.5)
      CALL COLOR(5)
      CALL WDCOUNT(TITLE,NT)
      IF(NT.EQ.0) GO TO 4
C
C   DRAW THE LABEL
C
      YB = 5.0
      XPOS = 5.0
      XRIGHT = XPOS + MIN(FLOAT(NT),11.)
      YTOP   = YB   + 1.
      CALL LABEL(TITLE, XPOS, YB, XRIGHT, YTOP, THETA)
      CALL FRAME
      GO TO 4
C
C   CLOSE THE PLOTTING DEVICE AND ERASE NON-LOCAL SCREENS
C
5     CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE CIRC(ID)
      CALL NEWFRM
      IF(ID.NE.1) THEN
        CALL DEFINE (0.,0.,10.,10.)
      ELSE
        CALL SCALNG (0.,0.,10.,10.,0.,0.,7.9,7.9,0.,0.,0.,0.)
      ENDIF
      CALL COLOR(2)
      CALL LINWID(4)
      CALL FILTYP(1)
      CALL CIRCLE(5.,5.,4.)
      CALL FRAME
      PAUSE
      CALL ERASE
      CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE CUBE
      COMMON QNORM(3,6),VERT(3,8),IEDGV1(12),IEDGV2(12),
     . ISURF(6),IPEDG(24),NVERT,NEDGS,NSURF,F,SCALE(4)
      INTEGER IV(2)/2,0/
C
      DIMENSION T(4,4),QM(4,4),QMT(4,4)
      DX = 4.
      DY = 4.
      XL = -1.5 - 0.1*4.
      XR = +3. + .15*4.
      XB = -1.5 - 0.1*4.
      XT = +3. + 0.15*4.
      CALL DEFINE (XL, XB, XR, XT)
      CALL DELAY(26)
      CALL NEWFRM
      CALL COLOR(5)
      CALL LINWID(3)
      NVERT = 8
      NEDGS = 12
      NSURF = 6
      F = 3.
      XINCR = .5 / 95.
      DO 100 I = 1, 4
      DO 100 J = 1, 4
      T(I,J) = 0.0
      IF(I.EQ.J) T(I,I) = 1.0
100   CONTINUE
      XC = 0.5
      YC = 0.5
      ZC = 0.5
      CALL SETMAT (QM,1,1.0,0,0,0)
      DO 1000 I = 1, 95
      CALL TRANS(T)
      XC1 = XC - XINCR
      YC1 = YC - XINCR
      ZC1 = ZC - XINCR
      QM(1,4) = -(QM(1,1)*XC + QM(1,2)*YC) + XC1
      QM(2,4) = -(QM(2,1)*XC + QM(2,2)*YC) + YC1
      QM(3,4) = -ZC + ZC1
      XC = XC1
      YC = YC1
      ZC = ZC1
      CALL MM (T,QM,T)
1000  CONTINUE
      CALL TRANS(T)
      CALL SETMAT(QM,2,1.0,0,0,0)
      DO 2000 I = 1, 192
      F = F + .15
2000  CALL TRANS(T)
      CALL SETMAT(QM,3,1.0,0,0,0)
      DO 3000 I = 1, 192
      F = F - .15
      CALL MM(T,QM,T)
3000  CALL TRANS(T)
      CALL SETMAT(QM,1,1.0,0,0,0)
```

```
      CALL SETMAT(QMT,2,1.0,0,0,0)
      CALL MM(QM,QMT,QM)
      DO 4000 I = 1, 96
      CALL MM(T,QM,T)
4000  CALL TRANS(T)
      CALL SETMAT(QMT,3,1.0,0,0,0)
      CALL MM (QM,QMT,QM)
      THETA = 3.14159265/180.
      ANGLE8 = 3.141592 + THETA
      DO 5000 I = 1, 360
      XC1 = SIN(ANGLE8)
      ZC1 = COS(ANGLE8)+ 1.0
      YC1 = FLOAT(I)/360.
      QM(1,4) = -(QM(1,1)*XC + QM(1,2)*YC + QM(1,3)*ZC) + XC1
      QM(2,4) = -(QM(2,1)*XC + QM(2,2)*YC + QM(2,3)*ZC) + YC1
      QM(3,4) = -(QM(3,1)*XC + QM(3,2)*YC + QM(3,3)*ZC) + ZC1
      XC = XC1
      YC = YC1
      ZC = ZC1
      ANGLE8 = ANGLE8 + THETA
      CALL MM (T,QM,T)
5000  CALL TRANS(T)
      CALL ENDFRM
      STOP
      END
```

```
          SUBROUTINE MM(A,B,C)
          DIMENSION A(4,4),B(4,4),C(4,4),D(4,4)
          DO 10  I = 1, 4
          DO 9 J = 1, 4
          D(I,J) = 0.0
          DO 8 K = 1, 4
 8        D(I,J) = D(I,J) + B(I,K) * C(K,J)
 9        CONTINUE
 10       CONTINUE
          DO 20 I = 1, 4
          DO 20 J = 1, 4
 20       A(I,J) = D(I,J)
          RETURN
          END
```

```
      SUBROUTINE MM3(A,B,C,K)
      DIMENSION B(4,4),A(3),C(3),D(3)
      DO 10 I = 1, 3
      IF(K.EQ.1) D(I) = B(I,4)
      IF(K.NE.1) D(I) = 0.0
      DO 9 J = 1, 3
9     D(I) = D(I) + B(I,J)*C(J)
10    CONTINUE
      DO 20 I = 1, 3
20    A(I) = D(I)
      RETURN
      END
```

```
      SUBROUTINE SETMAT(A,ITYPE,THETA,XT,YT,ZT)
      DIMENSION A(4,4)
      RAD = 3.141592 / 180.
      DO 10 I = 1, 4
      DO 10 J = 1, 4
 10   A(I,J) = 0.0
      A(4,4) = 1.0
      ANGLE = THETA * RAD
      CT = COS(ANGLE)
      ST = SIN(ANGLE)
      GO TO (100,200,300,400),ITYPE
 100  A(1,1) = CT
      A(2,1) = -ST
      A(1,2) = ST
      A(2,2) = CT
      A(3,3) = 1.0
      RETURN
 200  A(1,1) = 1.0
      A(2,2) = CT
      A(3,2) = ST
      A(2,3) = - ST
      A(3,3) = CT
      RETURN
 300  A(1,1) = CT
      A(3,1) = -ST
      A(1,3) = ST
      A(3,3) = CT
      A(2,2) = 1.0
      RETURN
 400  A(1,1) = 1.0
      A(2,2) = 1.0
      A(3,3) = 1.0
      A(1,4) = XT
      A(2,4) = YT
      A(3,4) = ZT
      RETURN
      END
```

```
      SUBROUTINE TRANS(R)
      COMMON QNORM(3,6),VERT(3,8),IEDGV1(12),IEDGV2(12),
     . ISURF(6),IPEDG(24),NVERT,NEDGS,NSURF,F,SCALE(4)
      DIMENSION R(4,4),RVERT(3,8),QN(4)
      DIMENSION IEDG(12),XP(12),YP(12)
      DO 10 I = 1, NEDGS
10    IEDG(I) = 0
      DO 20 I = 1, NVERT
      CALL MM3(RVERT(1,I),R,VERT(1,I),1)
20    CONTINUE
      IPTS = 1
      DO 100 I = 1, NSURF
      K = ISURF(I)
      IPTF = IPTS + K - 1
      CALL MM3(QN,R,QNORM(1,I),0)
      K1 = IPEDG(IPTS)
      K2 = IEDGV1(K1)
      XN2 = -RVERT(1,K2)
      YN2 = -RVERT(2,K2)
      ZN2 = -RVERT(3,K2) - F
      DOT = QN(1) * XN2 + QN(2) * YN2 + QN(3) * ZN2
      IF(DOT.LE.0.0) GO TO 100
      DO 30 J = IPTS, IPTF
      JJ = IPEDG(J)
30    IEDG(JJ) = IEDG(JJ) + 1
100   IPTS = IPTF + 1
      DO 200 I = 1, NVERT
      ZPF = RVERT(3,I) + F
      XP(I) = F * RVERT(1,I) / ZPF
      YP(I) = F * RVERT(2,I) / ZPF
200   CONTINUE
      CALL ERASE
      DO 300 I = 1, NEDGS
      IF(IEDG(I).EQ.0) GO TO 300
      I1 = IEDGV1(I)
      I2 = IEDGV2(I)
      CALL LINE (XP(I1), YP(I1), XP(I2), YP(I2))
300   CONTINUE
      CALL FRAME
C     CALL HDCOPY
      RETURN
      END
```

```
      BLOCKDATA BLK
      COMMON QNORM(3,6),VERT(3,8),IEDGV1(12),IEDGV2(12),
     . ISURF(6),IPEDG(24),NVERT,NEDGS,NSURF,F,SCALE(4)
      DATA QNORM/0,0,-1.,-1.,4*0.,2*1.,3*0.,-1.,0,0,1.,0/
      DATA VERT/4*0,1.,0,2*1.,0,1.,4*0,1.,0.,6*1.,0,1./
      DATA IEDGV1/1,2,3,4,5,6,7,8,1,2,3,4/
      DATA IEDGV2/2,3,4,1,6,7,8,5,5,6,7,8/
      DATA ISURF/6*4/
      DATA IPEDG/1,2,3,4,1,9,5,10,8,7,6,5,3,11,7,12,4,12,8,9,
     . 2,10,6,11/
      END
```

```
      SUBROUTINE GRAFIT1(ID)
C
      DIMENSION X(100), Y(100)
      CHARACTER*31 YAXIS/' IGNITION TIME (MINUTES)¶.'/
C
      CALL NEWFRM
C     CALL SETDEV(0,5)
      CALL COLOR(2)
      CALL GRISET (0., -50., 1023., 1023.)
      PIP = 3.1415
      PIM = - PIP
      CALL COLOR(3)
      CALL GRILAB(1,PIM,-2.,PIP,+2.)
      CALL GRAFIT (1,PIM, PIP, 200., 900., PIM, PIP, -2.0, +2.0,
     .  100., 800., -2.0, +2.0, 'X AXIS¶.', 3, YAXIS, 5)
      CALL COLOR(4)
      CALL GRILAB(2,PIM,-2.,PIP,+2.)
      CALL GRAFIT (2,PIM, PIP, 450., 650., PIM, PIP, -2.0, +2.0,
     .  200., 500., -2.0, +2.0, 'X AXIS¶.', 3, 'Y AXIS¶.', 5)
      XOFF = (PIP - PIM) / 100.
      DO 2 I = 1, 100
      XI = PIM + FLOAT(I-1) * XOFF
      YI = COS(XI)
      X(I) = XI
    2 Y(I) = YI
      CALL COLOR(1)
      CALL PLOTLN (1, X, Y, 100)
      CALL PLOTLN (2, X, Y, 100)
      CALL FRAME
      IF(ID.EQ.3) PAUSE
      CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE GRAFIT2(ID)
      DIMENSION X(100),Y(100)
C
C     READ IN SOME DATA (X VS Y)
C
      LIN = 3
      OPEN(UNIT=1,FILE='G1TEST2.DAT')
      CALL RDCNL(Y,I1,IY,I2)
      CALL RDCNL(X,I1,IX,I2)
      IZ = MIN0(IX, IY)
      I3 = IZ/3
    3 IREV = 0
      DO 4 I = 1, IZ-1
      II = I + 1
      IF(X(I).LE.X(II)) GO TO 4
      IREV = IREV + 1
      XP = X(I)
      YP = Y(I)
      X(I) = X(II)
      Y(I) = Y(II)
      X(II) = XP
      Y(II) = YP
    4 CONTINUE
      IF (IREV.GT.0) GO TO 3
C
C     SCALE - WE NOW KNOW HOW BIG TO MAKE THE AXES
C
      XMAX = 0.0
      XMIN = 1.E+10
      YMAX = 0.0
      YMIN = 1.E+10
      DO 1 I = 1, IZ
      XMAX = AMAX1(XMAX,X(I))
      XMIN = AMIN1(XMIN,X(I))
      YMAX = AMAX1(YMAX,Y(I))
    1 YMIN = AMIN1(YMIN,Y(I))
      YMINC = 1.E+10
      YMAXC = 0.0
      DO 5 I = 1, I3
      YMINC = AMIN1(YMINC,Y(I3-1+I))
    5 YMAXC = AMAX1(YMAXC,Y(I3-1+I))
C
C     NOW ROUND OFF THE AXES TO  APPROPRIATE WHOLE VALUES
C
          YMIN = 0.0
      DELTAX = (XMAX-XMIN)/9.
      DELTAY = (YMAX-YMIN)/9.
      XMAXA =        XMAX - DELTAX * 0.8
      XMINA =        XMIN + DELTAX * 1.5
      YMINA =        YMIN + DELTAY * 1.5
      YMAXA =        YMAX - DELTAY * .8
      XMINB = XMINA + DELTAX
      XMAXB = XMAXA - 2.0*DELTAX
      YMINB = YMINA + DELTAY * .76
      YMAXB = YMAXA - 2.7*DELTAY
```

```
      XMAX = IFIX(XMAX+DELTAX/2.)
      YMAX = IFIX((YMAX+DELTAY)/100.)*100.
      CALL NEWFRM
      CALL LINWID(LIN)
      CALL COLOR(1)
      CALL GRISET (XMIN, YMIN, XMAX, YMAX)
      CALL GRILAB(1,XMIN,YMIN,XMAX,YMAX)
      CALL GRAFIT(1,XMIN,XMAX,XMINA,XMAXA,XMIN,XMAX,YMIN,YMAX,
     . YMINA,YMAXA,YMIN,YMAX,' TIME(MINUTES)¶.',14,
     . 'TEMPERATURE(K)¶.',12)
      CALL GRILAB(2,X(I3),YMINC,X(2*I3),YMAXC)
      CALL GRAFIT(2,X(I3),X(2*I3),XMINB,XMAXB,X(I3),X(2*I3),
     . YMINC,YMAXC,YMINB,YMAXB,YMINC,YMAXC,
     . 'TIME(MINUTES)¶.', 5, 'FLUE TEMPERATURE(K)¶.', 5)
      CALL COLOR(2)
      CALL LINWID(4)
      CALL PLOTLN(1,X,Y,I3)
      CALL COLOR(3)
      CALL PLOTLN(1,X(I3-1),Y(I3-1),I3)
      CALL PLOTLN(2,X(I3),Y(I3),I3)
      CALL COLOR(2)
      CALL PLOTLN(1,X(2*(I3-1)),Y(2*(I3-1)),IZ-2*(I3-1))
      CALL FRAME
      IF(ID.EQ.3) PAUSE
      CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE RDCNL (REED,ICNL,MAXR,IEOF)
C
C     RRRRRRR   RRRRRRR    RRRRRR   RR    RR  RR
C     RRRRRRRR  RRRRRRRR  RRRRRRRR  RRR   RR  RR
C     RR    RR  RR    RR  RR    RR  RRRR  RR  RR
C     RR    RR  RR    RR  RR        RR RR RR  RR
C     RRRRRRR   RR    RR  RR    RR  RR  RRRR  RR
C     RR   RR   RRRRRRRR  RRRRRRRR  RR   RRR  RRRRRRRR
C     RR    RR  RRRRRRR    RRRRRR   RR    RR  RRRRRRRR
C
      DIMENSION REED(100)
      READ (1,10) MAXR,ICNL,IEND
      IF (IEND.EQ.999) THEN
        IEOF=1
      ELSE
        IEOF=0
        READ (1,20) (REED(IR),IR=1,MAXR)
      END IF
      RETURN
C
C
10    FORMAT (2I6,T78,I3)
20    FORMAT (7E11.5)
C
      END
```

```
      SUBROUTINE GRAFIT3(ID)
C
      DIMENSION X(100), Y(100)
      CHARACTER*31 YAXIS/'TIME SINCE IGNITION¶.MINUTES)¶.'/
      CHARACTER IC/'Z'/
C
      CALL SETDEV(0,5)
      XMIN = 0.
      XMAX = 1000.
      YMIN = 0.
      YMAX = 1000.
      XMAXG = 0.0
      YMAXG = 0.0
      DO 3 J = 1, 5
      CALL NEWFRM
      XMAXG = XMAXG + XMAX
      YMAXG = YMAXG + YMAX
      CALL GRISET(XMIN,YMIN,XMAXG,YMAXG)
      PIP = 3.1415
      PIM = - PIP
      CALL COLOR(3)
      DX = XMAX - XMIN
      DY = YMAX - YMIN
      X200 = XMIN + .2 * DX
      X900 = XMAX - .1 * DX
      Y100 = YMIN + .2 * DY
      Y800 = YMAX - .1 * DY
      CALL GRILAB(1,PIM,-2.,PIP,+2.)
      CALL GRAFIT (1,PIM, PIP, X200, X900, PIM, PIP, -2.0, +2.0,
     .   Y100, Y800, -2.0, +2.0, 'X AXIS¶.', 3, YAXIS, 5)
      CALL COLOR(4)
      X450 = XMIN + .45 * DX
      X650 = XMAX - .35 * DX
      Y200 = YMIN + .30 * DY
      Y500 = YMAX - .40 * DY
      CALL GRILAB(2,PIM,-2.,PIP,+2.)
      CALL GRAFIT (2,PIM, PIP, X450, X650, PIM, PIP, -2.0, +2.0,
     .   Y200, Y500, -2.0, +2.0, 'X AXIS¶.', 3, 'Y AXIS¶.', 5)
      XOFF = (PIP - PIM) / 10.
      DO 2 I = 1, 10
      XI = PIM + FLOAT(I-1) * XOFF
      YI = COS(XI)
      X(I) = XI
    2 Y(I) = YI
      CALL COLOR(1)
      CALL PLOTLN (1, X, Y, 10)
      CALL CHRSIZ(0., .05)
      CALL PLOTCH (1,X,Y,10,'*A')
      CALL PLOTLN (2, X, Y, 10)
      CALL PLOTCH (2,X,Y,10,IC)
      CALL FRAME
      IF(ID.EQ.3) PAUSE
    3 CONTINUE
      CALL ENDFRM
      STOP
```

```
      END
```

```
      SUBROUTINE GRAFIT4(ID)
      DIMENSION Q(550),VS(550),X(500),Y(500),TM(500),SQT(500)
      CHARACTER*16 FILE
      CHARACTER*40 TITLE/'GRAFIT(4)'/
C
      FILE = 'G1TEST4.DAT'
      TSTAR = 10.
      B = .321
      OPEN (UNIT=8,FILE=FILE)
      NP=0
      N1=1
    6 READ(8,3,END=5) N
    3 FORMAT(I2)
      READ(8,4) (VS(J),Q(J),TM(J),J=1,N)
      DO 10 I=1,N
      FT=B*SQRT(TM(I))
       IF(TM(I).GT.TSTAR) FM=1.
   10  SQT(I)=Q(I)*FT
    4 FORMAT(50X,3F10.2)
      N2=NP+(N-2)
      J=1
      DO 8, I=N1,N2
      Y(I)=VS(J)
      X(I)=SQT(J)
    8 J=J+1
      NP=N2
      N1=NP+1
      GO TO 6
    5 CONTINUE
      TYPE *, TITLE
      NP=N2
      CALL PLT(NP,X,Y)
      IF(ID.EQ.1) GO TO 50
      PAUSE
   50 CONTINUE
      CALL ENDFRM
      STOP
      END
C
```

```
      SUBROUTINE PLT(N,X,Y)
      DIMENSION X(550),Y(550)
      CHARACTER*40 TITL
      CHARACTER*25 YLABL
      CHARACTER*25 XLABL
C
      TITL = '  Rigid Foam¶.'
      XLABL='q . F(t) (W/cm¶U2¶0)¶.'
      YLABL='1/  V    (s/mm)¶.'
      ymax = 2.0
      IYTIC = 10
      CALL NEWFRM
      CALL COLOR(1)
      CALL GRISET(0.,0.,1100.,1100.)
      CALL COLOR(8)
      CALL GRAFIT(1,0.,6.,200.,900.,0.,6.,0.,YMAX,200.,900.,0.,YMAX,
     &XLABL,6,YLABL,IYTIC)
      CALL COLOR(1)
      CALL PLOTCH(1,X,Y,N,'o')
      CALL COLOR(5)
      CALL LABEL(TITL,250.,950.,850.,1000.,0.0)
      CALL FRAME
      RETURN
      END
```

```
      SUBROUTINE PHASE
      DIMENSION Z(30,30), X(8),Y(8), FL(8), TEST(30,30)
      DATA A/1./, B/-4./, C/5./, NINT/8/
      DATA X/100., 800., 250., 950., 100., 800., 250., 950./
      DATA Y/100., 100., 300., 300., 800., 800., 950., 950./
C
      DO 1 I = 1,30
      DO 1 J = 1, 30
1     Z(I,J) = A*COS(-.4+FLOAT(I)/5.)*
     . (1.+B*EXP(-C*(FLOAT(J-10)/7.)**2))+5
      ZMAX = -1.E+9
      ZMIN = +1.E+9
      DO 4 I = 1, 30
      DO 4 J = 1, 30
      ZMAX = MAX(ZMAX, Z(I,J))
4     ZMIN = MIN(ZMIN, Z(I,J))
      CALL LINWID(1)
      CALL NEWFRM
      CALL DEFINE(0.,-100.,1000.,1000.)
      CALL SRFSET(X,Y,0.,10.,30,30)
      CALL COLOR(2)
      CALL SURFAC(Z,30,30,1)
      CALL COLOR(3)
      CALL SURFAC(Z,30,30,-1)
C
C     SET UP A REASONABLE CONTOURING INTERVAL
C
      CALL CNTSET(X(1),Y(1),X(2),Y(2),X(4),Y(4),X(3),Y(3))
      DELTAZ = (ZMAX - ZMIN) * 0.9 / FLOAT (NINT)
      FL(1) = ZMIN + 0.5 * DELTAZ
      DO 2 I = 2, 8
2     FL(I) = FL(I-1) + DELTAZ
      DO 3 I = 1, 8
      CALL COLOR(I)
3     CALL CONTUR (Z, TEST, 30, 30, FL(I))
      CALL FRAME
      PAUSE
      CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE RUMTST
      COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ
      REAL*4 LSIZE, DELTA
      CALL NEWFRM
      CALL FILTYP(2)
      CALL ROOM (0.0, 5.0, 0.0, 2.5, .05, 1.0, 0.7)
      CALL VENTV (0.1, 0.5, 0.0, -1.)
      A = -1.5 / 20.
      B = 2.2 - A
      I = 1
      FSIZE = FLOAT(I)*A + B
      LSIZE = FSIZE + 0.2
      PSIZE = FSIZE
      CALL FIRE (4.0, 0.5, 0.2, FSIZE)
      CALL PLUME (4.0, 0.5, 0.5, PSIZE, 0.2)
      CALL LAYER (0.05, 2.45, 4.95, 2.48)
      CALL FRAME
      CALL IOWAIT(1000)
      DELTA = -A
      DO 101 I = 1, 21
      CALL FIRE (4.0, 0.5, 0.2, FSIZE)
      CALL PLUME (4.0, 0.5, 0.5, PSIZE, 0.2)
      FSIZE = FLOAT(I)*A + B
      LSIZE = FSIZE + 0.2
      PSIZE = FSIZE
      CALL FIRE (4.0, 0.5, 0.2, FSIZE-2*DELTA)
      CALL PLUME (4.0, 0.5, 0.5, PSIZE-DELTA, 0.2)
      CALL LAYER(0.05, LSIZE-DELTA, 4.95, 2.48)
      CALL FRAME
      CALL IOWAIT(1000)
101   CONTINUE
      PAUSE
      CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE VENTV (BOTM, TOP, SIDE, THICK)
      COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ
      REAL BOTM, TOP, SIDE
C
C     DRAW VERTICAL LINES IN BLACK AND HORIZONTAL LINES IN WHITE
C
      Y1 = BOTM
      Y2 = TOP
      DO 2 I = 1, 2
      X1 = SIDE + (I-1) * WALLSZ * THICK
      X2 = X1
    2 CALL LINE (X1, Y1, X2, Y2)
      X1 = SIDE
      X2 = WALLSZ * THICK
      DO 1 I = 1, 2
      Y1 = BOTM + (I-1) * (TOP-BOTM)
      Y2 = Y1
    1 CALL LINE (X1, Y1, X2, Y2)
      RETURN
      END
```

```
      SUBROUTINE LAYER (LEFT, BOTTOM, RIGHT, TOP)
C
C     DRAW SQUIGGLE FROM LEFT TO RIGHT TO REPRESENT
C     THE LAYER (HOT) ABOVE THE PLUME
C
      INTEGER DIVCNT
      REAL LEFT, RIGHT, BOTTOM, TOP, X(4), Y(4)
C
C     NOW DRAW THE SQUIGGLE
C
      CALL COLOR(2)
      X(1) = LEFT
      Y(1) = BOTTOM
      X(2) = LEFT
      Y(2) = TOP
      X(3) = RIGHT
      Y(3) = TOP
      X(4) = RIGHT
      Y(4) = BOTTOM
      CALL PLYGON(X, Y, 4)
      RETURN
      END
```

```
      SUBROUTINE FIRE (CENTER, WIDTH, HEIGHT, FSIZE)
      COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ
C
C PLOT THE FIRE SOURCE
C
          CALL COLOR(3)
      CALL VBXPLT (CENTER-.5*WIDTH, 0.0,
     .              CENTER+.5*WIDTH,HEIGHT)
C
C START AT HEIGHT [HP=(1+EPS)*HEIGHT] AND TRACE THE PARABOLA
C
      CALL COLOR(8)
      HP = 1.2 * HEIGHT
      W5 = WIDTH * .5
      SLOPE = FSIZE / W5**2
      W20 = WIDTH / 20.
      YB = HP
      XB = CENTER - W5
      DO 1 I = 1, 21
      XA = XB
      YA = YB
      XB = CENTER - W5 + W20* FLOAT(I-1)
      YB = -(XB-CENTER)**2*SLOPE + HP + FSIZE
    1 CALL LINE (XA, YA, XB, YB)
      RETURN
      END
```

```
      SUBROUTINE PLUME (CENTER, WIDTH, PBOTM, PTOP, FSOURC)
C
C     TO PUT THE PLUME IN, NORMALLY ABOVE THE FIRE
C     THIS ROUTINE ASSUMES A POINT SOURCE PLUME AT A
C     VIRTUAL DISTANCE BELOW THE FIRE SUCH THAT THE
C     PLUME SUBTENDS AN ANGLE OF 11 DEGREES AT THE FIRE
C     SOURCE.
C
      REAL CENTER, WIDTH, PBOTM, PTOP, FSOURC
      COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ
C
C     PLOT THE PLUME
C
      RADIUS = 0.5 * WIDTH
      THETA = 11. / 57.1
C     VIRTUAL = RADIUS / ATAN(THETA)
      RTAN = ATAN (THETA)
C
C     DRAW TWO LINES FROM THE VIRTUAL POINT WITH AN ANGLE
C     OF THETA AND VISIBLE SEGMENTS
C     FROM PBOTM TO PTOP
C
      CALL COLOR(7)
      Y1 = PBOTM
      Y2 = PTOP
      DO 1 I = 1, 2
      PM = (-1.0)**I
      X1 = CENTER + PM*(RADIUS+(PBOTM-FSOURC)*RTAN)
      X2 = CENTER + PM*(RADIUS+(PTOP -FSOURC)*RTAN)
    1 CALL LINE (X1, Y1, X2, Y2)
      RETURN
      END
```

```
      SUBROUTINE ROOM (L, R, B, T, DW, FW, FH)
      REAL L, R, T, B, F, SCALE(4)
      COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE,WALLSZ,CEILSZ
C
C - L = LEFT SIDE OF ROOM
C - R = RIGHT SIDE OF ROOM
C - T = TOP OF THE ROOM
C - B = BOTTOM OF THE ROOM
C - DW = WIDTH OF WALL
C - F IS THE FRACTION OF THE SCREEN TO BE USED
C
      XFW = AMIN1 (1., FW)
      XFH = AMIN1 (1., FH)
      DWMIN = AMIN1 (R-L, T-B)
      WALLSZ = DWMIN * DW
      CEILSZ = 0.5 * XFW/XFH * WALLSZ
      DY = T - B
      DX = R - L
      XL = L - DX/3.5
      XR = R + DX/3.5
      XB = B - DY/3.5
      XT = T + DY/3.5
      CALL DEFINE (XL, XB, XR, XT)
      CALL COLOR(1)
      DO 1 IW = 1, 2
      WALL = WALLSZ * (IW-1)
      CEIL = CEILSZ * (IW-1)
      XL = L - WALL
      XR = R + WALL
      XB = B
      XT = T + CEIL
    1 CALL BOXPLT (XL, XB, XR, XT)
      RETURN
      END
```

```
      SUBROUTINE SADDLE(ID)
C
C     PROGRAM TO TEST THE SURFACE PLOTTING AND CONTOUR ROUTINES.
C
      PARAMETER (NX=25,NY=13,NP=4)
      REAL      Z(NX,NY), XP(8,9),YP(8,9), FL(16), TEST(NX,NY)
      INTEGER   INDX(NP)
      DATA      ZMIN, ZMAX /-0.75, 200./, MX, MY /1, 1/
      DATA      INDX/1,3,7,9/
      DATA      XP/
     1          100., 800., 250., 900., 100., 800., 250., 900.,
     2          100., 900., 200., 800., 100., 900., 200., 800.,
     3          200., 900., 100., 750., 200., 900., 100., 750.,
     4          100., 800., 250., 900., 100., 800., 250., 900.,
     5          100., 900., 200., 800., 100., 900., 200., 800.,
     6          200., 900., 100., 750., 200., 900., 100., 750.,
     7          100., 800., 250., 900., 100., 800., 250., 900.,
     8          100., 900., 200., 800., 100., 900., 200., 800.,
     9          200., 900., 100., 750., 200., 900., 100., 750./
      DATA      YP/
     1          100., 100., 250., 250., 800., 800., 900., 900.,
     2          100., 100., 250., 250., 800., 800., 900., 900.,
     3          100., 100., 250., 250., 800., 800., 900., 900.,
     4          100., 100., 200., 200., 900., 900., 800., 800.,
     5          100., 100., 200., 200., 900., 900., 800., 800.,
     6          100., 100., 200., 200., 900., 900., 800., 800.,
     7          200., 200., 100., 100., 900., 900., 750., 750.,
     8          200., 200., 100., 100., 900., 900., 750., 750.,
     9          200., 200., 100., 100., 900., 900., 750., 750./
C
C     INITIALIZE THE DATA
C
      DO 20 J = 1, NY
      DO 20 I = 1, NX
      Z(I,J) = ABS(FLOAT(J)-7.)**2 + (144.-ABS(FLOAT(I)-13.)**2)
   20 CONTINUE
      DO 2 I = 1, 8
      FL(I) = -1.0 + FLOAT(I)*10.
    2 FL(I+8) = 0.2 + FLOAT(I)*10.
C
C     INITIALIZE THE GRAPHICS PACKAGE
C
      CALL DEFINE (0., 0., 1023., 1023.)
      CALL NEWFRM
C
C     LOOP OVER THE VARIOUS PERSPECTIVE PLOTS.
C
      IPP = 3
      DO 100 IPP = 1, NP
      IP = INDX(IPP)
C
C     PLOT THE UPPER PART OF THE SURFACE WITH FRONT SKIRT.
C
      CALL SRFSET (XP(1,IP), YP(1,IP), ZMIN, ZMAX, NX, NY)
      CALL ERASE
```

```
             CALL SETLUT
             CALL COLOR(5)
             CALL SURFAC (Z, NX, NY, +1)
             CALL COLOR(4)
             CALL SFRAME (2)
             CALL FRAME
             CALL IOWAIT(2000)
C
C      PLOT THE LOWER PART OF THE SURFACE.
C
             CALL SRFSET (XP(1,IP), YP(1,IP), ZMIN, ZMAX, NX, NY)
             CALL ERASE
             CALL COLOR(7)
             CALL SURFAC (Z, NX, NY, -1)
             CALL COLOR(4)
             CALL SFRAME (2)
             CALL FRAME
             CALL IOWAIT(2000)
C
C      PLOT BOTH PARTS OF THE SURFACE WITH SIDE SKIRTS.
C
             CALL SRFSET (XP(1,IP), YP(1,IP), ZMIN, ZMAX, NX, NY)
             CALL ERASE
             CALL COLOR(5)
             CALL SURFAC (Z, NX, NY, +1)
             CALL COLOR(7)
             CALL SURFAC (Z, NX, NY, -1)
             CALL SFRAME (2)
             CALL FRAME
             IF(ID.EQ.3) THEN
               PAUSE
             ELSE
               CALL IOWAIT(2000)
             ENDIF
    100      CONTINUE
             CALL ENDFRM
C
             STOP
        END
```

```
      SUBROUTINE SETCOL(IC)
      DIMENSION LUT(16,3)
C
      DO 1 I = 1, 3
      DO 1 J = 1, 16
1     LUT(J,I) = 0
      DO 2 I = 1, 8
      LUT(I+1,2) = I*2 - 1
2     LUT(I+1,3) = I*2 - 1
      DO 3 I = 1, 7
      LUT(I+9,1) = I*2 + 1
3     LUT(I+9,2) = I*2 + 1
      CALL DSLWT (16, 48, LUT)
      RETURN
      END
```

```
      SUBROUTINE SYMTST
C
C     A SIMPLE TEST OF THE CHARACTER STUFF
C
      CHARACTER*28 TITLE1/'A SET OF HARDWARE CHARACTERS'/
      CHARACTER*19 TITLE2/'¶1 SET FROM SET 1¶.'/
      CHARACTER*32 TITLE3/'¶:A SET ([§/*†]) FROM SET #10¶.'/
      ISC=0
    3 XSZ = 11. * 2.**ISC - 11.
      CALL NEWFRM
      CALL DEFINE(-XSZ, -XSZ, 11.+XSZ, 11.+XSZ)
      CALL COLOR(4)
      CALL BOXPLT (0.5, 0.5, 8.5, 4.5)
      CALL BOXPLT (0.5, 5.5, 8.5, 9.5)
      CALL BOXPLT (9.0, 0.5, 10.7, 9.5)
      DX = 5. / 28.
      XL = 4.5 - 14.*DX
      CALL COLOR(1)
      CALL CHRSIZ (DX/(11.+2.*XSZ),0.)
      DO 1 I = 1, 28
    1 CALL SYMBOL (XL+DX*FLOAT(I), 7.5, TITLE1(I:I))
      CALL COLOR(3)
      CALL LABEL(TITLE2, 2.0, 2.0, 8.0, 2.5, 0.0)
      CALL COLOR(2)
      CALL LABEL(TITLE3, 10.3, 1., 17.3, 2., 3.14/2.)
      CALL FRAME
      CALL IOWAIT(2000)
      ISC = ISC + 1
      IF(ID.NE.3) GO TO 4
      IF(ISC.GT.7) GO TO 4
      GO TO 3
    4 CALL ENDFRM
      STOP
      END
```

```
      SUBROUTINE VOLTST(ID)
C
C     VOLUME TEST PROGRAM WITH REDUCED RESOLUTION HIDDEN LINES.
C
C     THIS PROGRAM DEMONSTRATES THE USE OF THE "VOLUME" PLOTTER.
C     IT USES A SIMPLE SOLID ( TWO OVERLAPPING SPHERES)
C     WITH A SOLID PILLAR PLACED IN FRONT TO OBSCURE PART OF THE
C     SPHERES AND DEMOSTRATE THE HIDDEN LINE FEATURE.  THIS IS DONE
C     TOGETHER WITH THE NECESSARY INTERACTION WITH THE GRAPHICS
C     PACKAGE.  THIS SHOULD BE USED AS A MODEL FOR DESIGNING PROGRAMS UN
C     THE USER IS FAMILIAR WITH VOLUME.
C
          PARAMETER (NT=100, NT2=NT*NT)
          REAL T(NT2), CLEVE(2)
          REAL     XP(8,3), YP(8,3)
          REAL     F(20, 20, 20)
          INTEGER  MODE(2)
          DATA     NX,NY,NZ /20, 20, 20/
          DATA     MODE /-1, 2/, CLEVE /1.5, -1.0/
          DATA     NP, ITAPE /3, 105/
          DATA     XP/
     1             100., 800., 250., 900., 100., 800., 250., 900.,
     2             100., 900., 200., 800., 100., 900., 200., 800.,
     3             200., 900., 100., 750., 200., 900., 100., 750./
          DATA     YP/
     1             100., 100., 250., 250., 800., 800., 900., 900.,
     2             100., 100., 250., 250., 800., 800., 900., 900.,
     3             100., 100., 250., 250., 800., 800., 900., 900./
C
C     INITIALIZE THE PACKAGE
C
          CALL DEFINE(0., 0., 1024., 1024.)
C
C     LOOP OVER FOUR SIZES OF THE SPHERES
C
          DO 100 IPIC = 1,901,300
          NP = NP
          IP = 1
          FAC = 1.0 + 0.01*FLOAT(IPIC)
C
C     SET UP THE VALUE OF THE FUNCTION.
C     IT IS AN NRL PEANUT ( 2 OVERLAPPING SPHERES)
C     (X-13)**2 + (Y-10.5)**2 + (Z-10.5)**2
C     (X-8)**2 + (Y-10.5)**2 + (Z-10.5)**2
C
          DO 1 I = 1, NX
          DO 1 J = 1, NY
          DO 1 K = 1, NZ
          R1SQ = (I-8.0)**2 + (J-10.5)**2 + (K-10.5)**2
          R2SQ = (I-13.)**2 + (J-10.5)**2 + (K-10.5)**2
          F(I,J,K) = FAC/R1SQ + 2.0*FAC/R2SQ
    1     CONTINUE
C
C     CONSTRUCT THE PILLAR
C
```

```
          DO 2 I = 1, NX
          DO 2 J = 1, 5
          DO 2 K = 1, NZ
          RSQ = (I-10.5)**2 + (J-3.5)**2
          F(I,J,K) = -4.0/RSQ
    2     CONTINUE
C
C     INITIALIZE THE PLOTTING PACKAGE
C
C     IN THIS CASE, "VOLSET" IS CALLED FOR EACH CYCLE SO THAT THE
C     VARIOUS CONTOUR LEVELS WILL BE PLOTTED.  THE PILLAR IS PLOTTED
C     ONLY ONCE(THIRD CALL) BUT STUFF BEHIND IT IS ALWAYS HIDDEN
C     (NCL=2, MODE=2, CLEVE=-1.).
C
          CALL NEWFRM
C
C     DRAW THE FIGURE
C
          CALL COLOR(1)
          CLEVE(1) = 1.5
          CALL VOLSET (XP(1,IP), YP(1,IP), T, NT)
          CALL VOLUME(MODE, F, NX,NY,NZ, CLEVE, 2, T, NT)
C
          CALL COLOR(2)
          CLEVE(1) = 0.5
          CALL VOLSET (XP(1,IP), YP(1,IP), T, NT)
          CALL VOLUME(MODE, F, NX,NY,NZ, CLEVE, 2, T, NT)
C
          CALL COLOR(3)
          CALL VOLSET (XP(1,IP), YP(1,IP), T, NT)
          CALL VOLUME (+1, F, NX,NY,NZ, -1.0, 1, T, NT)
C
          CALL COLOR(4)
          CALL VOLUME (-2, F, NX,NY,NZ, 0.5, 1, T, NT)
C         CALL VOLFRM (2)
          CALL FRAME
          CALL IOWAIT(2000)
C
  100     CONTINUE
C
C     CLOSE-OUT GRAFIT.
C
          IF(ID.EQ.3) PAUSE
          CALL ENDFRM
C
          STOP
      END
```

```
      SUBROUTINE FILNGT
C
      CALL NEWFRM
      CALL DEFINE(-11.,-11.,11.,11.)
      DO 2 I = 1, 9
      CALL FILTYP(I-1)
      X = -11. + FLOAT(I)*2.1
      Y = -11. + FLOAT(I)*2.1
      CALL CIRCLE(X, Y, 2.0)
2     CONTINUE
      CALL FRAME
      PAUSE
      CALL ENDFRM
      STOP
      END
```

SUBROUTINE ALABEL . . . . page 1

FUNCTION AND . . . . page 2

SUBROUTINE BOXPLT . . . . page 3

ENTRY VBXPLT . . . . page 3

SUBROUTINE CHPLOT . . . . page 4

SUBROUTINE CIRCLE . . . . page 5

SUBROUTINE CONTUR . . . . page 6

SUBROUTINE DEVICE . . . . page 10

ENTRY SETDEV . . . . page 12
ENTRY NEWFRM . . . . page 12
ENTRY FRAME . . . . page 12
ENTRY ERASE . . . . page 12
ENTRY HDCOPY . . . . page 12
ENTRY COLOR . . . . page 12
ENTRY FILTYP . . . . page 12
ENTRY CHRSIZ . . . . page 13
ENTRY LINWID . . . . page 13
ENTRY SCALNG . . . . page 13
ENTRY DEFINE . . . . page 13
ENTRY ENDFRM . . . . page 14
ENTRY DELAY . . . . page 20
ENTRY IOWAIT . . . . page 20
ENTRY LEXDID . . . . page 20
ENTRY TABLET . . . . page 20

SUBROUTINE DEVINP . . . . page 22

SUBROUTINE ENUMBR . . . . page 24

SUBROUTINE FNUMBR . . . . page 25

SUBROUTINE GRAFIT . . . . page 28

ENTRY PLOTCH . . . . page 31
ENTRY PLOTLN . . . . page 31
ENTRY PLOTBAR . . . . page 32
ENTRY GRISET . . . . page 32
ENTRY GRILAB . . . . page 32

SUBROUTINE HHDRAW . . . . page 34

SUBROUTINE HSETS . . . . page 36

INTEGER FUNCTION IDCHAR . . . . page 38

ENTRY IREVCH . . . . page 38

SUBROUTINE LABEL . . . . page 39

SUBROUTINE LINE . . . . page 40

SUBROUTINE LINES . . . . page 41

SUBROUTINE LNPLOT . . . . page 42

INTEGER FUNCTION LSHF . . . . page 43

SUBROUTINE MAPIN . . . . page 44

SUBROUTINE MAPOUT . . . . page 45

SUBROUTINE PLYGON . . . . page 46

SUBROUTINE PLYPL1 . . . . page 47

SUBROUTINE PLYPL2 . . . . page 49

SUBROUTINE PLYPL3 . . . . page 50

SUBROUTINE PLYPL4 . . . . page 52

SUBROUTINE PLYPL5 . . . . page 54

SUBROUTINE PLYPLT . . . . page 56

ENTRY PLYGNS . . . . page 59

SUBROUTINE PROPOL . . . . page 61

ENTRY CNTSET . . . . page 63
ENTRY CNTFRM . . . . page 63

SUBROUTINE PUTCH . . . . page 64

SUBROUTINE PUTLNV . . . . page 66

SUBROUTINE SETLUT . . . . page 68

SUBROUTINE SRFSET . . . . page 69

ENTRY SFRAME . . . . page 71
ENTRY SSKIRT . . . . page 71
ENTRY SURFAC . . . . page 73

SUBROUTINE SURF4 . . . . page 77

ENTRY SURF5 . . . . page 77

SUBROUTINE SYMBOL . . . . page 78

SUBROUTINE VIEWTR . . . . page 79

SUBROUTINE VOLFRM . . . . page 83

SUBROUTINE VOLUM0 . . . . page 84

ENTRY VOLUM6 . . . . page 84

SUBROUTINE VOLUM7 . . . . page 85

SUBROUTINE VOLUM8 . . . . page 87

SUBROUTINE VOLUME . . . . page 89

ENTRY VOLSET . . . . page 92

SUBROUTINE WDCOUNT . . . . page 94

SUBROUTINE WDDRAW . . . . page 96

ENTRY CHRSET . . . . page 99

```
      SUBROUTINE ALABEL (CHARS, XL, XR, ANGLE)
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/XNGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      CHARACTER*1 CHARS(*)
      CALL WDCOUNT(CHARS, NC)
      IF(NC.LE.0) THEN
        IF(LUDIAG.GT.0) WRITE(LUDIAG,3) (CHARS(I),I=1,132)
3       FORMAT(' NO ESCAPE SEQUENCE IN ALABEL -',/,1X,132A1)
        RETURN
      ENDIF
      DX = (XR - XL) / MAX(1.,FLOAT(NC))
      DY = 4./3. * DX * XYCOOD(1)/XYCOOD(3)
      XNGLE = ANGLE
      CALL WDDRAW (XL, YB, DX, 0.0, DX, 0.0, DY, CHARS)
      RETURN
      END
```

```
      SUBROUTINE BOXPLT (X1, Y1, X2, Y2)
C
C***********************************************************************
C     BOXPLT
C***********************************************************************
C
      REAL XA(4),XB(4),YA(4),YB(4)
C
      ENTRY VBXPLT (X1, Y1, X2, Y2)
      XA(1)=X1
      YA(1)=Y1
      XB(1)=X2
      YB(1)=Y1
      XA(2)=X2
      YA(2)=Y1
      XB(2)=X2
      YB(2)=Y2
      XA(3)=X2
      YA(3)=Y2
      XB(3)=X1
      YB(3)=Y2
      XA(4)=X1
      YA(4)=Y2
      XB(4)=X1
      YB(4)=Y1
      CALL LINES(XA,YA,XB,YB,4)
      RETURN
      END
```

```
      SUBROUTINE CHPLOT (XARRAY, YARRAY, CHARAC, I1, I2, I3)
C
C      XARRAY REAL    - AN ARRAY, DIMENSIONED N, OF X COORDINATES OF
C                       THE DATA (MATH SPACE)
C      YARRAY REAL    - AN ARRAY, DIMENSIONED N, OF Y COORDINATES OF
C                       THE DATA (MATH SPACE)
C      CHARAC LITERAL - CHARACTER TO BE PLOTTED
C      I1     INTEGER - INDEX OF FIRST POINT TO BE PLOTTED
C      I2     INTEGER - INCREMENT AT WHICH POINTS ARE TO BE SELECTED
C                       TO PLOT
C      I3     INTEGER - INDEX OF LAST POINT IN DATA ARRAYS
C***********************************************************************
C     CHPLOT
C***********************************************************************
C
         COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
         COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
         INTEGER I1, I2, I3, CHCODE
         REAL XARRAY(4), YARRAY(4)
         CHARACTER*1 CHARAC
         INTEGER IX,IY
         NOPTS = (I3-I1+I2) / I2
         I1M = I1 - I2
         DO 1 I=1,NOPTS
         II=I1M+I2*I
    1    CALL SYMBOL (XARRAY(II), YARRAY(II), CHARAC)
         RETURN
      END
```

```
      SUBROUTINE CIRCLE(X, Y, R)
C
C     DRAW A CIRLE OF WIDTH LINWDM AND FILL WITH LFLMAT.
C
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFMOD/LFLMAT(5),LINWDM(5),LCLMAT(5)
      REAL XO, YO, XN, YN, DX, DY, THETA, ARC
C
      IF(IDEVIC.EQ.0) RETURN
      GO TO (1,1,2,1,1), IDEVIC
    1 ARC = 3.1415/50.
      THETA = 0.0
      XO = X + R
      YO = Y
      DO 10 I = 1, 101
      THETA = THETA + ARC
      XN =   R * COS(THETA) + X
      YN =   R * SIN(THETA) + Y
      CALL LINE(XO, YO, XN, YN)
      XO = XN
      YO = YN
   10 CONTINUE
      RETURN
    2 XL = MAX(X * XYCOOD(1) + XYCOOD(2),0.)
      YL = MAX(Y * XYCOOD(3) + XYCOOD(4),0.)
      RL = R * MIN(XYCOOD(1), XYCOOD(3))
      CALL GMOVA(XL, YL)
      CALL GCIRA(RL)
      RETURN
      END
```

```
      SUBROUTINE CONTUR (F, TEST, NX, NY, FL)
          PARAMETER (NPT=101)
          REAL     F(NX,NY), TEST(NX,NY)
          REAL     XT1(NPT), XT2(NPT), XT3(NPT), YT1(NPT), YT2(NPT),
     1             YT3(NPT), FT1(NPT), FT2(NPT), FT3(NPT)
          REAL     T00(NPT), T10(NPT), T01(NPT), T11(NPT)
C
C       SUBARRAYS OUT OF THE DATA CANNOT BE CONTOURED. THE
C      ENTIRE F(I,J) ARRAY IS CONTOURED SO A COPY OVER INTO SCRATCH SPACE
C      IS REQUIRED IF ONLY A PORTION OF F IS TO BE PLOTTED.
C
C      F      REAL ARRAY (NX,NY)  THE REAL VALUES OF THE FUNCTION TO
C                                 CONTOURED. A TOPOLOGICALLY SQUARE
C                                 GRID CELL IS ASSUMED AND DX = DY = 1.
C      TEST   REAL ARRAY (NX,NY)  A USER SUPPLIED SCRATCH ARRAY OF
C                                 THE SAME DIMENSIONALITY AS F.
C      NX     INTEGER             RANGE AND DIMENSION OF I IN F(I,J).
C      NY     INTEGER             RANGE AND DIMENSION OF J IN F(I,J).
C      FL     REAL                THE VALUE OF F(I,J) FOR CONTOURING.
C
C      CALCULATE THE AVERAGE VALUE OF F AT THE CENTERS OF THE CELLS.
C
          NTMAX = NPT - 2
          NTRIA = 0
          NXM = NX - 1
          NYM = NY - 1
          DO 200 J = 1, NYM
          DO 200 I = 1, NXM
          TEST(I,J) = F(I,J) + F(I+1,J)
          TEST(I,J) = TEST(I,J) + F(I,J+1)
          TEST(I,J) = TEST(I,J) + F(I+1,J+1)
  200     TEST(I,J) = 0.25*TEST(I,J)
C
C      NOW CALCULATE THE CROSSINGS WHICH PARELLEL THE "I" AXIS.
C
          DO 310 J = 1, NY
          DO 300 I = 1, NXM
  300     T00(I) = (F(I+1,J) - FL)*(FL - F(I,J))
C
C      A CROSS OCCURS IF THE CONTOUR PASSES THRU F(I,J) IN THE BOX I->I+1
C
          DO 309 I = 1, NXM
          IF (T00(I) .LT. 0.0) GO TO 309
C
C      THE LINE SEGMENT IS A HIT. TREAT IN A SCALAR WAY FIRST THE UPPER
C      AND THEN THE LOWER TRIANGLE.
C
          IF (J.EQ.NY) GO TO 305
          IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1        XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
          NTRIA = NTRIA + 1
          XT1(NTRIA) = FLOAT(I)
          YT1(NTRIA) = FLOAT(J)
          FT1(NTRIA) = F(I,J)
          XT2(NTRIA) = FLOAT(I + 1)
```

```
      YT2(NTRIA) = FLOAT(J)
      FT2(NTRIA) = F(I+1,J)
      XT3(NTRIA) = FLOAT(I) + 0.5
      YT3(NTRIA) = FLOAT(J) + 0.5
      FT3(NTRIA) = TEST(I,J)
  305 IF (J.EQ.1) GO TO 309
      IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1    XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
      NTRIA = NTRIA + 1
      XT1(NTRIA) = FLOAT(I + 1)
      YT1(NTRIA) = FLOAT(J)
      FT1(NTRIA) = F(I+1,J)
      XT2(NTRIA) = FLOAT(I)
      YT2(NTRIA) = FLOAT(J)
      FT2(NTRIA) = F(I, J)
      XT3(NTRIA) = FLOAT(I) + 0.5
      YT3(NTRIA) = FLOAT(J) - 0.5
      FT3(NTRIA) = TEST(I,J-1)
  309 CONTINUE
  310 CONTINUE
C
C     NEXT CALCULATE THE CROSSINGS ALONG THE "J" AXIS.
C
      DO 360 J = 1, NYM
      DO 350 I = 1, NX
  350 T11(I) = (F(I,J+1) - FL)*(FL - F(I,J))
C
C     CORSS OCCURS IF THE CONTOUR VALUE PASSES THRU THE BOX IN F(I,J)
C     IN THE RANGE J->J+1
C
      DO 359 I = 1, NX
      IF (T11(I) .LT. 0.0) GO TO 359
C
C     THE LINE SEGMENT IS A HIT. TREAT IN A SCALAR WAY FIRST THE RIGHT
C     AND THEN THE LEFT TRIANGLE.
C
      IF (I.EQ.NX) GO TO 355
      IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1    XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
      NTRIA = NTRIA + 1
      XT1(NTRIA) = FLOAT(I)
      YT1(NTRIA) = FLOAT(J + 1)
      FT1(NTRIA) = F(I,J+1)
      XT2(NTRIA) = FLOAT(I)
      YT2(NTRIA) = FLOAT(J)
      FT2(NTRIA) = F(I,J)
      XT3(NTRIA) = FLOAT(I) + 0.5
      YT3(NTRIA) = FLOAT(J) + 0.5
      FT3(NTRIA) = TEST(I,J)
  355 IF (I.EQ.1) GO TO 359
      IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1    XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
      NTRIA = NTRIA + 1
      XT1(NTRIA) = FLOAT(I)
      YT1(NTRIA) = FLOAT(J)
```

```
            FT1(NTRIA) = F(I,J)
            XT2(NTRIA) = FLOAT(I)
            YT2(NTRIA) = FLOAT(J + 1)
            FT2(NTRIA) = F(I,J+1)
            XT3(NTRIA) = FLOAT(I) - 0.5
            YT3(NTRIA) = FLOAT(J) + 0.5
            FT3(NTRIA) = TEST(I-1,J)
   359      CONTINUE
   360      CONTINUE
C
C      NOW SEEK ALL TRIANGLES WITH TWO DIAGONAL CROSSINGS.
C
            DO 490 J = 1, NYM
            DO 420 I = 1, NXM
            T00(I) = (TEST(I,J) - FL)*(FL - F(I,J))
            T10(I) = (FL - TEST(I,J))*(F(I+1,J) - FL)
            T01(I) = (FL - TEST(I,J))*(F(I,J+1) - FL)
   420      T11(I) = (TEST(I,J) - FL)*(FL - F(I+1,J+1))
C
C      CONSIDER THE LOWER TRIANGLE IN THE SQUARE.
C
            DO 430 I = 1, NXM
            IF (AMIN1(T00(I), T10(I)) .LT. 0.0) GO TO 430
            IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1         XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
            NTRIA = NTRIA + 1
            XT1(NTRIA) = FLOAT(I) + 0.5
            YT1(NTRIA) = FLOAT(J) + 0.5
            FT1(NTRIA) = TEST(I,J)
            XT2(NTRIA) = FLOAT(I)
            YT2(NTRIA) = FLOAT(J)
            FT2(NTRIA) = F(I,J)
            XT3(NTRIA) = FLOAT(I + 1)
            YT3(NTRIA) = FLOAT(J)
            FT3(NTRIA) = F(I+1,J)
   430      CONTINUE
C
C      CONSIDER THE RIGHT TRIANGLE IN THE SQUARE.
C
            DO 440 I = 1, NXM
            IF (AMIN1(T10(I), T11(I)) .LT. 0.0) GO TO 440
            IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1         XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
            NTRIA = NTRIA + 1
            XT1(NTRIA) = FLOAT(I) + 0.5
            YT1(NTRIA) = FLOAT(J) + 0.5
            FT1(NTRIA) = TEST(I,J)
            XT2(NTRIA) = FLOAT(I + 1)
            YT2(NTRIA) = FLOAT(J)
            FT2(NTRIA) = F(I+1,J)
            XT3(NTRIA) = FLOAT(I + 1)
            YT3(NTRIA) = FLOAT(J + 1)
            FT3(NTRIA) = F(I+1,J+1)
   440      CONTINUE
C
```

```
C     CONSIDER THE UPPER TRIANGLE IN THE SQUARE.
C
          DO 450 I = 1, NXM
          IF (AMIN1(T11(I), T01(I)) .LT. 0.0) GO TO 450
          IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1         XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
          NTRIA = NTRIA + 1
          XT1(NTRIA) = FLOAT(I) + 0.5
          YT1(NTRIA) = FLOAT(J) + 0.5
          FT1(NTRIA) = TEST(I,J)
          XT2(NTRIA) = FLOAT(I + 1)
          YT2(NTRIA) = FLOAT(J + 1)
          FT2(NTRIA) = F(I+1,J+1)
          XT3(NTRIA) = FLOAT(I)
          YT3(NTRIA) = FLOAT(J + 1)
          FT3(NTRIA) = F(I,J+1)
  450     CONTINUE
C
C     CONSIDER THE LEFT TRIANGLE IN THE SQUARE.
C
          DO 460 I = 1, NXM
          IF (AMIN1(T01(I), T00(I)) .LT. 0.0) GO TO 460
          IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1         XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
          NTRIA = NTRIA + 1
          XT1(NTRIA) = FLOAT(I) + 0.5
          YT1(NTRIA) = FLOAT(J) + 0.5
          FT1(NTRIA) = TEST(I,J)
          XT2(NTRIA) = FLOAT(I)
          YT2(NTRIA) = FLOAT(J + 1)
          FT2(NTRIA) = F(I,J+1)
          XT3(NTRIA) = FLOAT(I)
          YT3(NTRIA) = FLOAT(J)
          FT3(NTRIA) = F(I,J)
  460     CONTINUE
  490     CONTINUE
C
C     IF THE BUFFER IS ONLY PARTIALLY FULL, FLUSH AT THE END.
C
          IF (NTRIA .GT. 0) CALL PROPOL (NTRIA, XT1,YT1,FT1,
     1         XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
C
          RETURN
      END
```

```
      SUBROUTINE DEVICE (N)
C
C     DEVICE          1 CALCOMP PLOTTER
C                     2 PRINTER PLOT
C                     3 LEXIDATA DISPLAY UNIT WITH MATRIX CAMERA
C                     4 ADM WITH RETROGRAPHICS OR TEK 40XX SERIES
C                     5 EMPTY SLOT
C
C     COMMANDS            DEVICE - SELECT DEVICE (1-5)
C                         SETDEV - SET THE LOGICAL UNITS FRO OUTPUT
C                         LINWID - NUMBER OF STROKES IN A LINE
C                         FILTYP - TYPE OF FILLING FOR POLYGONS AND CIRCL
C                         CHRSIZ - HEIGHT OF CHARACTERS (IN RASTER UNITS)
C
C                     1   NEWFRM - SET UP DEVICE FOR A NEW PLOT
C                     2   FRAME  - FORCE A FLUSH OF ALL BUFFERS
C                     3   ERASE  - ERASE THE SCREEEN
C                     4   HDCOPY - HARD COPY COMMAND - SHUTTER ON MATRIX
C                     5   COLOR  - SELECT THE COLOR TO DRAW
C                     6   SCALE  - SCALE FOR PHYSICAL DEVICE
C                     7   ENDFRM - TERMINATE THE DEVICE AND ADVANCE
C
C     DEFAULT DEVICE ASSIGNMENTS
C
C                     TYPE                LU    RESULT
C
C                     DIAGNOSTICS         0     NO OUTPUT
C                     GRAPHICS            7     NORMAL - ASSIGNED
C                     CAMERA              8     ASSIGNED WITH LEXIDATA
C                     CHARACTER SET       0     UNIT 0 - ASSIGNED
C                     CALCOMP             L7:   RS232 LINE
C                     PRINTER             PR:   WHATEVER
C                     LEXIDATA            LEX:  DMA (L34DVR IN SYSTEM)
C                     MATRIX CAMERA       LE:   RS232
C                     TEKTRONIX           C:    CONSOLE
C                     OPEN SLOT           NULL: BIT BUCKET
C                     CONSOLE INPUT       5
C                     CHARACTER SET       9 (CLOSED THEN OPENED)
C                     TABLET INPUT        8 (CLOSED THEN OPENED -
C                                           CAMERA AND TABLET CAN
C                                           NOT BE ACTIVE SIMULTANEOUSLY)
C
      INTEGER  LB, COUNT(5), NFRAME(5), PEN, COMMND, DEVID, PBLK(6)
      CHARACTER*8 IDFLTO(5), MATFIL, TABFIL
      INTEGER ICODET(2), ICTRLX(2), ICHANX(2,2), SCRATCH(60)
      INTEGER HBLANK, HMINUS, HAPOSA, HAPOSB, HASTRA, HASTRB, NHOLD
      INTEGER VSNUM(2), TEKRAS(4), LEXRAS(4), HWSIZE(20), CALRAS(4)
      INTEGER LPAGE(30,65), START, LINE, SLASH, ENDIT
      INTEGER MATUNT, TABUNT, USPAT(12,4)
      LOGICAL LSW, LTSW, OPEN, OPENANY, OPNCAL, OPNLEX, IRVRSE
      LOGICAL OPNMAT, OPNTAB, OPENTB
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      COMMON/DEVPOS/X1R(5),Y1R(5),X2R(5),Y2R(5)
      COMMON/GRFMOD/LFLMAT(5),LINWDM(5),LCLMAT(5)
```

```
          EQUIVALENCE (HWSIZE(1),CALRAS),(HWSIZE(9),LEXRAS)
          EQUIVALENCE (HWSIZE(13),TEKRAS)
          DATA START,LINE,SLASH,ENDIT/4H(    , 4H108H, 4H   /, 4H   ) /
          DATA HBLANK, HMINUS, HAPOSA, HAPOSB, HASTRA, HASTRB
     1    /4H     , 4H---- , 4H¶    , 4H   ¶ , 4H*--- , 4H---* /
          DATA OPENANY/.FALSE./, OPNLEX/.TRUE./
          DATA ITIMET/20/, IDLETM/50/, LUDIAG/0/
          DATA COUNT/5*0/, NFRAME/5*0/, ANGLE/0.0/
          DATA TEKRAS/0,0,1022,768/,LEXRAS/50,50,32700,26160/
          DATA CALRAS/0, 0, 10750, 7900/
          DATA IDFLTO/'L7: ','PR: ','LEX:','C:','NULL:'/
          DATA X1R/0.,0.,1.E+7,0.,0./, X2R/2*1279.,11279000.,2*1279./
          DATA Y1R/0.,0.,1.E+7,0.,0./, Y2R/2*1023.,11023000.,2*1023./
          DATA LUSET/7/, ICODET/27,23/, ICTRLX/31,24/
          DATA ICHANX/27,97,27,127/, OPNCAL/.TRUE./
          DATA NFRAME/5*0/,COUNT/5*0/,LUHSET/9/, OPNMAT/.TRUE./
          DATA LFLMAT/0,0,1,0,0/, LINWDM/ 1,1,2,1,1/, LCLMAT/5*1/
          DATA MATUNT/8/, MATFIL/'L14:'/, TABFIL/'L19:'/, TABUNT/8/
          DATA XYCOOD/1.,0.,1.,0./, OPNTAB/.TRUE./
          DATA CHSIZE/31.,31.,0.3,0.5,0.04,0.58,0.,0.,.03/
          DATA USPAT/3*3640,3*455,3*3640,3*455,0,0,778,
     .    992,504,240,240,504,992,778,0,0,2*0,Z20,Z70,ZF8,
     .    Z1FC,Z3FE,Z7FF,4*0,
     .    0,96,240,504,1020,2046,4095,3999,3855,3591,3075,2049/
C
C     *** START OF THE SECTION WHICH HANDELS FUNCTIONS
C
C
C     DEVICE
C
          IF(OPENANY) GO TO 1100
          IDEVIC = N
          LUOUT = LUSET
          ITKWIT = 40
          IF (IDEVIC.LT.1.OR.IDEVIC.GT.5) IDEVIC = 4
          CLOSE(LUOUT)
          OPEN(UNIT=LUOUT,IOSTAT=IOS,ERR=901,FILE=IDFLTO(IDEVIC),
     1    TYPE='DEVICE',SHARE='ERW',RKEY=0,WKEY=0)
          OPENANY = .TRUE.
          IF (LUDIAG.NE.0) WRITE (LUDIAG,1003) IDEVIC
          RETURN
C
C     ERROR ON CALL TO DEVICE
C
  901     CLOSE(LUOUT)
          IF (LUDIAG.NE.0) WRITE (LUDIAG,1014) IDEVIC, IOS
C
C     SET A VALUE OF ZERO IN THE DEVICE TYPE TO INDICATE THAT
C     A DEVICE CAN NOT BE ACCESSED. ADDED 1/24/85
C
          IDEVIC = 0
          RETURN
C
C     SETDEV - CHANGE THE LOGICAL UNIT
C
```

```
      ENTRY SETDEV (N1, N2)
      IF (N1.GT.0) LUSET = N1
      IF (N2.GE.0) LUDIAG = N2
      IF (LUDIAG.NE.0) WRITE (LUDIAG, 1005) LUOUT, LUSET
      RETURN
C
C     NEWFRM
C
      ENTRY NEWFRM
      MODEP = 1
      IF (.NOT.OPENANY) GO TO 1000
      COUNT(IDEVIC) = COUNT(IDEVIC) + 1
      NFRAME(IDEVIC) = 1
      IRVRSE = .FALSE.
      IF (LUDIAG.NE.0) WRITE (LUDIAG, 1101) COUNT(IDEVIC)
      GO TO (1,2,3,4,5), IDEVIC
C
C     FRAME - FORCE A WRITE OF THE BUFFER
C
      ENTRY FRAME
      MODEP = 2
      IF (.NOT.OPENANY) GO TO 1000
      NFRAME(IDEVIC) = NFRAME(IDEVIC) + 1
      IF (LUDIAG.NE.0) WRITE (LUDIAG, 1001) NFRAME(IDEVIC),IDEVIC
      GO TO (1,2,3,4,5), IDEVIC
C
C     ERASE
C
      ENTRY ERASE
      MODEP = 3
      IF (.NOT.OPENANY) GO TO 1000
      IF (LUDIAG.NE.0) WRITE (LUDIAG,1501) NFRAME(IDEVIC)
      GO TO (1,2,3,4,5), IDEVIC
C
C     HARD COPY
C
      ENTRY HDCOPY
      MODEP = 4
      IF (.NOT.OPENANY) GO TO 1000
      IF (LUDIAG.NE.0) WRITE (LUDIAG,1502) NFRAME(IDEVIC)
      GO TO (1,2,3,4,5), IDEVIC
C
C     COLOR
C
      ENTRY COLOR (N)
      MODEP = 5
      IF(.NOT.OPENANY) GO TO 1000
      ICOLOR = N
      IF (LUDIAG.NE.0) WRITE (LUDIAG, 1214) ICOLOR
      GO TO (1,2,3,4,5), IDEVIC
C
C     SET THE TYPE FOR POLYGON AND CIRCLE FILLING 0, 1, 2, OR 3
C
      ENTRY FILTYP(IFIL)
      IF(.NOT.OPENANY) GO TO 1000
```

```
            LFLMAT(IDEVIC) = IFIL
            IF(LUDIAG.GT.0) WRITE(LUDIAG,1503) LFLMAT
            MODEP = 5
            GO TO (1,2,3,4,5), IDEVIC
C
C     SET THE SIZE OF THE HARDWARE CHARACTERS
C
            ENTRY CHRSIZ (CHFRZ, GCHFRZ)
            IF(.NOT.OPENANY) GO TO 1000
            IF(CHFRZ.GT.0) CHSIZE(9) = CHFRZ
            CHSIZE(1) = CHSIZE(9) * (Y2R(IDEVIC)-Y1R(IDEVIC))
            CHSIZE(2) = CHSIZE(1)
            IF(GCHFRZ.GT.0) CHSIZE(5) = GCHFRZ
            IF(LUDIAG.GT.0) WRITE(LUDIAG,1505) CHFRZ, GCHFRZ
            MODEP = 5
            RETURN
C
C     SET THE LINE WIDTH
C
            ENTRY LINWID(LINWD)
            IF(.NOT.OPENANY) GO TO 1000
            LINWDM(IDEVIC) = MAX0(LINWD,1)
            IF(LUDIAG.GT.0) WRITE(LUDIAG,1504) LINWDM
            MODEP = 5
            GO TO (1,2,3,4,5), IDEVIC
C
C     SCALE
C
            ENTRY SCALNG(X1,Y1,X2,Y2,X1H,Y1H,X2H,Y2H,X1S,Y1S,X2S,Y2S)
            IF(.NOT.OPENANY) GO TO 1000
            IND = (IDEVIC-1)*4
            IF(IDEVIC.EQ.1) THEN
              SCLRAS = 1000.
            ELSE
              SCLRAS = 1.
            ENDIF
            IF (X1H.GE.0.0) HWSIZE(IND+1) = X1H * SCLRAS
            IF (Y1H.GT.0.0) HWSIZE(IND+2) = Y1H * SCLRAS
            IF (X2H.GE.0.0) HWSIZE(IND+3) = X2H * SCLRAS
            IF (Y2H.GT.0.0) HWSIZE(IND+4) = Y2H * SCLRAS
            IF (X1S.NE.0.0) X1R(IDEVIC) = X1S
            IF (X2S.NE.0.0) X2R(IDEVIC) = X2S
            IF (Y1S.NE.0.0) Y1R(IDEVIC) = Y1S
            IF (Y2S.NE.0.0) Y2R(IDEVIC) = Y2S
            GO TO 11
            ENTRY DEFINE (X1, Y1, X2, Y2)
   11       MODEP = 6
            IF(.NOT.OPENANY) GO TO 1000
            DX = X2 - X1
            IF (DX.EQ.0.0) DX = 1.0
            XYCOOD(1) = (X2R(IDEVIC) - X1R(IDEVIC))/DX
            XYCOOD(2) = X1R(IDEVIC) - XYCOOD(1)*X1
            DY = Y2 - Y1
            IF (DY.EQ.0.0) DY = 1.0
            XYCOOD(3) = (Y2R(IDEVIC) - Y1R(IDEVIC))/DY
```

```
      XYCOOD(4) = Y1R(IDEVIC) - XYCOOD(3)*Y1
      IF (LUDIAG.NE.0) WRITE (LUDIAG, 1400) XYCOOD
      CHSIZE(1) = CHSIZE(9) * (Y2R(IDEVIC)-Y1R(IDEVIC))
      CHSIZE(2) = CHSIZE(1)
      GO TO (1,2,3,4,5), IDEVIC
C
C    ENDFRM - TO CLOSE THE LOGICAL UNIT
C
      ENTRY ENDFRM
      IF(.NOT.OPENANY) GO TO 1000
      MODEP = 7
      IF (LUDIAG.NE.0) WRITE (LUDIAG,1201) COUNT(IDEVIC),
     .    NFRAME(IDEVIC)
      NFRAME(IDEVIC) = 0
      GO TO (1,2,3,4,5), IDEVIC
   10 CLOSE(7)
      OPENANY = .FALSE.
      IDEVIC = 0
      RETURN
C
C    *** START OF THE SECTION WHICH HANDELS THE HARDWARE ***
C
C
C    CALCOMP SECTION
C
    1 GO TO (101,102,103,103,105,106,107), MODEP
C
C    CALCOMP INITIALIZATION
C
  101 CALL CALBUF (LUOUT,0,1,IDXX)
      IF(OPNCAL) CALL CALPLT (0.0, +1.0, 1007)
      OPNCAL = .FALSE.
      CALL CALPLT (FLOAT(CALRAS(3))/((X2R(1)-X1R(1))*1000.),
     .  FLOAT(CALRAS(4))/((Y2R(1)-Y1R(1))*1000.), 1001)
      CALL CALPEN (1)
      LSW = .TRUE.
      RETURN
C
C    CALCOMP FRAME (FLUSH BUFFER AND RESET, IGNORE HARDCOPY)
C
  102 RETURN
C
C    ERASE THE SCREEN AND PRINT HARDCOPY ARE IGNORED FOR THE CALCOMP
C
  103 RETURN
C
C    COLOR SELECTION PRINTOUT.
C
  105 PEN = MOD(ICOLOR-1,4) + 1
      PEN = MIN0(MAX0(PEN,1),4)
      LCLMAT(IDEVIC) = PEN
      CALL CALPEN(PEN)
      RETURN
C
C    CALCOMP AXIS SET.
```

```
C
  106     CONTINUE
          CALL CALPLT (FLOAT(CALRAS(3))/((X2R(1)-X1R(1))*1000.),
     .     FLOAT(CALRAS(4))/((Y2R(1)-Y1R(1))*1000.), 1001)
          RETURN
C
C     ENDFRM FOR CALCOMP
C
  107     CONTINUE
          CALL CALPLT(0., 0., 999)
          GO TO 10
C
C     PAPER PLOT SECTION
C
    2     GO TO (201,202,203,204,205,206,207), MODEP
C
C     PAPER PLOT INITIALIZATION.
C     THE PAGE PLOT IS OUTPUT AS ONE LARGE VARIABLE FORMAT.
C     SET UP THE FORMAT STATEMENT IN LPAGE.
C
  201     CONTINUE
  203     DO 211 J=1,65
          LPAGE(2,J)=LINE
  211     LPAGE(30,J)=SLASH
          LPAGE(1,1)=START
          LPAGE(30,65)=ENDIT
C
C     SET UP THE PAGE AND CLEAN THE WORK AREA.
C
          DO 221 J = 1, 65
          DO 221 I = 4, 29
  221     LPAGE(I,J) = HBLANK
          DO 222 I = 5, 28
          LPAGE(I,1) = HMINUS
  222     LPAGE(I,63) = HMINUS
          DO 223 J = 2, 62
          LPAGE(4,J) = HAPOSA
  223     LPAGE(29,J) = HAPOSB
          LPAGE(29,1) = HASTRB
          LPAGE(29,63) = HASTRB
          LPAGE(4,63) = HASTRA
          LPAGE(4, 1) = HASTRA
          RETURN
C
C     PAPER PLOT- WRITE OUT THE PAGE AS IT IS.
C
  202     WRITE(LUOUT, LPAGE)
          RETURN
C
C     HARDCOPY COMMAND
C
  204     RETURN
C
C     COLOR SELECTION PRINTOUT.
C
```

```
  205     RETURN
C
C     SET SCALING FOR PAGE PLOT
C
  206     RETURN
C
C     ENDFRM FOR THE PAPER PLOTTER
C
  207     WRITE (LUOUT,1212)
          GO TO 10
C
C     LEXIDATA/MATRIX SECTION
C
    3     GO TO (301,302,303,304,305,306,307), MODEP
C
C     INITIALIZE THE LEXIDATA/MATRIX - OPEN AND SET THE LUT ONLY
C     IF THE DEVICE IS/WAS CLOSED
C
  301     IF(OPNLEX) THEN
          CALL GSOPN(LUOUT, 1, LIERR)
          IF(LIERR.NE.0) GO TO 1002
C
C     CURSOR OFFSET - THIS IS INSTALLATION SPECIFIC
C
          CALL DSCSL(10, 153, 75)
          CALL DSCER
          CALL SETLUT ; SET THE DEFAULT LOOK UP TABLE
          CALL GSTYPE(3)
C
C     SET VIEW CORRESPONDENCE
C
          CALL GDEFVS (1, 15, 0, 1, 4, 0)
          CALL GINTVS (1)
          CALL GACTVS (1)
          CALL GDFWIN (1, X1R(IDEVIC), Y1R(IDEVIC), X2R(IDEVIC),
     .      Y2R(IDEVIC))
          CALL GDFVP  (1, LEXRAS(1), LEXRAS(2), LEXRAS(3), LEXRAS(4))
          VSNUM(1) = 2
          VSNUM(2) = 0
          CALL GDFVU  (1, 1, 1, VSNUM)
          CALL GACTVU(1)
          CALL GDASEG
          CALL GCRSEG(100)
          DO 3001 I = 1, 4
 3001     CALL GDPAT(100+I, USPAT(1,I))
          CALL GCLSEG
          CALL GSVUAS(100, VSNUM)
          CALL GDLSEG(100)
          CALL GSIVIS(1)
          CALL GSVUAS (0, VSNUM)
C
C  DEFINE LOCATOR PORT EXTENTS AND SET TRACKING TYPE
C
          CALL GDVINI(DEVID, 2, 1)
          CALL GDFLPT(DEVID, LEXRAS(1),LEXRAS(2),LEXRAS(3),LEXRAS(4))
```

```
                CALL GSTTRK(DEVID, 2, 1)
C
C  SET ECHO TYPE
C
                CALL GSTDVE(DEVID, 1, 1)
                CALL GDVENB(DEVID)
                CALL GRSLG (DEVID, 255)
                OPNLEX = .FALSE.
                ENDIF
C
C      SET THE PARAMETERS FOR THE LEXIDATA DISPLAY.
C
                CALL GCLSEG
                CALL DSCLR(-1)
                CALL GCTSEG
                CALL GSFTYP(LFLMAT(IDEVIC))
                CALL GSLWID(LINWDM(IDEVIC))
                CALL GSCHSZ(CHSIZE(2))
                CALL GSCNDX (1)
                RETURN
C
C      FRAME - FORCE OUT THE CONTENTS OF THE BUFFER
C
   302          CALL GCLSEG
                CALL GMPCUR
                CALL GCTSEG
                RETURN
C
C      ERASE THE SCREEN AND THE CURSOR
C
   303          CALL DSCLR(-1)
                CALL DSCSL (10, 153, 75)
                RETURN
C
C      HARD COPY - CAMERA
C
   304          IF(OPNMAT) THEN
                  CLOSE(MATUNT)
                  OPNTAB = .TRUE.
                  OPEN(UNIT=MATUNT,IOSTAT=IOS,ERR=314,FILE=MATFIL,
     .            TYPE='DEVICE',SHARE='ERW',RKEY=0,WKEY=0)
                  OPNMAT = .FALSE.
                ENDIF
                IF(LUDIAG.NE.0) WRITE(LUDIAG,1003) MATUNT
                CALL SYSIO(PBLK,41,MATUNT,'.CE.',4,0)
                CALL WAIT(ITIMET,2,IOS)
                RETURN
C
C      ERROR ON CALL TO DEVICE
C
   314          CLOSE(MATUNT)
                IF (LUDIAG.NE.0) WRITE (LUDIAG,1014) MATUNT, IOS
                STOP 'camera can not be accessed'
C
C      SET COLOR - THIS APPLIES TO WHAT IS IN THE LUT
```

```
C
  305     LEXCOL = MOD(ICOLOR-1,15) + 1
          LEXCOL = MAXO(LEXCOL,1)
          LCLMAT(IDEVIC) = LEXCOL
          CALL GSCNDX(LCLMAT(IDEVIC))
          CALL GSLWID(LINWDM(IDEVIC))
          LFLLEX = LFLMAT(IDEVIC)
          IF(LFLLEX.LE.4) THEN
            CALL GSFTYP(LFLLEX)
          ELSE
            CALL GSFTYP(96+LFLLEX)
          ENDIF
          RETURN
C
C     SET THE WINDOW AND AREA OF NORMALIZED DEVICE COORDINATES
C
  306     IF(OPNLEX) GO TO 301
          CALL GCLSEG
          CALL GDAVU(1)
          CALL GDFWIN (1, X1R(IDEVIC), Y1R(IDEVIC), X2R(IDEVIC),
     .      Y2R(IDEVIC))
          CALL GDFVP  (1, LEXRAS(1), LEXRAS(2), LEXRAS(3), LEXRAS(4))
          CALL GDFVU  (1, 1, 1, VSNUM)
          CALL GACTVU (1)
C
C     OPEN A NEW SEGMENT
C
          CALL GCTSEG
          RETURN
C
C
  307     OPNLEX = .TRUE.
          IF (.NOT.OPNMAT) THEN
            CLOSE(MATUNT)
            OPNMAT = .TRUE.
          ENDIF
          CALL GCLSEG
          CALL GDASEG
          CALL GFRAME
          CALL DSCLR(-1)
          CALL GRSLG(DEVID,255)
          CALL GDVDSB(DEVID)
          GO TO 10
C
C     TEKTRONIX SECTION
C
    4     GO TO (401,402,403,404,405,406,407), MODEP
C
C     TEKTRONIX INITIALIZATION
C
  401     CALL TEKINT (120)
          CALL SWINDO (TEKRAS(1),TEKRAS(3),TEKRAS(2),TEKRAS(4))
          CALL VWINDO (X1R(IDEVIC), X2R(IDEVIC)-X1R(IDEVIC),
     .                 Y1R(IDEVIC), Y2R(IDEVIC)-Y1R(IDEVIC))
          LTSW = .TRUE.
```

```
          CALL TEKHOM
          CALL TTKSND
          CALL WAIT (IDLETM, 1, IS)
          RETURN
C
C     FRAME (NO MODE SWITCHING)
C
  402     CONTINUE
          CALL TTKSND
          CALL WAIT (IDLETM, 1, IS)
          RETURN
C
C     ERASE THE SCREEN
C
  403     CONTINUE
          CALL TEKHOM
          CALL TEKERA
          CALL TOUTST (2, ICTRLX)
          CALL TTKSND
          CALL WAIT (IDLETM, 1, IS)
          RETURN
C
C     ISSUE HARD COPY COMMAND
C
  404     CONTINUE
          CALL TOUTST (2, ICODET)
          CALL TEKHOM
          CALL TTKSND
          CALL WAIT (ITIMET, 2, IS)
          RETURN
C
C     COLOR - DOES NOT APPLY TO ADM OR TEKTRONIX 4054
C
  405     CONTINUE
          RETURN
C
C     SET SCREEN CORRESPONDENCE
C
  406     CONTINUE
          CALL SWINDO(TEKRAS(1),TEKRAS(3),TEKRAS(2),TEKRAS(4))
          CALL VWINDO (X1R(IDEVIC), X2R(IDEVIC)-X1R(IDEVIC),
     .                 Y1R(IDEVIC), Y2R(IDEVIC)-Y1R(IDEVIC))
          RETURN
C
C     END THE FRAME ON THE ADM OR TEK 4054
C
  407     CALL TEKHOM
          CALL TOUTST (2, ICTRLX)
          CALL TTKSND
          CALL WAIT (IDLETM, 1, IS)
          CALL SVSTAT (SCRATCH)
          GO TO 10
C
C     SLOT FOR AN EXTRA DEVICE
C
```

```
      5     STOP 74
C
C     DELAY IN SECONDS FOR HARD COPY UNITS
C
            ENTRY DELAY(ITIME)
            ITIMET = ITIME
            RETURN
C
C     IOWAIT
C
            ENTRY IOWAIT(IT)
            CALL WAIT (IT, 1, IS)
            RETURN
C
C     GET THE DEVICE ID FOR THE LEXIDATA TRACKING TYPE
C
            ENTRY LEXDID(IDVL)
            IDVL = DEVID
            RETURN
C
C     OPEN ACCESS TO THE TABLET - IF POSSIBLE
C
            ENTRY TABLET (OPENTB)
            OPENTB = .FALSE.
            IF(OPNTAB) THEN
              IF(.NOT.OPNMAT) CLOSE(MATUNT)
              OPEN(UNIT=TABUNT,IOSTAT=IOS,ERR=1304,FILE=TABFIL,
     .                               TYPE='DEVICE',SHARE='ERW',RKEY=0,
     .                               WKEY=0)
              OPNTAB = .FALSE.
              OPENTB = .TRUE.
              OPNMAT = .TRUE.
            ELSE
              IF(LUDIAG.GT.0) WRITE(LUDIAG,1302)
              OPENTB = .TRUE.
            ENDIF
            RETURN
C
C     THE TABLET HAS GENERATED AN ERROR
C
   1304     CLOSE(TABUNT)
            IF(LUDIAG.GT.0) WRITE(LUDIAG,1303) TABUNT,IOS
            STOP 'can not access tablet'
C
C     PUNISH IF NOT INITIALIZED.
C
   1000     IF (LUDIAG.NE.0) WRITE (LUDIAG, 1004) MODEP, IDEVIC
            IDEVIC = 0
            RETURN
   1002     IF (LUDIAG.NE.0) WRITE(LUDIAG, 1013)LIERR
            STOP 'Lexidata can not be accessed'
   1100     IF (LUDIAG.NE.0) WRITE(LUDIAG, 1006) IDEVIC
            RETURN
C
C     FORMATS
```

```
C
 1001     FORMAT (' FRAMES PLOTTED = ',I5,' ON DEVICE ',I3)
 1003     FORMAT (' DEVICE = ',I2,' INITIALIZED')
 1004     FORMAT (' DEVICE NOT INITIALIZED. ', 2I5)
 1005     FORMAT (' LUOUT = ',I3,' AND WILL BE SET TO ',I3,
     .    ' AT THE NEXT CALL TO "DEVICE"')
 1006     FORMAT (' DEVICE ',I3,' IS ALREADY OPEN')
 1013     FORMAT (' CAN NOT OPEN LEXIDATA, ERROR =',I3)
 1014     FORMAT (' DEVICE ',I3,' CAN NOT BE ACESSED, ERROR =',I3)
 1101     FORMAT (' FRAME NUMBER ',I5,' INITIALIZED')
 1201     FORMAT (' PLOTTER CLOSED WITH',2I5,' FRAMES')
 1212     FORMAT ('1')
 1214     FORMAT (' COLOR SELECT = ',I3)
 1302     FORMAT(' CAMERA IS ALREADY AVAILABLE')
 1303     FORMAT (' CAMERA ',I3,' CAN NOT BE ACESSED, ERROR =',I3)
 1400     FORMAT (' FACTOR ',4F8.2)
 1501     FORMAT (' ERASE COMMAND ISSUED AT FRAME ',I5)
 1502     FORMAT (' HARDCOPY COMMAND ISSUED AT FRAME ',I5)
 1503     FORMAT (' SET THE FILL TYPE FOR SURFACES ',5I3)
 1504     FORMAT (' SET THE LINE WIDTH ',5I3)
 1505     FORMAT (' SET THE CHARACTER SIZE (%) ',2F6.3)
          END
```

```
      SUBROUTINE DEVINP(INPUT,BTMASK,IDV,X,Y,Z)
      LOGICAL JFIRST/.TRUE./, TFIRST/.TRUE./, KFIRST/.TRUE./
      INTEGER BTMASK,DEVID,PBLK(6),TLOOK(4)/1,2,4,8/
      CHARACTER*1 INST(2)/Z1B,'A'/
      CHARACTER*20 DATAIN
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/TXXDAT/TABIN,TABWIN,TABXYC
      REAL TABIN(2),TABWIN(4),TABXYC(4)
      COMMON/DEVPOS/X1R(5),Y1R(5),X2R(5),Y2R(5)
C
C     SELECT THE INPUT FROM THE DEVICE CODE
C
      X = 0.
      Y = 0.
      Z = 0.
      IDV = 0
      IF(INPUT.LT.1.OR.INPUT.GT.3) RETURN
      IF(INPUT.NE.1.AND.(.NOT.JFIRST)) THEN
        CALL DSCSL (10, 153, 75)
        JFIRST = .TRUE.
      ENDIF
      GO TO (10, 20, 30), INPUT
C
C     GET THE DEVICE ID FOR THE JOYSTICK
C
   10 IF(JFIRST) THEN
        CALL LEXDID(DEVID)
        CALL GDFBGF(DEVID, 15, 0, IDUM)
        CALL DSCSL (8, 153, 75)
        JFIRST = .FALSE.
      ENDIF
C
C     SET APPROPRIATE LIGHTS
C
      CALL GSLTG(DEVID,BTMASK)
C
C     READ THE LOCATOR
C
      XWC = 0.0
      YWC = 0.0
    3 CALL GRBTG(DEVID, IDV)
      IDV = IAND(IAND(IDV,BTMASK),15)
      IF(IDV.EQ.0) GO TO 3 ; NO SWITCH HAS BEEN TOGGLED
      CALL GSLC(DEVID, IDVV, LCX, LCY, XWC, YWC)
      IF(LCX.LT.0) GO TO 3
      CALL GRSLG (DEVID, 255)
    2 CALL GRBTG (DEVID, IDVV)
      IF(IDVV.NE.0) GO TO 2
C
C     NORMALIZE TO <RE>WINDOWED COORDINATES
C
      X = (XWC-XYCOOD(2)) / XYCOOD(1)
      Y = (YWC-XYCOOD(4)) / XYCOOD(3)
      Z = 0.0
      RETURN
```

```
C
C     THE TABLET HAS BEEN SELECTED
C
   20   IF(TFIRST) THEN
          TFIRST = .FALSE.
          CALL TABLET(OPENTB)
          IF(IDEVIC.EQ.0) THEN
            DX = 1280.
            DY = 1024.
          ELSE
            DX = (X2R(IDEVIC)-X1R(IDEVIC)) / XYCOOD(1)
            DY = (Y2R(IDEVIC)-Y1R(IDEVIC)) / XYCOOD(3)
          ENDIF
          TABXYC(1) = 8.2034E-5 * DX
          TABXYC(2) = 0.0
          TABXYC(3) = 1.0938E-4 * DY
          TABXYC(4) = 0.0
        ENDIF
   21   CALL SYSIO(PBLK,72,8,DATAIN,20,0)
        READ(DATAIN,22,ERR=21) IX,IY,IBT
   22   FORMAT(1X,I5,2X,I5,1X,I2)
        IF(IBT.EQ.0) GO TO 21
C
C     CONVERT THE CURSOR BUTTON TO A STANDARD FORMAT
C
        IDV = TLOOK(IBT)
        IDV = IAND(BTMASK, IDV)
        IF(IDV.EQ.0) GO TO 21
        TABIN(1) = IX
        TABIN(2) = IY
        X = FLOAT(IX)*TABXYC(1) + TABXYC(2)
        Y = FLOAT(IY)*TABXYC(3) + TABXYC(4)
        Z = 0.0
        RETURN
C
C     INPUT FROM THE KEYBOARD
C
   30   CALL SYSIO (PBLK,41,5,'X,Y,Z,IDV = ',12,0)
        READ(5,32,END=33,ERR=30) XCOR,YCOR,ZCOR,IDV
   32   FORMAT(3F,I)
        X = XCOR
        Y = YCOR
        Z = ZCOR
        IDV = IAND(BTMASK,IDV)
        RETURN
   33   IDV = BTMASK
        RETURN
        END
```

```
      SUBROUTINE ENUMBR (REALNO, XR1, YR1, XR2, YR2)
C
C*************************************************************************
C     ENUMBR
C*************************************************************************
C
         DX = (XR2 - XR1)/10.0
         THENO = REALNO
         EXPON = 0.001
         IF (REALNO .EQ. 0.0) GO TO 10
   12    IF (ABS(THENO) .LT. 10.0) GO TO 11
         EXPON = EXPON + 1.0
         THENO = THENO*0.1000001
         GO TO 12
   11    IF (ABS(THENO) .GE. 1.0) GO TO 10
         EXPON = EXPON - 1.0
         THENO = THENO*9.99999
         GO TO 11
   10    CALL FNUMBR (XR1, YR1, DX, 0.0, 0.7*DX, 0.0, YR2-YR1,
     1      THENO, 5, 2)
         CALL WDDRAW (XR1+5.0*DX, YR1, DX, 0.0, 0.7*DX, 0.0,
     1      YR2-YR1, '*10¶.')
         CALL FNUMBR (XR1+8.0*DX, (YR1+YR2)/2.0, 0.66*DX, 0.0,
     1      0.75*DX, 0.0, 0.64*(YR2-YR1), EXPON, 3, 0)
C    1      0.50*DX, 0.0, 0.6*(YR2-YR1), EXPON, 3, 0)
         RETURN
      END
```

```
      SUBROUTINE FNUMBR (XSTART, YSTART, DX, DY, SX, SXY, SY,
     1    NUMBR, WIDTH, DIGITS)
C
C       X       REAL    - X COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
C                         CHARACTER TO BE DRAWN
C       Y       REAL    - Y COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
C                         CHARACTER TO BE DRAWN
C       DX      REAL    - INCREMENT TO BE ADDED TO THE X COORDINATE FOR
C                         EACH CHARACTER DRAWN
C       DY      REAL    - INCREMENT TO BE ADDED TO THE Y COORDINATE FOR
C                         EACH CHARACTER DRAWN
C       SX      REAL    - X MATH SPACE SIZE FOR CHARACTERS
C       SXY     REAL    - SLANT MODIFIER FOR CHARACTERS
C       SY      REAL    - Y MATH SPACE SIZE FOR CHARACTERS
C       RNUMB   REAL    - NUMBER TO BE PLOTTED IN F FORMAT
C       WIDTH   INTEGER - TOTAL WIDTH OF FIELD INCLUDING DECIMAL POINT
C       DIGITS  INTEGER - NUMBER TO FRACTIONAL PLACES TO BE DRAWN
C
      REAL NUMBER, NUMBR, TEST(6)
      INTEGER    WIDTH, DIGITS, NUM, NN(22)
      LOGICAL  SPACE
      INTEGER   WORD(7), HASTR, HTEMP, SFIL(3)
      INTEGER TEXT(13)
      CHARACTER*1 WWORD(28), BLANK, HMINUS
      DATA TEXT /Z00000030, Z00000031, Z00000032, Z00000033,
     1           Z00000034, Z00000035, Z00000036, Z00000037,
     2           Z00000038, Z00000039, Z00000020, Z0000002E,
     3           Z0000007C /
      DATA SFIL / '*¶..', '**¶.', '***¶' /
      DATA HMINUS, HASTR / '-   ', '****' /, BLANK/' '/
      EQUIVALENCE (WORD(1),TEST(1),WWORD(1))
C
      I1 = 1
      I2 = 1
      NUMBER = ABS(NUMBR)
      IF (DIGITS.EQ.0)  NUM = NUMBER + 0.5
      IF (DIGITS.EQ.0)  GO TO 5
C
      WIDTH = WIDTH - 1
      NUM = NUMBER*10**DIGITS +0.5
    5 DO 10  I = 1,WIDTH
      J = WIDTH - I + 1
      NN(J) = NUM - 10*(NUM/10)
C
C     BREAK OUT THE DIGITS.
C
      NUM = NUM/10
   10 CONTINUE
      J = WIDTH - DIGITS
      IF(J .NE. 0) GO TO 18
      WORD(1) = TEXT(11)
      I1 = I1 + 1
      GO TO 25
   18 SPACE = .TRUE.
      DO 20  K = 1,J
```

```
            I = NN(K)
C
C       LEADING ZERO = SPACE
C
            IF (I.NE.0)  SPACE = .FALSE.
            IF(K .EQ. J .AND. SPACE) SPACE = .FALSE.
            IF (I.EQ.0.AND.SPACE)  I = 10
            I1 = I1 + 1
            HTEMP = ISHFT(WORD(I2), 8)
            WORD(I2) = IOR(HTEMP,TEXT(I+1))
            IF(I1 .LT. 5) GO TO 20
            I2 = I2 + 1
            I1 = 1
   20       CONTINUE
            IF (DIGITS.EQ.0)  GO TO 40
C
C       PUT IN DECIMAL POINT
C
   25       I1 = I1 + 1
            HTEMP = ISHFT(WORD(I2), 8)
            WORD(I2) = IOR(HTEMP,TEXT(12))
            IF(I1 .LT. 5) GO TO 26
            I2 = I2 + 1
            I1 = 1
   26       DO 30  K = 1,DIGITS
            I = NN(J+K)
            I1 = I1 + 1
            HTEMP = ISHFT(WORD(I2), 8)
            WORD(I2) = IOR(HTEMP,TEXT(I+1))
            IF(I1 .LT. 5) GO TO 30
            I2 = I2 + 1
            I1 = 1
 30         CONTINUE
            WIDTH = WIDTH + 1
C
C       PUT IN ¶.  (END OF TEXT)
C
   40       I1 = I1 + 1
            HTEMP = ISHFT(WORD(I2), 8)
            WORD(I2) = IOR(HTEMP,TEXT(13))
            IF(I1 .LT. 5) GO TO 42
            I2 = I2 + 1
            I1 = 1
   42       HTEMP = ISHFT(WORD(I2), 8)
            WORD(I2) = IOR(HTEMP,TEXT(12))
C
C   LEFT JUSTIFY LAST WORD.
C
   44       I1 = I1 + 1
            IF(I1 .GE. 5) GO TO 46
            WORD(I2) = ISHFT(WORD(I2), 8)
            GO TO 44
C
C       PUT IN THE MINUS SIGN
C
```

```
   46 IF (NUMBR .GE. 0.0) GO TO 50
      IF(WWORD(1).NE.BLANK) GO TO 60
      NUMMX = 4 * I2
      DO 47 I = 2, NUMMX
      IP = I
      IF(WWORD(I).NE.BLANK) GO TO 48
   47 CONTINUE
   48 WWORD(IP-1) = HMINUS
      GO TO 50
   60 I2 = WIDTH/4
      IF(I2 .EQ. 0)GO TO 64
      DO 62 K = 1, I2
   62 WORD(K) = HASTR
   64 I1 = WIDTH - 4*I2
      IF(I1 .GT. 0) WORD(I2+1) = SFIL(I1)
C
   50 CALL WDDRAW(XSTART,YSTART,DX,DY,SX,SXY,SY,WORD)
      RETURN
      END
```

```
      SUBROUTINE GRAFIT (NPL,X1,X2,X1R,X2R,IX1,IX2,Y1,Y2,Y1R,Y2R,IY1,
     .    IY2,XTIT,NDVX,YTIT,NDVY)
C
C       NPL     INTEGER - THE NUMBER OF THE PLOT REFERENCED.  UP TO FOUR
C                         PLOTS MAY BE PLACED ON ONE GRAPH
C       X1      REAL    - A FLOATING POINT NUMBER, THE LABEL OF THE X
C                         AXIS MINIMUM.  THE LITERAL STRING 'NONE'
C                         PRODUCES NO LABEL
C       X2      REAL    - A FLOATING POINT NUMBER, THE LABEL OF THE X AXIS
C                         MAXIMUM.  THE LITERAL STRING 'NONE' PRO-
C                         DUCES NO LABEL
C       X1R     REAL    - THE USER SPACE VALUE OF THE X AXIS MINIMUM
C       X2R     REAL    - THE USER SPACE VALUE OF THE X AXIS MAXIMUM
C       IX1     REAL    - THE MINIMUM VALUE OF X IN THE DATA TO BE PLOTTED
C       IX2     REAL    - THE MAXIMUM VALUE OF X IN THE DATA TO BE PLOTTED
C       Y1      REAL    - A FLOATING POINT NUMBER, THE LABEL OF THE Y
C                         AXIS MINIMUM.  THE LITERAL STRING 'NONE'
C                         PRODUCES NO LABEL.
C       Y2      REAL    - A FLOATING POINT NUMBER, THE LABEL OF THE Y
C                         AXIS MAXIMUM.  THE LITERAL STRING 'NONE'
C                         PRODUCES NO LABEL
C       Y1R     REAL    - THE USER SPACE VALUE OF THE Y AXIS MINIMUM
C       Y2R     REAL    - THE USER SPACE VALUE OF THE Y AXIS MAXIMUM
C       IY1     REAL    - THE MINIMUM VALUE OF Y IN THE DATA TO BE PLOTTED
C       IY2     REAL    - THE MAXIMUM VALUE OF Y IN THE DATA TO BE PLOTTED
C       XTIT    CHAR    - A STRING OF LITERAL CHARACTERS TERMINATED BY
C                         A ¶.  THIS IS THE TITLE FOR THE X AXIS AND
C                         SHOULD BE LESS THAN 30 CHARACTERS
C       NDVX    INTEGER - THE NUMBER OF INTERVALS TO BE DRAWN ON THE
C                         X AXIS
C       YTIT    CHAR    - A STRING OF LITERAL CHARACTERS TERMINATED BY
C                         A ¶.  THIS IS THE TITLE FOR THE Y AXIS AND
C                         SHOULD BE LESS THAN 30 CHARACTERS
C       NDVY    INTEGER - THE NUMBER OF INTERVALS TO BE DRAWN ON THE
C                         Y AXIS
C
C       ENTRY POINTS (WITH ARGUMENTS):
C
C       PLOTCH (NPL,X,Y,N,CHARAC) - PLOTS ALPHANUMERIC CHARACTERS AT THE
C                                   COORDINATES PROVIDED
C       X       REAL    - A ONE DIMENSIONAL ARRAY CONTAINING THE
C                         COORDINATES FOR X
C       Y       REAL    - A ONE DIMENSIONAL ARRAY CONTAINING THE
C                         COORDINATES FOR Y
C       N       INTEGER - THE NUMBER OF ENTRIES IN 'X' AND 'Y'.  IF 'N' =
C                         NO COORDINATES ARE PLOTTED.
C       PCHR    CHAR*1    A SINGLE CHARACTER TO BE USED FOR PLOTTING
C
C       PLOTLN (NPL,X,Y,N) - PLOTS STRAIGHT LINES THROUGH THE COORDINATES
C                         PROVIDED.  ARGUMENTS ARE THE SAME AS FOR
C                         PLOTCH.
C
      PARAMETER (NLINES=200,NPLTMX=4)
      REAL IX1, IX2, IY1, IY2, NONE, HOLD(12,NPLTMX), WX(NLINES)
      REAL CC(16), C(16), WY(NLINES)
```

```
      CHARACTER*1 XTIT(NDVX), YTIT(NDVY), PCHR, TITLE(132)
      INTEGER  IX, IY, IT, NTX, NTY, INDX(16)
      DIMENSION X(1), Y(1) , XL(1), YB(1), XR(1), YT(1)
      LOGICAL PLOTON(NPLTMX)/NPLTMX*.FALSE./
      LOGICAL LLABPL(NPLTMX)/NPLTMX*.FALSE./
      DIMENSION NTICX(NPLTMX),NTICY(NPLTMX),XMINO(NPLTMX),XMAXO(NPLTMX)
      DIMENSION YMINO(NPLTMX),YMAXO(NPLTMX)
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG
      DATA NONE/'NONE'/
      DATA CC/.0500,.0600,.0310,.0600,.11,.15,
     .            .0800,.0781,.1780,.2500,.0300,.0315,.0600,.040,
     .            .020,.025/
      DATA INDX/3,3,3,9,9,9,3,3,3,3,9,9,9,3,3,9/
C
C     THIS ROUTINE INITIALIZES A GRAPHICS GRID OR CONTOUR DIAGRAM
C     WITH A FULL SET OF LABELS AND SWITCHES THE REST OF THE CONTOUR
C     GRAPHICS MATERIAL TO THE GRAPHICS MODE.
C
C     DEFINE THE PLOTTING REGIONS.
C
      CHSIZE(4) = .44
      IF(NPL.GT.NPLTMX.OR.NPL.LT.1) STOP 72
      PLOTON(NPL) = .TRUE.
      HOLD(1,NPL) = X1
      HOLD(2,NPL) = X2
      HOLD(3,NPL) = X1R
      HOLD(4,NPL) = X2R
      HOLD(5,NPL) = IX1
      HOLD(6,NPL) = IX2
      HOLD(7,NPL) = Y1
      HOLD(8,NPL) = Y2
      HOLD(9,NPL) = Y1R
      HOLD(10,NPL) = Y2R
      HOLD(11,NPL) = IY1
      HOLD(12,NPL) = IY2
      IF(LLABPL(NPL)) THEN
        HOLD(1,NPL) = XMINO(NPL)
        HOLD(2,NPL) = XMAXO(NPL)
        HOLD(5,NPL) = XMINO(NPL)
        HOLD(6,NPL) = XMAXO(NPL)
        HOLD(7,NPL) = YMINO(NPL)
        HOLD(8,NPL) = YMAXO(NPL)
        HOLD(11,NPL) = YMINO(NPL)
        HOLD(12,NPL) = YMAXO(NPL)
      ENDIF
      CALL BOXPLT (X1R, Y1R, X2R, Y2R)
      DO 11 I = 1, 16
   11 C(I) = CC(I) * (HOLD(INDX(I)+1,NPL)-HOLD(INDX(I),NPL))
C
C     DETERMINE NUMBER OF CHARACTERS IN AXIS TITLES.
C
      TITLE(1) = '¶'
      TITLE(2) = 'M'
      MAXT = 130
```

```
      CALL WDCOUNT(XTIT, NTX)
      IF (NTX.LT.1) THEN
        IF (LUDIAG.GT.0) WRITE (LUDIAG,12)
      ELSE
        DO 15 I = 1, MAXT
   15   TITLE(I+2) = XTIT(I)
        C(8) =   (X2R+X1R)/2.
        XRR = C(8) + NTX*C(1)
        CALL LABEL(TITLE, C(8), Y1R-C(6), XRR, Y1R-C(6)+C(13), 0.0)
      END IF
   12 FORMAT(' NO ESCAPE SEQUENCE IN TITLE - GRAFIT')
      CALL WDCOUNT(YTIT, NTY)
      IF(NTY.LT.1) THEN
        IF (LUDIAG.GT.0) WRITE (LUDIAG,12)
      ELSE
        C(11) = (Y1R+Y2R) / 2.
        DO 16 I = 1, MAXT
   16   TITLE(I+2) = YTIT(I)
        CALL LABEL(TITLE, X1R-C(7), C(11), X1R-C(7)+NTY*C(1),
     .  C(11)+C(13), 1.5707)
      ENDIF
C
C      DECIDE WHICH LABELS TO USE
C
      IF(LLABPL(NPL)) THEN
        CALL FNUMBR (X1R-C(10), Y1R-C(16), C(3), 0.0,
     .    C(2),0.0,C(4), YMINO(NPL), 6, 1)
        CALL FNUMBR (X1R-C(10), Y2R-C(16), C(3), 0.0,
     .    C(2),0.0,C(4), YMAXO(NPL), 6, 1)
        CALL FNUMBR (X1R-C(9), Y1R-C(5), C(3), 0.0,
     .    C(2),0.0,C(4), XMINO(NPL), 6, 1)
        CALL FNUMBR (X2R-C(9), Y1R-C(5), C(3), 0.0,
     .    C(2),0.0,C(4), XMAXO(NPL), 6, 1)
      ELSE
        IF(Y1.NE.NONE)
     .    CALL FNUMBR (X1R-C(10), Y1R-C(16), C(3), 0.0,
     .    C(2),0.0,C(4), Y1, 6, 1)
        IF(Y2.NE.NONE)
     .    CALL FNUMBR (X1R-C(10), Y2R-C(16), C(3), 0.0,
     .    C(2),0.0,C(4), Y2, 6, 1)
        IF(X1.NE.NONE)
     .    CALL FNUMBR (X1R-C(9), Y1R-C(5), C(3), 0.0,
     .    C(2),0.0,C(4), X1, 6, 1)
        IF(X2.NE.NONE)
     .    CALL FNUMBR (X2R-C(9), Y1R-C(5), C(3), 0.0,
     .    C(2),0.0,C(4), X2, 6, 1)
      ENDIF
C
C     PUT TIKS ON THE GRAPH IF THE NUMBER IS GREATER THAN ZERO.
C
      IF(LLABPL(NPL)) THEN
        NDVXLL = NTICX(NPL)
        NDVYLL = NTICY(NPL)
      ELSE
        NDVXLL = NDVX
```

```
        NDVYLL = NDVY
      ENDIF
      IF (NDVXLL.GT.1) THEN
        DELX = (X2R - X1R)/FLOAT(NDVXLL)
        DO 3 I = 1, NDVXLL-1
        XX = X1R + FLOAT(I)*DELX
    3   CALL LINE (XX, Y1R, XX, Y1R+C(16))
        DO 7 I = 1, NDVXLL-1
        XX = X1R + FLOAT(I)*DELX
    7   CALL LINE(XX, Y2R, XX, Y2R-C(16))
      END IF
      IF (NDVYLL.GT.1) THEN
        DELY = (Y2R - Y1R)/FLOAT(NDVYLL)
        DO 4 I = 1, NDVYLL-1
        YY = Y1R + FLOAT(I)*DELY
    4   CALL LINE (X1R, YY, X1R+C(15), YY)
        DO 8 I = 1, NDVYLL-1
        YY = Y1R + FLOAT(I)*DELY
    8   CALL LINE(X2R, YY, X2R-C(15), YY)
      END IF
      RETURN
C
      ENTRY PLOTCH (NPL, X, Y, N, PCHR)
C
      IF (N.LE.0) RETURN
      IF(NPL.GT.NPLTMX.OR.NPL.LT.1) STOP 72
      IF(.NOT.PLOTON(NPL)) RETURN
      DX = (HOLD(4,NPL)-HOLD(3,NPL)) / (HOLD(6,NPL)-HOLD(5,NPL))
      DY=(HOLD(10,NPL)-HOLD(9,NPL))/(HOLD(12,NPL)-HOLD(11,NPL))
      CHSIZE(2) = (HOLD(10,NPL)-HOLD(9,NPL)) * XYCOOD(3) * CHSIZE(5)
      NLM = 1
   13 IST = NLM
      NLM = MIN(NLM-1+NLINES, N)
      DO 5 I = IST, NLM
      WX(I-IST+1) = HOLD(3,NPL) + DX * (X(I)-HOLD(5,NPL))
    5 WY(I-IST+1) = HOLD(9,NPL) + DY * (Y(I)-HOLD(11,NPL))
      CALL CHPLOT (WX, WY, PCHR, 1, 1, NLM-IST+1)
      IF(NLM.LT.N) GO TO 13
      CHSIZE(2) = CHSIZE(1)
      RETURN
C
      ENTRY PLOTLN (NPL, X, Y, N)
C
      IF (N.LE.0) RETURN
      IF(NPL.GT.NPLTMX.OR.NPL.LT.1) STOP 72
      IF(.NOT.PLOTON(NPL)) RETURN
      DX = (HOLD(4,NPL)-HOLD(3,NPL)) / (HOLD(6,NPL)-HOLD(5,NPL))
      DY=(HOLD(10,NPL)-HOLD(9,NPL))/(HOLD(12,NPL)-HOLD(11,NPL))
      NLM = 1
   14 IST = NLM
      NLM = MIN(NLM-1+NLINES,N)
      DO 6 I = IST, NLM
      WX(I-IST+1) = HOLD(3,NPL) + DX * (X(I)-HOLD(5,NPL))
    6 WY(I-IST+1) = HOLD(9,NPL) + DY * (Y(I)-HOLD(11,NPL))
      CALL LNPLOT (WX, WY, 1, 1, NLM-IST+1)
```

```
      IF(NLM.GE.N) RETURN
      GO TO 14
C
      ENTRY PLOTBAR (NPL, XL, YB, XR, YT, N, VALUE)
C
      IF (N.LE.0) RETURN
      IF(NPL.GT.NPLTMX.OR.NPL.LT.1) STOP 72
      IF(.NOT.PLOTON(NPL)) RETURN
      DX = (HOLD(4,NPL)-HOLD(3,NPL)) / (HOLD(6,NPL)-HOLD(5,NPL))
      DY = (HOLD(10,NPL)-HOLD(9,NPL)) / (HOLD(12,NPL)-HOLD(11,NPL))
      NLM = 1
   20 IST = NLM
      NLM = MIN(NLM-1+NLINES,N)
      CALL LINWID(1)
      DO 21 I = IST, NLM
      X1 =      HOLD(3,NPL) + DX * (XR(I)-HOLD(5,NPL))
      X0 =      HOLD(3,NPL) + DX * (XL(I)-HOLD(5,NPL))
      Y1 =      HOLD(9,NPL) + DY * (YT(I)-HOLD(11,NPL))
      Y0 =      HOLD(9,NPL) + DY * (YB(I)-HOLD(11,NPL))
      WX(1) = X0
      WX(2) = X0
      WX(3) = X1
      WX(4) = X1
      WY(1) = Y0
      WY(2) = Y1
      WY(3) = Y1
      WY(4) = Y0
      CALL PLYGON(WX, WY, 4)
      TX = X1 - X0
      TY = HOLD(10,NPL) - HOLD(9,NPL)
      IDIGIT = 0
      IF (ABS(VALUE).LT.10.) IDIGIT = 1
      NDIG = 3
      IF(ABS(VALUE).GE.1000.) NDIG = 4
      YOFF = Y1 - .02 * TY
      IF (VALUE.LT.0.0) YOFF = Y0 - 0.15 * TY
      FAC = 1. / FLOAT(NDIG)
      XST = FAC * .95
      IF(VALUE.NE.0.0) CALL FNUMBR(X0+0.01*TX,YOFF, ; -.025->+.01
     . FAC*TX,0.0,XST*TX,0.0,0.12*TY,ABS(VALUE),NDIG,IDIGIT)
   21 CONTINUE
      CALL LINWID(3)
      IF(NLM.GE.N) RETURN
      GO TO 20
C
      ENTRY GRISET (XL, YB, XR, YT)
C
      CALL DEFINE (XL, YB, XR, YT)
      RETURN
C
      ENTRY GRILAB (NPL, XLL, YBT, XRT, YTT)
      DX = (XRT-XLL) / 3.
      DY = (YTT-YBT) / 3.
      XINT1 = 10.**(INT(ALOG10(DX)))
      YINT1 = 10.**(INT(ALOG10(DY)))
```

```
      XINT2 = DX / XINT1
      YINT2 = DY / YINT1
      IF(XINT2.LT.2.0) THEN
        DX = XINT1
      ELSE IF (XINT2.LT.5.0) THEN
        DX = 2.*XINT1
      ELSE IF (XINT2.LT.10.) THEN
        DX = 5.*XINT1
      ELSE
        DX = 10.*XINT1
      ENDIF
      IF(YINT2.LT.2.0) THEN
        DY = YINT1
      ELSE IF (YINT2.LT.5.0) THEN
        DY = 2.*YINT1
      ELSE IF (YINT2.LT.10.) THEN
        DY = 5.*YINT1
      ELSE
        DY = 10.*YINT1
      ENDIF
      XINT2 = XLL / DX
      YINT2 = YBT / DY
      IF(XINT2.LT.0.0) XINT2 = XINT2 - 0.99999
      IF(YINT2.LT.0.0) YINT2 = YINT2 - 0.99999
      XMINO(NPL) = DX * (INT(ABS(XINT2))*SIGN(1.,XINT2))
      YMINO(NPL) = DY * (INT(ABS(YINT2))*SIGN(1.,YINT2))
      XINT2 = XRT / DX
      YINT2 = YTT / DY
      IF(XINT2.GT.0.0) XINT2 = XINT2 + 0.99999
      IF(YINT2.GT.0.0) YINT2 = YINT2 + 0.99999
      XMAXO(NPL) = DX * (INT(ABS(XINT2))*SIGN(1.,XINT2))
      YMAXO(NPL) = DY * (INT(ABS(YINT2))*SIGN(1.,YINT2))
      NTICX(NPL) = (XMAXO(NPL)-XMINO(NPL)+0.5) / DX
      NTICY(NPL) = (YMAXO(NPL)-YMINO(NPL)+0.5) / DY
      LLABPL(NPL) = .TRUE.
      RETURN
      END
```

```
      SUBROUTINE HHDRAW (XCHAR, YCHAR, SX,SXY,SY, CHNUMB, SET, IERR)
C
C      XCHAR  REAL    - X STARTING COORDINATE OF THE CHARACTER TO BE
C                       DRAWN
C      YCHAR  REAL    - Y STARTING COORDINATE OF THE CHARACTER TO BE
C                       DRAWN
C      SX     REAL    - X MATH SPACE SIZE FOR CHARACTERS
C      SXY    REAL    - SLANT MODIFIER FOR CHARACTERS
C      SY     REAL    - Y MATH SPACE SIZE FOR CHARACTERS
C      CHNUMB INTEGER - INDEXX TO IDENTIFY CHARACTERS WITHIN THE SPECI-
C                       FIED SET
C      SET    INTEGER - NUMBER SPECIFYING A PARTICULAR CHARACTER SET
C
C
        REAL       XCHAR,YCHAR,SX,SXY,SY,X(128),Y(128)
        INTEGER    M, CHNUMB,SET,PEN(128),ERROR,IERR
        CHARACTER*1 CHRIRV
        COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
        COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
C
C
        IERR = 1
        IF (SET.GT.1) GO TO 3
        IF (SET.LT.1) RETURN
C
        IRV = IREVCH(CHNUMB, CHRIRV)
        CALL SYMBOL(XCHAR+CHSIZE(7),YCHAR+CHSIZE(8),CHRIRV)
        RETURN
C
    3   CALL HSETS (CHNUMB,X,Y,PEN,N,SET-1,ERROR)
        IF(ERROR.NE.0) RETURN
        IERR = 0
C
        M = N
        IF (M.EQ.0) RETURN
C
C    ROTATE COORDINATES IF ANGLE>0.0
C
        SXP = ABS(XYCOOD(1))
        SYP = ABS(XYCOOD(3))
        OSXP = 1. / SXP
        OSYP = 1. / SYP
        SXP = SXP * SX
        SYP = SYP * SY
        DO 9 I = 1, M
        X(I) = X(I) * 0.216 * SXP
    9   Y(I) = Y(I) * 0.0800 * SYP + X(I) * SXY
        IF (ANGLE.EQ.0.0) GO TO 4
        SINX = SIN(ANGLE)
        COSX = COS(ANGLE)
        DO 6 I = 1, M
        XP = X(I)*COSX - Y(I)*SINX
        YP = X(I)*SINX + Y(I)*COSX
        X(I) = XP
    6   Y(I) = YP
```

```
    4     DO 1 I = 1, M
          X(I) = X(I) * OSXP
    1     Y(I) = Y(I) * OSYP
C
C     ADD THE ABSOLUTE POSITION
C
          DO 7 I = 1, M
          X(I) = X(I) + XCHAR
    7     Y(I) = Y(I) + YCHAR
C
C     WRITE THE CHARACTER AS A SET OF CONNECTED STROKES
C
          IS = 1
          IL = IS
   10     IF(PEN(IL+1).EQ.0.OR.IL.GE.M) GO TO 2
          IL = IL + 1
          GO TO 10
    2     IF(IL.GT.IS) CALL LINES(X(IS),Y(IS),X(IS+1),Y(IS+1),IL-IS)
          IF(IL.GE.M) RETURN
          IL = IL + 1
          IS = IL
          GO TO 10
          RETURN
      END
```

```
      SUBROUTINE HSETS (CHNUMB,XX,YY,PEN,N,SET,ERROR)
C
C************************************************************************
C    HSETS
C************************************************************************
C
C THIS SUBROUTINE RECEIVES A SET NUMBER AND A CHARACTER NUMBER
C AND SEARCHES THROUGH 'HSETS.DAT' (FILE CONTAINING THE PACKED
C HERSHEY CHARACTER SUBSETS) TO FIND THE CHARACTER'S COORDINATES
C AND CORRESPONDING 'PEN' VALUES.
C
      PARAMETER(MAXSET=23,MAX1=24)
      LOGICAL FOPEN/.FALSE./
      INTEGER IN(125,96),IOS,CHAR,SET,COORD,ERROR
      INTEGER  CHNUMB,PEN(125),MASK1,MASK2,TABLE(MAX1)
      INTEGER FIRST,LAST,NCHAR(MAXSET),TT(64),IOB(64),IOBC
      EQUIVALENCE (TT(1),TABLE(1)),(TT(33),NCHAR(1))
      DATA MASK2/ZFFFF/,MASK1/Z7FFFFFFF/,NSET/-1/
      REAL  XX(125),YY(125)
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
C
C OPEN FILE CONTAINING PACKED HERSHEY CHARACTER SETS IF NOT
C PREVIOUSLY OPENED
C
          IF(SET.EQ.NSET.AND.FOPEN) GO TO 41
          CLOSE(LUHSET)
C
C  OPEN A NEW FILE FOR THE PACKED HERSHEY CHARACTER SUBSETS
C
          OPEN(UNIT=LUHSET,IOSTAT=IOS,ERR=50,FILE='SYS:H1.CAT/S',
     *    ACCESS='DIRECT',FORM='BINARY',RECL=256,SIZE=730,
     *    BLOCKSIZE=256,TYPE='CONTIG')
          REWIND LUHSET
          FOPEN=.TRUE.
          IF(LUDIAG.NE.0) WRITE(LUDIAG,20) LUHSET
20        FORMAT(' HSET CHARACTER FILE OPENED TO UNIT =',I3)
C
C     READ STARTING RECORD POSITIONS OF THE SETS INTO 'TABLE'
C     READ THE NUMBER OF CHARACTERS IN EACH SET INTO 'NCHAR'
C
          READ(LUHSET) TT
C
C CHECK THAT THE SET AND CHARACTER NUMBERS ARE VALID
C
          ERROR=0
          IF(SET.GT.MAXSET) THEN      ;SET NOT IN FILE
              ERROR=1
              GO TO 52
          ENDIF
          IF(CHNUMB.GT.NCHAR(SET)) THEN    ,CHARACTER NOT IN SET
              ERROR=2
              GO TO 52
          ENDIF
C
C READ THE CHARACTERS' COORDINATES INTO THE 'IN' ARRAY
```

```
C
                NSET = SET
                FIRST=TABLE(SET)    ;FIRST RECORD OF SET
                LAST=TABLE(SET+1) - 1     ;LAST RECORD OF SET
                CHAR=1
                COORD=0
                DO 30 I=FIRST,LAST
                READ(LUHSET,REC=I) IOB
C
C   STORE DATA IN ARRAY ELEMENTS FOR 1 CHARACTER AT A TIME
C
                J=0
 32             J=J+1
                COORD=COORD+1
                IN(COORD,CHAR)=IOB(J)
                IF((IOB(J).NE.MASK1).AND.(J.LT.64)) GO TO 32
C
C   END OF CHARACTER FOUND; READ COORDINATES FOR NEXT CHARACTER
C
                IF(IOB(J).EQ.MASK1) THEN
                    CHAR=CHAR+1
                    COORD=0
                IF(CHAR.GT.NCHAR(SET)) GO TO 31
                ENDIF
                IF(J.LT.64) GO TO 32
 30             CONTINUE    ;READ NEXT RECORD
 31             CLOSE(LUHSET)
C
C UNPACK THE DATA FOR THE SPECIFIED CHARACTER & DETERMINE THE REAL
C VALUES OF THE COORDINATES AND THE INTEGER VALUE OF THE PEN.
C
 41       N=0
 40       IF(IN(N+1,CHNUMB).EQ.MASK1) RETURN     ;END OF CHARACTER
          N=N+1
C
C   FIND THE 'PEN' VALUE & ELIMINATE 'PEN' BIT FROM PACKED WORD
C
          PEN(N)=ISHFT(IN(N,CHNUMB),-31)
          IOBC=IAND(IN(N,CHNUMB),MASK1)
C
C   FIND THE X & Y COORDINATES
C
          XX(N)=(ISHFT(IOBC,-16))/100.00
          YY(N)=(IAND(IN(N,CHNUMB),MASK2))/100.0
          GO TO 40      ;UNPACK NEXT COORDINATE AND PEN VALUE
C
C ERROR
C
 50       ERROR = IOS
          IF(LUDIAG.NE.0) WRITE(LUDIAG,51) IOS
 51       FORMAT(' ERROR IN HSET - UNABLE TO OPEN CHARACTER FILE, ERROR =',
         . I4)
 52       CLOSE(LUHSET)
          RETURN
          END
```

```
      INTEGER FUNCTION IDCHAR (CHARX)
C
C*****************************************************************************
C     IDHCAR
C*****************************************************************************
C
      LOGICAL ZEROP/.TRUE./
      INTEGER  IDN(128), CHAR, KA, MASK/127/
      CHARACTER*1 CHARX, CHARXX(4), JA(4), CHRIRV
      CHARACTER*8  ASCII(16)
      CHARACTER*1  ASCIII(128)
      EQUIVALENCE (CHAR,CHARXX)
      EQUIVALENCE (JA,KA), (ASCII,ASCIII)
      DATA ASCII/'01234567','89ABCDEF','GHIJKLMN','OPQRSTUV',
     .    'WXYZabcd','efghijkl','mnopqrst','uvwxyz %',
     .    '&@$#.,:;','!?-+=*/(',')[]§†¶"<','>°'' _.-',
     .    4*'        '/
C
C     THIS ROUTINE INITIALIZES THE IDENTITY ARRAY ID TO THE IDENTITY
C     NUMBER OF THE ACCEPTED ASCII TERMINAL CHARACTERS.
C
      IF(ZEROP) THEN
      DO 2 I = 1, 128
    2 IDN(I) = 127
      DO 1 I = 1, 128
      JA(4) = ASCIII(I)
      KA = IAND(KA, MASK)
    1 IF(IDN(KA+1).EQ.127) IDN(KA+1) = I
      ZEROP = .FALSE.
      ENDIF
C
C        THIS INTEGER FUNCTION RETURNS THE IDENTITY NUMBER OF THE ONE
C     BYTE CHARACTER ENTERED THROUGH THE ARGUMENT LIST AS A CHARACTER*1
C     VARIABLE.
C
      CHARXX(4) = CHARX
      CHAR = IAND(CHAR, MASK)
      IDCHAR = IDN(CHAR+1)
      RETURN
C
C     THIS ENTRY REVERSES THE CHARACTER PROCESS - USED BY HHDRAW WHEN
C     CALLING SYMBOL. CHANGES THE NUMBER BACK INTO A CHARACTER
C
      ENTRY IREVCH (ICHR, CHRIRV)
      IREVCH = 0
      IF(ICHR.GE.1.AND.ICHR.LE.128) CHRIRV = ASCIII(ICHR)
      RETURN
      END
```

```
      SUBROUTINE LABEL (CHARS, XL, YB, XR, YT, ANGLE)
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/XNGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      CHARACTER*1 CHARS(*)
      CALL WDCOUNT(CHARS, NC)
      IF(NC.LE.0) THEN
        IF(LUDIAG.GT.0) WRITE(LUDIAG,3) (CHARS(I),I=1,132)
3       FORMAT(' NO ESCAPE SEQUENCE IN LABEL -',/,1X,132A1)
        RETURN
      ENDIF
      DX = (XR - XL) / MAX(1.,FLOAT(NC))
      DY = (YT - YB)
      XNGLE = ANGLE
      CALL WDDRAW (XL, YB, DX, 0.0, DX, 0.0, DY, CHARS)
      RETURN
      END
```

```
      SUBROUTINE LINE (X1, Y1, X2, Y2)
C
C***********************************************************************
C     LINE
C***********************************************************************
C
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
C
C     THIS ROUTINE PLOTS A LINE FROM (X1,Y1) TO (X2,Y2).
C     THE PRESENT ALGORITHM ALWAYS PLOTS FROM THE CURRENT LOCATION TO
C     THE TARGET POINT THUS INSURING PERFECT LINE CONTINUITY.
C
C
      REAL IX1, IX2, IY1, IY2, IDX, IDY
C
      IX1 = XYCOOD(1)*X1 + XYCOOD(2)
      IX2 = XYCOOD(1)*X2 + XYCOOD(2)
      IY1 = XYCOOD(3)*Y1 + XYCOOD(4)
      IY2 = XYCOOD(3)*Y2 + XYCOOD(4)
      IDX =IX2 - IX1
      IDY = IY2 - IY1
     CALL PUTLNV (IX1,IY1,IDX,IDY,1)
      RETURN
      END
```

```
      SUBROUTINE LINES(X1,Y1,X2,Y2,NL)
C
C     THIS SUBROUTINE PLOTS LINES FROM (X1(J),Y1(J)) TO (X2(J),Y2(J))
C     WHILE CONVERTING TO RASTER NUMBERS THROUGH SCALE.
C
C***********************************************************************
C     LINES
C***********************************************************************
C
      PARAMETER (NLINES=200)
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      REAL X1(NL),X2(NL),Y1(NL),Y2(NL)
      REAL IX1(NLINES), IY1(NLINES), IDX(NLINES), IDY(NLINES)
C
C
      NLM = NL
      I2 = 1
    5 I1 = I2
      I2 = MIN0(I2-1+NLINES,NL)
      NLMIN = I2 - I1 + 1
      DO 10 I = I1, I2
      IX1(I-I1+1)=XYCOOD(1)*X1(I)+XYCOOD(2)
      IDX(I-I1+1)=XYCOOD(1)*X2(I)+XYCOOD(2)-IX1(I-I1+1)
      IY1(I-I1+1)=XYCOOD(3)*Y1(I)+XYCOOD(4)
   10 IDY(I-I1+1)=XYCOOD(3)*Y2(I)+XYCOOD(4)-IY1(I-I1+1)
C
      CALL PUTLNV(IX1,IY1,IDX,IDY,NLMIN)
      IF (I2.GE.NL) RETURN
      GO TO 5
      RETURN
      END
```

```
      SUBROUTINE LNPLOT (XARRAY, YARRAY, I1, I2, I3)
C
      PARAMETER (NLINES=200)
      REAL       XARRAY(1), YARRAY(1)
      REAL XA(NLINES),YA(NLINES),XB(NLINES),YB(NLINES)
      INTEGER    I1, I2, I3, IST, I, NOPTS, NOPT
C
      NOPTS=(I3-I1+I2)/I2-1
      IST=I1
    2 NOPT=NOPTS
      NOPT = MIN0 (NLINES,NOPT)
      DO 1 I=1,NOPT
      XA(I)=XARRAY(IST+I2*I-I2)
      YA(I)=YARRAY(IST+I2*I-I2)
      XB(I)=XARRAY(IST+I2*I)
      YB(I)=YARRAY(IST+I2*I)
    1 CONTINUE
      CALL LINES(XA,YA,XB,YB,NOPT)
      NOPTS=NOPTS-NOPT
      IST=IST+I2*NOPT
      IF(NOPTS.GT.0) GO TO 2
      RETURN
  END
```

```
      SUBROUTINE MAPIN (MAPFIL,
     . WIN, XYZ, NV, NE, NES, NEX, NP, NPS, NPX)
C
      PARAMETER (NVM=4000,NEM=400,NPM=400)
      CHARACTER*16 MAPFIL
      REAL XYZ(3,NV), WIN(4)
      INTEGER  NP(8,NPM), NPS(4,NPM),  NE(2,NEM), NES(4,NEM)
      LOGICAL FIRSTC/.TRUE./
C
C     INITIALIZE THE COUNTERS THE FIRST TIME THRU THIS ROUTINE
C
      IF(FIRSTC) THEN
        NV = 0
        NEX = 0
        NPX = 0
        FIRSTC = .FALSE.
      ENDIF
C
C     OPEN AND INPUT A STRUCTURES FILE
C
      NV1 = NVM
      NE1 = NEM
      NP1 = NPM
      CALL PLYPLT (MAPFIL,0)
      CALL PLYGNS(WIN, XYZ(1,NV+1), NV1, NE(1,NEX+1), NES(1,NEX+1),
     . NE1, NP(1,NPX+1), NPS(1,NPX+1), NP1)
      IF(NV1.LE.0) RETURN
C
C     UPDATE THE EDGE AND POLYGON POINTERS
C
      DO 132 I = NEX+1, NEX+1+NE1
      DO 132 J = 1, 2
  132 NE(J,I) = NE(J,I) + NV
      DO 133 I = NPX+1, NPX+1+NP1
      DO 133 J = 1, 8
      IF(NP(J,I).GT.0) NP(J,I) = NP(J,I) + NV
  133 CONTINUE
      NV = NV + NV1
      NEX = NEX + NE1
      NPX = NPX + NP1
      RETURN
      END
```

```
      SUBROUTINE MAPOUT(WIN, XYZ, NV, NE, NES, NEX, NP, NPS, NPX)
C
$INCLUDE BUILD.COM (NLIST)
      REAL XYZ(3,NV), WIN(4)
      INTEGER  NP(NPVERT,NPX), NPS(NSPEC,NPX), NE(2,NEX), NES(NSPEC,NEX)
C
      CALL DEFINE(WIN(1),WIN(2),WIN(3),WIN(4))
      CALL PLYPL4(NE,NEX,XYZ,WIN,NES)
      CALL PLYPL5(NP,NPX,XYZ,WIN,NPS,NPVERT)
      CALL FRAME
      RETURN
      END
```

```
      SUBROUTINE PLYGON(X, Y, N)
C
C     DRAW A COMPLETE POLYGON AND FILL IT WITH 'LFLMAT'
C
      PARAMETER (NPMAX=32)
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFMOD/LFLMAT(5),LINWDM(5),LCLMAT(5)
      DIMENSION X(N), Y(N), XL(2*NPMAX)
C
      IF(IDEVIC.EQ.0) RETURN
      NL = MIN0(NPMAX, N)
      GO TO (1,1,3,1,1), IDEVIC
    1 DO 11 I = 1, N-1
   11 CALL LINE(X(I), Y(I), X(I+1), Y(I+1))
      CALL LINE(X(N), Y(N), X(1), Y(1))
      RETURN
    3 DO 31 I = 1, NL
      XL(2*I-1) = MAX(X(I) * XYCOOD(1) + XYCOOD(2), 0.0)
   31 XL(2*I) = MAX(Y(I) * XYCOOD(3) + XYCOOD(4), 0.0)
      CALL GMOVA(XL(1), XL(2))
      CALL GPOLA (XL(3), NL-1)
      RETURN
      END
```

```
      SUBROUTINE PLYPL1(CHOOSE,ENDSTR,SSTART,SFIRST,SLAST,SVALID,
     1IBLANK)
C
C        THIS ROUTINE FINDS POSITIONS OF SUBSTRINGS WITHIN A CHARACTER
C     STRING.  COMMAS AND SPACES ARE SUBSTRING DELIMITERS.  WHEN
C     A VALID SUBSTRING IS FOUND, ITS FIRST AND LAST CHARACTER
C     POSITIONS ARE RETURNED TO THE CALLING ROUTINE.
C        THIS ROUTINE ALSO INCLUDES THE OPTION OF CONSIDERING A GROUP
C     OF BLANKS SET OF BY COMMAS TO BE A VALID SUBSTRING.  IF SUCH
C     A SUBSTRING IS FOUND, THE POSITIONS OF THE FIRST AND LAST
C     BLANKS ARE NOT RETURNED, BUT RATHER THE POSITIONS OF THE
C     DELIMITERS (COMMAS).
C
C     VARIABLES USED:
C      CFIND  = POSITION OF 1ST COMMA FOUND (IF 'BLANK' OPTION SET)
C      CHOOSE = INTEGER CONTAINING CHARACTER STRING
C      ENDSTR = NUMBER OF CHARACTERS IN THE ENTIRE STRING
C      IBLANK = 'BLANK' FLAG
C         (input) = 0 - BLANKS NOT VALID SUBSTRING
C         (input) = 1 - BLANKS FOLLOWED BY 1 COMMA - VALID SUBSTRING
C         (input) = 2 - BLANKS FOLLOWED BY 2 COMMAS - VALID SUBSTRING
C         (output)= -1,-2 - SPECIFIED BLANKS (1 OR 2) FOUND
C      SFIRST  = POSITION OF FIRST CHARACTER OF SUBSTRING
C      SLAST   = POSITION OF LAST CHARACTER OF SUBSTRING
C      SSTART  = STARTING POSITION FOR SUBSTRING SEARCH
C      STRING  = CHARACTER ARRAY CONTAINING THE CHARACTER STRING
C      SVALID  = 'VALID SUBSTRING' FLAG
C                = TRUE - VALID SUBSTRING FOUND
C                = FALSE - NO VALID SUBSTRING FOUND
C
      LOGICAL SVALID
      INTEGER SFIRST,SLAST,SSTART,ENDSTR
      INTEGER IBLANK,CFIND,CHOOSE(20),STR(20)
      CHARACTER*1 STRING(80),SPACE/' '/,CR/ZD/,COMMA/Z2C/
      EQUIVALENCE (STR,STRING)
C
C     ASSUME VALID SUBSTRING
C
      SVALID=.TRUE.
C
C     FORM AN EQUIVALENT CHARACTER STRING
C
      DO 10 I=1,20
10    STR(I)=CHOOSE(I)
C
C     INVALID STARTING POSITION - PAST END OF STRING
C
      IF(ENDSTR.LT.SSTART) GO TO 40
C
C     FIND POSITION OF FIRST ELEMENT OF SUBSTRING
C
      CFIND=0
      DO 20 I=SSTART,ENDSTR
      SFIRST=I
C
```

```
C     A COMMA FOUND
C
      IF (IBLANK.NE.0.AND.STRING(I).EQ.COMMA) THEN
C
C        FOUND FIRST COMMA
C
         IF (CFIND.EQ.0) THEN
            CFIND=I
            IF (IBLANK.EQ.2) GO TO 20
         ENDIF
C
C        FOUND BLANK SUBSTRING; NOTE POSITIONS OF DELIMITERS (COMMAS)
C
         SLAST=SFIRST
         SFIRST=CFIND
         IBLANK=-IBLANK
         GO TO 100
      ENDIF
C
C     FOUND FIRST CHARACTER OF SUBSTRING - NOW FIND LAST
C
      IF((STRING(I).NE.SPACE).AND.(STRING(I).NE.CR).AND.
     1 (STRING(I).NE.COMMA)) GO TO 60
 20   CONTINUE
C
C     NO SUBSTRING FOUND - ONLY DELIMITER
C
      WRITE(6,998)
 998  FORMAT(' REACHED THE END WITHOUT FINDING A NON-BLANK CHARACTER')
      GO TO 40
C
C     FIND POSITION OF LAST CHARACTER OF SUBSTRING
C
 60   IF (SFIRST.EQ.ENDSTR) GO TO 45
      DO 50 J=SFIRST+1,ENDSTR
      SLAST=J-1
C
C     FOUND SUBSTRING DELIMITER - CAN RETURN NOW
C
      IF((STRING(J).EQ.SPACE).OR.(STRING(J).EQ.COMMA)) GO TO 100
 50   CONTINUE
C
C     NO SUBSTRING DELIMITER => LAST CHARACTER OF SUBSTRING IS THE
C     LAST CHARACTER OF THE STRING
C
 45   SLAST=ENDSTR
      GO TO 100
C
C NO SUBSTRING FOUND
C
 40   SVALID=.FALSE.
 100  RETURN
      END
```

```
      SUBROUTINE PLYPL2(WINDOW,OPENN,WORLD)
C
C  READ WINDOW COORDINATES AND STORE THEM IN 'WINDOW' ARRAY.  ALSO
C  CHECK VALIDITY OF COORDINATES.  IF VALID, 'OPENN' IS TRUE;
C  OTHERWISE, 'OPENN' IS FALSE. WINDOW COORDINATES WILL BE STORED
C  AS FOLLOWS:
C                          INDEX / WINDOW(INDEX)
C                          -----   -------------
C                            1          left
C                            2          bottom
C                            3          right
C                            4          top
C
      LOGICAL OPENN
      REAL WINDOW(4),WORLD(6)
C
C     INITIALIZE
C
      OPENN=.FALSE.   ;ASSUME INVALID COORDINATES
C
C  CHECK VALIDITY OF WINDOW COORDINATES
C
      DO 30 I=1,2
      IF(WINDOW(I).GE.WINDOW(I+2)) RETURN   ;INVALID COORDINATES
 30   CONTINUE
C
C  WINDOW COORDINATES VALID
C
      OPENN=.TRUE.
      RETURN
      END
```

```
      SUBROUTINE PLYPL3(ARRAY,NARRAY,SP,NPNT,NVERT)
C
C THIS SUBROUTINE PLOTS VERTICES FOUND WITHIN THE WINDOW AND THEIR
C CORRESPONDING LINE NUMBERS.
C
      INTEGER CHARAC/'.   '/,NARRAY,CHR,NPNT(NVERT),DIGITS
      REAL ARRAY(3,NARRAY),SP(4),NUM,NUMBR,SSP(4)
      REAL NX,NY,XINC,XSTART
      COMMON/PLSCMR/DOZ,WCX,WCY,FANGLE,TANAL
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
C
C DEFINE THE WINDOW SPACE
C
      DO 1 I = 1, 4
1     SSP(I) = XYCOOD(I)
      CALL DEFINE(SP(1),SP(2),SP(3),SP(4))
C
C PLOT 1 VERTEX AND NUMBER AT A TIME
C
      DO 20 I=1,NARRAY
C
C DO NOT PLOT A 'DELETED' VERTEX
C
      IF(NPNT(I).EQ.0) GO TO 20
C
C ADD PERSPECTIVE
C
      D = DOZ/((DOZ+ARRAY(3,I))*TANAL)
      XX = (ARRAY(1,I)-WCX)*D + WCX
      YY = (ARRAY(2,I)-WCY)*D + WCY
C
C DO NOT PLOT A VERTEX WHICH IS OUTSIDE OF WINDOW
C
      IF((XX.LT.SP(1)).OR.(XX.GT.SP(3))) GO TO 20
      IF((YY.LT.SP(2)).OR.(YY.GT.SP(4))) GO TO 20
C
C PLOT THE VERTEX USING THE CHARACTER MODE OF THE HARDWARE
C
      CALL SYMBOL(XX,YY,CHARAC)
C
C SCALE MULTIPLIERS FOR NUMBERS
C
      XINC=0.025*(SP(3)-SP(1))
      NX=0.035*(SP(3)-SP(1))
      NY=0.035*(SP(4)-SP(2))
      NUMBR=I
C
C DETERMINE NUMBER OF DIGITS 'NUM' (THE VERTEX LINE NUMBER)
C
      NUM=I
      DIGITS=0
10    NUM=NUM/10.0
      DIGITS=DIGITS+1
      IF (NUM.GE.1) GO TO 10
C
```

```
C  CALCULATE X COORDINATE OF FIRST DIGIT OF THE NUMBER
C
      IF (DIGITS.EQ.1) THEN
          XSTART=XX
      ELSE
          IF (MOD(DIGITS,2).EQ.0) THEN
              XSTART=XX-(DIGITS*XINC)/4
          ELSE
              XSTART=XX-(DIGITS*XINC)/2
          ENDIF
      ENDIF
C
C  PLOT THE NUMBER
C
      CALL FNUMBR(XSTART,YY-NY,XINC,0.0,NX,0.0,
     * NY,NUMBR,DIGITS,0)      ;PRINT VERTEX NUMBER
20    CONTINUE
C
C     RESTORE THE COORDINATE SYSTEM

C
      DO 2 I = 1, 4
2     XYCOOD(I) = SSP(I)
      RETURN
      END
```

```
      SUBROUTINE PLYPL4(ARRAY,NARRAY,VERTX,SP,SPEC)
C
C  THIS SUBROUTINE PLOTS EDGES WHICH ARE WITHIN THE WINDOW OR
C  EDGE SEGMENTS WHICH CROSS A PORTION OF THE WINDOW.
C
      LOGICAL VALID
      INTEGER ARRAY(2,NARRAY), SPEC(4,NARRAY)
      REAL VERTX(3,NARRAY),SP(4),SSP(4),U1,V1,U2,V2
      COMMON/PLSCMR/DOZ,WCX,WCY,FANGLE,TANAL
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
C
C  DEFINE THE WINDOW SPACE
C
      DO 1 I = 1, 4
1     SSP(I) = XYCOOD(I)
      CALL DEFINE(SP(1),SP(2),SP(3),SP(4))
C
C  PLOT ONE EDGE AT A TIME
C
      LWD = -1
      LCL = -1
      DO 150 J=1,NARRAY
C
C  ADD PERSPECTIVE
C
      D = DOZ/((DOZ+VERTX(3,ARRAY(1,J)))*TANAL)
      U1=(VERTX(1,ARRAY(1,J))-WCX)*D + WCX
      V1=(VERTX(2,ARRAY(1,J))-WCY)*D + WCY
      D = DOZ/((DOZ+VERTX(3,ARRAY(2,J)))*TANAL)
      U2=(VERTX(1,ARRAY(2,J))-WCX)*D + WCX
      V2=(VERTX(2,ARRAY(2,J))-WCY)*D + WCY
C
C  CLIP LEFT EDGE
C
      IF (U1.GE.SP(1).AND.U2.GE.SP(1)) GO TO 50
      IF (U1.LT.SP(1).AND.U2.LT.SP(1)) GO TO 150
      IF (U1.GT.SP(1)) GO TO 40
      V1 = (V1-V2)*U1/(U2-U1)+V1
      U1 = SP(1)
      GO TO 50
40    V2 = (V2-V1)*U2/(U1-U2)+V2
      U2 = SP(1)
C
C  CLIP RIGHT EDGE
C
50    IF (U1.LE.SP(3).AND.U2.LE.SP(3)) GO TO 70
      IF (U1.GT.SP(3).AND.U2.GT.SP(3)) GO TO 150
      IF (U1.GT.SP(3)) GO TO 60
      V2 = (V2-V1)*(SP(3)-U1)/(U2-U1)+V1
      U2 = SP(3)
      GO TO 70
60    V1 = (V1-V2)*(SP(3)-U2)/(U1-U2)+V2
      U1 = SP(3)
C
C  CLIP BOTTOM EDGE
```

```
C
70     IF (V1.GE.SP(2).AND.V2.GE.SP(2)) GO TO 90
       IF (V1.LT.SP(2).AND.V2.LT.SP(2)) GO TO 150
       IF (V1.GT.SP(2)) GO TO 80
       U1 = (U1-U2)*V1/(V2-V1)+U1
       V1 = SP(2)
       GO TO 90
80     U2 = (U2-U1)*V2/(V1-V2)+U2
       V2 = SP(2)
C
C  CLIP TOP EDGE
C
90     IF (V1.LE.SP(4).AND.V2.LE.SP(4)) GO TO 110
       IF (V1.GT.SP(4).AND.V2.GT.SP(4)) GO TO 150
       IF (V1.GT.SP(4)) GO TO 100
       U2 = (U2-U1)*(SP(4)-V1)/(V2-V1)+U1
       V2  = SP(4)
       GO TO 110
100    U1 = (U1-U2)*(SP(4)-V2)/(V1-V2)+U2
       V1 = SP(4)
C
C  PLOT THE EDGE
C
110    IF(LWD.NE.SPEC(1,J)) THEN
         LWD = SPEC(1,J)
         CALL LINWID(LWD)
       ENDIF
       IF(LCL.NE.SPEC(2,J)) THEN
         LCL = SPEC(2,J)
         CALL COLOR(LCL)
       ENDIF
       CALL LINE(U1, V1, U2, V2)
150     CONTINUE
C
C      RESTORE THE ORIGINAL COORDINATE SYSTEM
C
       DO 2 I = 1, 4
2      XYCOOD(I) = SSP(I)
       RETURN
       END
```

```
      SUBROUTINE PLYPL5(ARRAY,NARRAY,VERTX,SP,SPEC,NPVERT)
C
C  THIS SUBROUTINE GRAPHS POLYGONS WHICH ARE WITHIN THE WINDOW OR
C  PARTS OF POLYGONS WHICH CROSS A PORTION OF THE WINDOW
C
      PARAMETER(MAXLN=10)
      LOGICAL PASS1
      INTEGER ARRAY(NPVERT,NARRAY),NARRAY,FIRST,SECOND,NUM,S1
      INTEGER INVALD,VINDX,ENDPT,SPEC(4,NARRAY)
      REAL VERTX(3,NARRAY),SP(4),D,SSP(4)
      REAL X1,X2,Y1,Y2,X(MAXLN),Y(MAXLN)
      COMMON/PLSCMR/DOZ,WCX,WCY,FANGLE,TANAL
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
C
C  DEFINE WINDOW SPACE
C
      DO 1 I = 1, 4
1     SSP(I) = XYCOOD(I)
      CALL DEFINE(SP(1),SP(2),SP(3),SP(4))
C
C  PLOT POLYGONS ONE AT A TIME
C
      LWD = -1
      NDX = -1
      LFL = -1
      DO 10 I = 1, NARRAY
      NUM=0
C
C     DETERMINE THE NUMBER OF VERTICES (NON-ZERO ENTRIES) IN THE
C     POLYGON
C
      DO 15 J=1,NPVERT
      IF(ARRAY(J,I).NE.0) NUM=NUM+1 ;NUMBER OF VERTICES IN POLYGON
15    CONTINUE
C
C     FIND THE PERSPECTIVE
C
      DO 60 IV=1,NUM
      D = DOZ/((DOZ+VERTX(3,ARRAY(IV,I)))*TANAL)
      X(IV) = (VERTX(1,ARRAY(IV,I)) - WCX)*D  +  WCX
      Y(IV) = (VERTX(2,ARRAY(IV,I)) - WCY)*D  +  WCY
60    CONTINUE
C
C  PLOT THE POLYGON
C
      IF(SPEC(1,I).NE.LWD) THEN
        LWD = SPEC(1,I)
        CALL LINWID(LWD)
      ENDIF
      IF(SPEC(2,I).NE.NDX) THEN
        NDX = SPEC(2,I)
        CALL COLOR(NDX)
      ENDIF
      IF(SPEC(3,I).NE.LFL) THEN
        LFL = SPEC(3,I)
```

```
            CALL LINES(X(1), Y(1), X(2), Y(2), NUM-1)
          ENDIF
   10     CONTINUE                          ;GET NEXT POLYGON
C
C         RESTORE THE COORDINATE SYSTEM
C
          DO 2 I = 1, 4
    2     XYCOOD(I) = SSP(I)
          RETURN
          END
```

```
      SUBROUTINE PLYPLT(FILE,OPTION)
$INCLUDE BUILD.COM
      PARAMETER      (NPART=20)
      LOGICAL        EXIST, VALID
      INTEGER        NVERTO, NELMTO, NEDGEO, NPOLYO, FIRST, LAST, IOS
      INTEGER        EDGS(NSPEC,NEDGE), POLS(NSPEC,NPOLY), SELECT
      INTEGER        FNM(5), OPTION, EDG(2,NEDGE), POL(NPVERT,NPOLY)
      INTEGER        NPL(2,NPART), ISPEC(NSPEC), JP(NPVERT)
      REAL           WIN(4), VER(3,NVERT), SPACE(4)
      CHARACTER*5    SOURCE
      CHARACTER*17 FNAME, FILENM, OLDFIL, FILE
      CHARACTER*20 NOREC, BLANK
      DATA           VALID/.FALSE./, BLANK/' '/, LUNIT/9/, OLDFIL/' '/
      DATA           IUNIT/5/, ZERO/0/
      EQUIVALENCE  (FILENM,FNM)
      COMMON/PLSCMR/DOZ,WCX,WCY,FANGLE,TANAL
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
C*********************************************************************
C     DATA FILE
C*********************************************************************
C
C  CREATE EQUIVALENT CHARACTER STRING OF FILENAME
C
      FILENM = FILE
      IF(FILENM.EQ.OLDFIL) GO TO 1100
C
C  READ FILENAME
C
      CLOSE (LUNIT)
      CALL PLYPL1(FNM,17,1,FIRST,LAST,EXIST,0)
      IF(.NOT.EXIST) GO TO 90
      FNAME = BLANK
      DO 10 I = FIRST, LAST
      J = I - FIRST + 1
   10 FNAME(J:J) = FILENM(I:I)
C
C  FILE EXIST?
C
      INQUIRE(FILE=FNAME,IOSTAT=IOS,ERR=91,EXIST=EXIST)
      IF(.NOT.EXIST) GO TO 93
C
C  OPEN FILE
C
      OPEN(UNIT=LUNIT,IOSTAT=IOS,ERR=94,FILE=FNAME,STATUS='OLD')
      REWIND LUNIT
C
C  GET(1)    READ DATA INTO VARIABLES AND ARRAYS
C
      READ(LUNIT,FMT=1500,ERR=98) SOURCE,(WINDOW(I),I=1,4)
      IF(SOURCE.NE.'BUILD'.AND.SOURCE.NE.'MOVIE'.AND.SOURCE.NE.'FILNG')
     .  GO TO 98
      READ(LUNIT,FMT=1510,ERR=98) (WORLD(I),I=1,6)
      READ(LUNIT,FMT=1520,ERR=98) NPARTO,NVERTO,NELMTO,NEDGEO
      IF (NPARTO.LT.0) GO TO 98
      NPOLYO=NELMTO-NEDGEO
```

```
         READ(LUNIT,FMT=1520,ERR=98) ((NPL(I,J),I=1,2),J=1,NPARTO)
         IF(NVERTO.LE.0) GO TO 1090
         DO 1020 J=1,NVERTO
 1020    READ(LUNIT,FMT=1530,ERR=98) NPOINT(J),(VERTEX(I,J),I=1,3)
         IF (NELMTO.LE.0) GO TO 1090
C
C        INITIALIZE EDGE AND POLYGON COUNTERS
C
         NE=0
         NP=0
         DO 1080 J=1,NELMTO
            READ(LUNIT,FMT=1525,ERR=98) (JP(I),I=1,NPVERT),(ISPEC(I),I=1,
     1      NSPEC)
C
C           COUNT NUMBER OF VERTICES IN ELEMENT
C
            NCON=0
            DO 1030 I=1,NPVERT
            IF (JP(I).EQ.0) GO TO 1035
 1030       NCON=NCON+1
C
C           ELEMENT IS EDGE
C
 1035       IF (NCON.EQ.2) THEN
               NE=NE+1
               DO 1040 I=1,2
 1040          EDGE(I,NE)=JP(I)
               DO 1050 I=1,NSPEC
 1050          ESPEC(I,NE)=ISPEC(I)
C
C           ELEMENT IS POLYGON
C
            ELSE
               NP=NP+1
               DO 1060 I=1,NCON
 1060          POLY(I,NP)=JP(I)
               IF (NCON.LT.NPVERT) THEN
                  DO 1065 I=NCON+1,NPVERT
 1065             POLY(I,NP)=0
               ENDIF
               DO 1070 I=1,NSPEC
 1070          PSPEC(I,NP)=ISPEC(I)
            ENDIF
 1080    CONTINUE
 1090    CLOSE(LUNIT)
C***********************************************************************
C  DISPLAY
C***********************************************************************
C
C **SPECIFY WINDOW**
C
 1100    IF (NVERTO.NE.0) THEN
            CALL PLYPL2(WINDOW,VALID,WORLD)
         ELSE
             NOREC = 'NO RECORDS'
```

```
             IF(LUDIAG.GT.0) WRITE(LUDIAG,108) NOREC
             RETURN
          ENDIF
C
C   IF VALID WINDOW , SET 'SPACE' COORDINATES TO 'WINDOW' ,OTHERWISE
C   SET THE 'SPACE' COORDINATES TO 'WORLD'
C
          IF(VALID) THEN            ;VALID WINDOW
             DO 40 I=1,4
 40          SPACE(I)=WINDOW(I)
          ELSE
             IF(LUDIAG.GT.0) WRITE(LUDIAG,107)
             DO 41 I=1,2
 41          SPACE(I)=WORLD(I)
             DO 42 I=3,4
 42          SPACE(I)=WORLD(I+1)
          ENDIF
C
C   **FIND CENTER OF WINDOW**
C
          WCX=(SPACE(3)+SPACE(1))/2
          WCY=(SPACE(4)+SPACE(2))/2
C
C   **INITIALIZE FIELD OF VIEW**
C
          DOZ=10000.
          FANGLE=90.
          TANAL=1.0
C
C         INITIALIZE AND SET UP FOR A SINGLE GRAPH - 'NOREC', IF SET, WILL
C         INDICATE THAT THERE ARE NO RECORDS TO BE DISPLAYED
C
          NOREC=BLANK
C
          IF (OPTION.EQ.0) THEN       ;NO PLOTS - READ IN DATA FILE ONLY
             RETURN
          ELSE IF (OPTION.EQ.1) THEN
             IF (NEDGEO.EQ.0) NOREC='NO EDGES'
          ELSE IF(OPTION.EQ.2) THEN
             IF (NVERTO.EQ.0) NOREC='NO VERTICES'
          ELSE IF (OPTION.EQ.3) THEN
             IF (NPOLYO.EQ.0) NOREC='NO POLYGONS'
          ELSE IF (OPTION.EQ.4) THEN
             IF (NEDGEO.EQ.0.AND.NPOLYO.EQ.0) NOREC='NO EDGES OR POLYGONS'
          ELSE
             NOREC='INVALID OPTION'
             IF(LUDIAG.GT.0) WRITE(LUDIAG,108) NOREC
             RETURN
          ENDIF
C
C   **PLOT**
C
          IF (NOREC.EQ.BLANK) THEN
             IF (OPTION.EQ.1) THEN
                CALL PLYPL4(EDGE,NEDGEO,VERTEX,SPACE,ESPEC)
```

```
            ELSE IF (OPTION.EQ.2) THEN
               CALL PLYPL3(VERTEX,NVERTO,SPACE,NPOINT,NVERT)
            ELSE IF (OPTION.EQ.3) THEN
               CALL PLYPL5(POLY,NPOLYO,VERTEX,SPACE,PSPEC,NPVERT)
            ELSE IF (OPTION.EQ.4) THEN
                IF(NEDGEO.GT.0) CALL PLYPL4(EDGE,NEDGEO,VERTEX,SPACE,ESPEC)
                IF(NPOLYO.GT.0) CALL PLYPL5(POLY,NPOLYO,VERTEX,SPACE,PSPEC,
     .          NPVERT)
            ENDIF
         ELSE
            IF(LUDIAG.GT.0) WRITE(LUDIAG,108) NOREC
         ENDIF
         RETURN
C*******************************************************************
C   ENTRY PLYGNS
C*******************************************************************
C
C        A ROUTINE TO RETURN THE COORDINATE ARRAY VALUES FROM A DATA FILE
C
         ENTRY PLYGNS(WIN,VER,NVM,EDG,EDGS,NEM,POL,POLS,NPM)
         NVM = MIN(NVERT,NVM,NVERTO)
         DO 80 I = 1, 4
 80      WIN(I) = WINDOW(I)
         DO 81 I = 1, NVM
         DO 82 J = 1, 3
 82      VER(J,I) = VERTEX(J,I)
 81      CONTINUE
         NEM = MIN(NEDGE,NEM,NE)
         DO 88 I = 1, NEM
         DO 83 J = 1, 2
 83      EDG(J,I) = EDGE(J,I)
         DO 84 J = 1, NSPEC
 84      EDGS(J,I) = ESPEC(J,I)
 88      CONTINUE
         NPM = MIN(NPOLY,NPM,NP)
         DO 85 I = 1, NPM
         DO 86 J = 1, NPVERT
 86      POL(J,I) = POLY(J,I)
         DO 87 J = 1, NSPEC
 87      POLS(J,I) = PSPEC(J,I)
 85      CONTINUE
         RETURN
C
C   ERRORS
C
 90      IF(LUDIAG.GT.0) WRITE(LUDIAG,100) FILE
         RETURN
 91      CLOSE(LUNIT)
         IF (IOS.EQ.349) THEN
            IF(LUDIAG.GT.0) WRITE(LUDIAG,100) FNAME
         ELSE IF (IOS.EQ.324) THEN
            IF(LUDIAG.GT.0) WRITE(LUDIAG,101)
         ELSE
            IF(LUDIAG.GT.0) WRITE(LUDIAG,102)  IOS
         ENDIF
```

```
      RETURN
93    IF(LUDIAG.GT.0) WRITE(LUDIAG,103)
      RETURN
94    CLOSE(UNIT=LUNIT,IOSTAT=IOS,ERR=95)
95    IF(LUDIAG.GT.0)WRITE(LUDAIG,104) IOS
      RETURN
98    IF(LUDIAG.GT.0) WRITE(LUDIAG,110)
      RETURN
C
C  FORMATS
C
100   FORMAT(' INVALID FILE DESCRIPTOR',2X,A17)
101   FORMAT(' NO RECORDS IN FILE')
102   FORMAT(' INQUIRE ERROR = ',I4)
103   FORMAT(' FILE DOES NOT EXIST')
104   FORMAT(' FILE ERROR',I4)
107   FORMAT(' WINDOW DEFAULTS TO "WORLD"')
108   FORMAT(1X,A20,/,/)
110   FORMAT(' ERROR - DATA  FORMAT PROBLEMS -- ENTER NEW COMMAND')
1500  FORMAT(A5,4E12.5)
1510  FORMAT(6E12.5)
1520  FORMAT(16I5)
1525  FORMAT(8I5,5X,8I5)
1530  FORMAT(I5,3E12.5)
      END
```

```
      SUBROUTINE PROPOL (NTRIA, XT1,YT1,FT1,XT2,YT2,FT2,XT3,YT3,FT3,
     1        FL, NX, NY)
C
      PARAMETER (NPT=101)
      REAL     XT1(NTRIA), XT2(NTRIA), XT3(NTRIA)
      REAL     YT1(NTRIA), YT2(NTRIA), YT3(NTRIA)
      REAL     FT1(NTRIA), FT2(NTRIA), FT3(NTRIA)
      INTEGER  MASK(NPT)
      REAL     DIFF1(NPT), DIFF2(NPT), DIFF3(NPT)
      REAL     SGN1(NPT),  SGN2(NPT),  SGN3(NPT)
      REAL     DFL1(NPT),  DFL2(NPT),  DFL3(NPT)
      REAL     ALF1(NPT), ALF2(NPT), ALF3(NPT), X1(NPT), X2(NPT)
      REAL     X3(NPT), Y1(NPT), Y2(NPT), Y3(NPT), TEST(NPT)
      REAL     DELTA, MASK1, MASKS, RMASK(NPT)
      EQUIVALENCE (MASK(1), RMASK(1))
      DATA     MASK1, DELTA, MASKS / 1.0, 1.0E-20, Z80000000/
      DATA XMN1,YMN1,XMN2,YMN2,XMX1,YMX1,XMX2,YMX2/4*0.,4*100./
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     PROPOL (NTRIA, XT1, YT1, FT1, XT2, YT2, FT2, XT3, YT3, FT3,
C                    FL, NX, NY)
C
C     NTRIA  INTEGER             NUMBER OF TRIANGLES IN LIST          I
C     XT1    REAL ARRAY(NTRIA)   X POSITIONS OF FIRST VERTICES        I
C     YT1    REAL ARRAY(NTRIA)   Y POSITIONS OF FIRST VERTICES        I
C     FT1    REAL ARRAY(NTRIA)   FUNCTION VALUES AT FIRST VERTICES    I
C     XT2    REAL ARRAY(NTRIA)   X POSITIONS OF SECOND VERTICES       I
C     YT2    REAL ARRAY(NTRIA)   Y POSITIONS OF SECOND VERTICES       I
C     FT2    REAL ARRAY(NTRIA)   FUNCTION VALUES AT SECOND VERTICES   I
C     XT3    REAL ARRAY(NTRIA)   X POSITIONS OF THIRD VERTICES        I
C     YT3    REAL ARRAY(NTRIA)   Y POSITIONS OF THIRD VERTICES        I
C     FT3    REAL ARRAY(NTRIA)   FUNCTION VALUES AT THIRD VERTICES    I
C     FL     REAL                VALUE OF F AT WHICH CONTOUR IS DRAWN I
C     NX     INTEGER             MAXIMUM VALUE OF X POSITIONS         I
C     NY     INTEGER             MAXIMUM VALUE OF Y POSITIONS         I
C
C      PROPOL ASSUMES A CONTOUR CROSSING LIES
C     BETWEEN (XT1,YT1) AND (XT2,YT2). THE VERTICES MUST BE NUMBERED
C     COUNTER-CLOCKWISE AROUND THE TRIANGLE. THE CODE IN PROPOL IS SPEC-
C     IALIZED TO THE RECTANGULAR X-Y CASE BY THE TRANSFORMATION OF DO
C     LOOP 145. PLOAR PLOTS CAN BE FORMED BY CHANGING THIS TRANSFORMA-
C     TION.
C
C     ENTRY POINTS:  CNTSET, CNTFRM (SEE DOCUMENTATION OR LISTING BELOW)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      IF (NTRIA .GT. NPT) NTRIA = NPT
      XD = (XMN2-XMN1) / FLOAT(NX-1)
      XB = ((XMN2-XMN1)-(XMN3-XMN4)) / FLOAT((NY-1)*(NX-1))
      XC = (XMN4 - XMN1) / FLOAT(NY-1)
      YD = (YMN4 - YMN1 ) / FLOAT(NX-1)
      YB = ((YMN4-YMN1)-(YMN3-YMN2)) / FLOAT((NY-1)*(NX-1))
      YC = (YMN2 - YMN1) / FLOAT(NX-1)
```

```
C
          DO 100 I = 1, NTRIA
          DIFF3(I) = FT2(I) - FT1(I)
          DIFF1(I) = FT3(I) - FT1(I)
          DIFF2(I) = FT3(I) - FT2(I)
          SGN3(I) = SIGN (1.0, DIFF3(I))
          SGN1(I) = SIGN (1.0, DIFF1(I))
  100     SGN2(I) = SIGN (1.0, DIFF2(I))
          DO 110 I = 1, NTRIA
          DIFF3(I) = ABS(DIFF3(I)) + DELTA
          DIFF1(I) = ABS(DIFF1(I)) + DELTA
          DIFF2(I) = ABS(DIFF2(I)) + DELTA
          DFL3(I) = SGN3(I)*(FL - FT1(I))
          DFL1(I) = SGN1(I)*(FL - FT1(I))
  110     DFL2(I) = SGN2(I)*(FL - FT2(I))
          DO 120 I = 1, NTRIA
          ALF3(I) = DFL3(I)/DIFF3(I)
          ALF1(I) = DFL1(I)/DIFF1(I)
          ALF2(I) = DFL2(I)/DIFF2(I)
          X3(I) = XT1(I) + ALF3(I)*(XT2(I) - XT1(I))
          Y3(I) = YT1(I) + ALF3(I)*(YT2(I) - YT1(I))
          X1(I) = XT1(I) + ALF1(I)*(XT3(I) - XT1(I))
          Y1(I) = YT1(I) + ALF1(I)*(YT3(I) - YT1(I))
          X2(I) = XT2(I) + ALF2(I)*(XT3(I) - XT2(I))
          Y2(I) = YT2(I) + ALF2(I)*(YT3(I) - YT2(I))
  120     TEST(I) = (1.0 - ALF2(I))*ALF2(I)
C
C     IF TEST IS LESS THAN ZERO, THEN MISS, OTHERWISE HIT.
C
          DO 130 I = 1, NTRIA
          MASK(I) = LSHF  (TEST(I), -31)
          MASK(I) = MASK(I) - 1
          X2(I) = AND (RMASK(I), X2(I))
  130     Y2(I) = AND (RMASK(I), Y2(I))
          DO 140 I = 1, NTRIA
          MASK(I) = -1 - MASK(I)
          X1(I) = AND (RMASK(I), X1(I))
          Y1(I) = AND (RMASK(I), Y1(I))
          X1(I) = X1(I) + X2(I)
  140     Y1(I) = Y1(I) + Y2(I)
C
C     NOW PLOT THE LINE SEGMENTS.
C
          DO 145 I = 1, NTRIA
          X1(I) = (X1(I)-1.)*(XD-XB*(Y1(I)-1.))+XC*(Y1(I)-1.)+XMN1
          X3(I) = (X3(I)-1.)*(XD-XB*(Y3(I)-1.))+XC*(Y3(I)-1.)+XMN1
          Y1(I) = (Y1(I)-1.)*(YD-YB*(X1(I)-1.))+YC*(X1(I)-1.)+YMN1
  145     Y3(I) = (Y3(I)-1.)*(YD-YB*(X3(I)-1.))+YC*(X3(I)-1.)+YMN1
          DO 150 I = 1, NTRIA
          X3(I) = AMIN1(XMAX, AMAX1(XMIN, X3(I)))
          X1(I) = AMIN1(XMAX, AMAX1(XMIN, X1(I)))
          Y3(I) = AMIN1(YMAX, AMAX1(YMIN, Y3(I)))
  150     Y1(I) = AMIN1(YMAX, AMAX1(YMIN, Y1(I)))
          CALL LINES (X3, Y3, X1, Y1, NTRIA)
          NTRIA = 0
```

```
      RETURN
C
C
C     ------------------------------------------------------------------
C
      ENTRY CNTSET (XM1, YM1, XM2, YM2, XM3, YM3, XM4, YM4)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DESCRIPTION:  CNTSET MAY BE CALLED BY THE USER TO MOVE THE CONTOUR
C      PLOT GENERATED AROUND ON THE PLOTTING REGION OR TO STRETCH OR
C     COMPRESS THE PLOT. THE DATA STATEMENT GIVES THE DEFAULT FOR A
C     LARGE SQUARE PLOT. THE PLOT WILL EXTEND FROM XXMIN TO XXMAX IN THE
C     HORIZONTAL AND FROM YYMIN TO YYMAX IN THE VERTICAL. ALL FOUR OF
C     THESE VALUES SHOULD BE IN THE RANGE 1 TO 1023.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      XMN1 = XM1
      YMN1 = YM1
      XMN2 = XM2
      YMN2 = YM2
      XMN3 = XM3
      YMN3 = YM3
      XMN4 = XM4
      YMN4 = YM4
      XMAX = MAX(XMN1, XMN2, XMN3, XMN4)
      XMIN = MIN(XMN1, XMN2, XMN3, XMN4)
      YMAX = MAX(YMN1, YMN2, YMN3, YMN4)
      YMIN = MIN(YMN1, YMN2, YMN3, YMN4)
      RETURN
C
C
C     ------------------------------------------------------------------
C
      ENTRY CNTFRM
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DESCRIPTION:  CNTFRM IS A USER-CALLED ROUTINE TO PLOT THE RECTAN-
C     GULAR BOUNDARY OF THE CONTOURED REGION. CNTFRM HAS NO ARGUMENTS
C     SINCE THE REGION IS SPECIFIED BY DEFAULT OR VIA A PREVIOUS CALL TO
C     CNTSET.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      CALL LINE(XMN1,YMN1,XMN2,YMN2)
      CALL LINE(XMN2,YMN2,XMN3,YMN3)
      CALL LINE(XMN3,YMN3,XMN4,YMN4)
      CALL LINE(XMN4,YMN4,XMN1,YMN1)
      RETURN
      END
```

```
      SUBROUTINE PUTCH (X, Y, CHARAC)
C
      INTEGER LPAGE(30, 65)
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      CHARACTER*1 CHARAC, CH(4)
      EQUIVALENCE (CH,ICH)
      INTEGER K, K1, M, IK, IM, MASK(5)
      INTEGER ISHFT, AND, OR
      LOGICAL LSW, LTSW
      DATA MASK /ZFFFF00FF, ZFF00FFFF, Z00FFFFFF, ZFFFFFF00,
     .     ZFF000000/, CHSZ/0.0/
C
      ICH = 0
      CH(1) = CHARAC
      CHS = CHSIZE(2)
      GO TO  (1,2,3,5), IDEVIC
C
C     CALCOMP SECTION.
C
 1    CALL CALSYM (X, Y, CHS, CH, 0, 0.0, 1)
      LSW = .TRUE.
      RETURN
C
C     PAGE PLOT SECTION.
C
 2    JX1 = MAX(0, MIN(IFIX(X),1023))
      JY1 = MAX(0, MIN(IFIX(Y),1023))
      I = (JX1 + 5)/10 + 1
      J = 65 - (JY1+8)/16
      K = I/4
      M = I - 4*K
      K1 = 4 - M
      IF(M .EQ. 0) K = K - 1
      IM = IAND(LPAGE(K+4, J), MASK(K1))
      IK = IAND(ICH, MASK(5))
      IF(M .EQ. 0) IK = ISHFT(IK, -24)
      IF(M .EQ. 3) IK = ISHFT(IK, -16)
      IF(M .EQ. 2) IK = ISHFT(IK, -8)
      LPAGE(K+4, J) = IOR(IK, IM)
      RETURN
C
C     LEXIDATA/MATRIX SECTION
C
 3    CALL GSCHSZ (CHS)
      XL = MAX(X,0.)
      YL = MAX(Y,0.)
      CALL GMOVA (XL, YL)
      CALL GTXTWC (CH, 1)
      RETURN
C
C     TEKTRONIX SECTION
C
 5    CONTINUE
      CALL MOVEA(X,Y)
```

```
          CALL A1OUT(1,ICH)
          LTSW=.TRUE.
          RETURN
      END
```

```
      SUBROUTINE PUTLNV (X1, Y1, DX, DY,  NL)
C
      PARAMETER (NLINES=200)
      INTEGER LENGTH, COUNT
      REAL X1(NL), Y1(NL), DX(NL), DY(NL)
      INTEGER MASK1/15/,OP11/12/,IVERT/1023/
      INTEGER LPAGE(30,65), HSYMBL, HDOT, HAPOS, HMINUS
      REAL JX1(NLINES),JY1(NLINES),JX2(NLINES),JY2(NLINES)
      REAL XY(2,NLINES), JXS, JYS, IDDX, IDDY
      COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      COMMON/GRFMOD/LFLMAT(5),LINWDM(5),LCLMAT(5)
      INTEGER K, K1, M, M1, IK, IM, MASK(4)
      LOGICAL LSW, LTSW, IRVRSE, LEXLSW
      DATA HDOT, HAPOS, HMINUS /Z0000002E, Z00000027, Z0000002D /
      DATA MASK /ZFFFF00FF, ZFF00FFFF, Z00FFFFFF, ZFFFFFF00 /
      DATA JXS, JYS / 2*0.0 /
C
      IF(IDEVIC.EQ.0) RETURN
      IF(NL.GT.NLINES.AND.LUDIAG.GT.0) WRITE(LUDIAG,1002)
 1002 FORMAT(' *** BUFFER SPACE EXCEEDED IN PUTLNV ***')
      NLP = MIN0(NLINES,NL)
C
C   MASK THE LINE SEGMENT FOR CALCOMP AND PAPER, AND TEKTRONIX.
C
      DO 10 I=1,NLP
      JX1(I) = X1(I)
      JY1(I) = Y1(I)
      JX2(I) = JX1(I) + DX(I)
   10 JY2(I) = JY1(I) + DY(I)
      DO 11 I = 1, NLP
      JX1(I) = MAX(JX1(I),0.)
      JY1(I) = MAX(JY1(I),0.)
      JX2(I) = MAX(JX2(I),0.)
   11 JY2(I) = MAX(JY2(I),0.)
      GO TO (1,2,3,5), IDEVIC
C
C   CALCOMP SECTION.
C
    1 DO 16 I = 1, NLP
      IF(LSW) GO TO 12
      IF(JXS.EQ.JX1(I) .AND. JYS.EQ.JY1(I))GO TO 15
   12 CALL CALPLT(JX1(I), JY1(I), 3)
      LSW =.FALSE.
   15 CALL CALPLT(JX2(I), JY2(I), 2)
      JXS=JX2(I)
      JYS=JY2(I)
   16 CONTINUE
      RETURN
C
C   PAGE PLOT SECTION.
C
    2 DO 22 I=1,NLP
      HSYMBL = HDOT
      IF(DX(I).GT.2*DY(I))HSYMBL=HMINUS
```

```
             IF(DY(I).GT.2*DX(I))HSYMBL=HAPOS
             IDDX=JX2(I)-JX1(I)
             IDDY=JY2(I)-JY1(I)
             DO 20 L = 1, 9
             JX=JX1(I)+((L-1)*IDDX)/8
             JY=JY1(I)+((L-1)*IDDY)/8
             IX= (JX+5)/10 + 1
             IY= 65 - (JY+8)/16
             K=IX/4
             M = IX- 4*K
             IK = HSYMBL
             K1 = 4 - M
             M1 = 8*K1
             IF(M .EQ. 0) K = K - 1
             IM = IAND(LPAGE(K+4,IY), MASK(K1))
             IF( M .NE. 0) IK = ISHFT(HSYMBL, M1)
      20     LPAGE(K+4,IY) = IOR(IM, IK)
      22     CONTINUE
             RETURN
C
C      LEXIDATA/MATRIX SECTION
C
       3     IL = 1
      32     IS = 1
             CALL GMOVA(JX1(IL),JY1(IL))
             XY(1,IS) = JX2(IL)
             XY(2,IS) = JY2(IL)
      31     IL = IL + 1
             IF(JX1(IL).NE.XY(1,IS).OR.JY1(IL).NE.XY(2,IS).OR.IL.GT.NLP)
        .    GO TO 30
             IS = IS + 1
             XY(1,IS) = JX2(IL)
             XY(2,IS) = JY2(IL)
             GO TO 31
      30     CALL GPLNA(XY,IS)
             IF(IL.GT.NLP) RETURN
             GO TO 32
             RETURN
C
C      TEKTRONIX SECTION
C
       5     CONTINUE
             DO 40 I=1,NLP
             IF(LTSW) GO TO 42
             IF(JXS.EQ.JX1(I).AND.JYS.EQ.JY1(I)) GO TO 45
      42     CALL MOVEA(JX1(I),JY1(I))
             LTSW=.FALSE.
      45     CALL DRAWA(JX2(I),JY2(I))
             JXS=JX2(I)
             JYS=JY2(I)
      40     CONTINUE
             RETURN
       END
```

```
      SUBROUTINE SETLUT
C
      DIMENSION LUT(48)
      DATA LUT/0,15,15,0,0,15,15,0,15,1,3,5,7,9,11,13,
     . 0,15,0,15,0,15,0,15,8,1,3,5,7,9,11,13,
     . 0,15,0,0,15,0,15,15,0,1,3,5,7,9,11,13/
C
      CALL DSLWT (16, 48, LUT)
      RETURN
      END
```

```
      SUBROUTINE SRFSET(XX, YY, ZZMIN, ZZMAX, NX, NY)
C
C     THIS SUBROUTINE AND ITS ASSOCIATED ENTRIES CONSTRUCT AND
C     DELIVER PLOTS OF A SURFACE Z(I,J) WITH HIDDEN LINES REMOVED
C     FOR I = 1, ..., NX AND J = 1, ..., NY.  THE TYPE, ORIENTATION,
C     AND DETAILS OF THE PLOTS ARE QUITE FLEXIBLE AS SEEN BY CAREFUL
C     STUDY OF THE TEST PROGRAM AND THE RESULTING OUTPUT
C     THE PARTICULAR VERSION IS AN INTERFACE TO THE GENERAL PURPOSE
C     PLOTTING PACKAGE WHICH CAN DRAW COLOR PLOTS ON THE CALCOMP,
C     LEXIDATA, PRINTER OR TEKTRONICS (4000'S SERIES).
C
C     SURFACE INITIALIZES THE SURFACE PLOTTING PACKAGE AND MUST
C     BE CALLED EACH TIME A NEW PLOT IS DESIRED TO RESET THE
C     HIDDEN LINE ARRAYS.
C     NOTE THAT MAX(NX,NY) MUST BE LESS THAN NPT
C
C     THE LOGICAL PLOTTING REGION IS A 3D RECTANGULAR PARALLELE-
C     PIPED WITH 8 CORNER VERTICES.
C     XX     REAL    - ARRAY DIMENSIONED 8 CONTAINING THE 8 CORNER
C                      VERTEX X LOCATIONS. THE VERTICES ARE NUMBERED
C                      AS SHOWN ON THE SURFACE GEOMETRY SHEET (AVAILABL
C                      FROM THE SPL LIBRARIAN OR THE AUTHOR).
C     YY     REAL    - ARRAY DIMENSIONED 8 CONTAINING THE 8 CORNER
C                      VERTEX Y LOCATIONS.  THE VERTICES ARE NUMBERED
C                      AS SHOWN ON THE SURFACE GEOMETRY SHEET WHICH IS
C                      AVAILABLE FROM THE SPL LIBRARIAN OR THE AUTHOR.
C     ZZMIN  REAL    - THE VALUE OF Z(I,J) TO BE PLOTTED AT THE
C                      BOTTOM SURFACE OF THE PARALLELEPIPED.
C                      SMALLER Z(I,J) ARE SET (I.E. LIMITED) TO
C                      ZZMIN.
C     ZZMAX  REAL    - THE VALUE OF Z(I,J) TO BE PLOTTED AT THE
C                      UPPER SURFACE OF THE PARALLELEPIPED.
C                      LARGER Z(I,J) ARE SET (I.E. LIMITED) TO
C                      ZZMAX.
C     NX     INTEGER - DIMENSION AND RANGE OF I IN Z(I,J)
C     NY     INTEGER - DIMENSION AND RANGE OF J IN Z(I,J)
C
C     SFRAME (MODE1) - PLOTS THE FRAME OF THE RECTANGULAR PARALLELE-
C            PIPED PLOTTING REGION.
C     MODE1  INTEGER - MODE1 = 1  PLOTS + AT THE VERTICES
C                            = 2  ALSO PLOTS LINE SEGMENTS
C                                 CONNECTING THE VERTICES.
C
C     SSKIRT (ZZ, NX, NY, MODE2) - PLOTS ANY OF THE SKIRTS WHICH MAY
C            BE DESIRED.  ONLY ONE SKIRT IS PLOTTED PER CALL AND THE
C            HIDDEN LINE ALGORITHM IS INVOKED.  THEREFORE, THE SIDE
C            SKIRTS (+2, -2, +4, -4) AND THE DATA SURFACE BACK SKIRTS
C            (+3, -3) SHOULD ONLY BE PLOTTED AFTER THE DATA SURFACE
C            HAS BEEN CONSTRUCTED USING SURFACE.  THIS ENTRY DOES NOT
C            PLOT THE Z SURFACE (HERE THE DATA ARE CALLED ZZ TO AVOID
C            DECLARATION CONFLICTS).  HOWEVER Z(I,J) ARE NEEDED TO
C            DEFINE THE SKIRT POSITIONS.
C     ZZ     REAL    - ARRAY CONTAINING THE DATA TO BE PLOTTED AS A
C                      SURFACE
C     NX     INTEGER - DIMENSION AND RANGE OF I IN Z(I,J)
```

```
C       NY     INTEGER - DIMENSION AND RANGE OF J IN Z(I,J)
C       MODE2  INTEGER - MODE2 = 1(-1) LOWER (UPPER) FRONT SKIRT
C                              = 2(-2) LOWER (UPPER) RIGHT SKIRT
C                              = 3(-3) LOWER (UPPER) BACK SKIRT
C                              = 4(-4) LOWER (UPPER) LEFT SKIRT
C       SURFAC (Z, NX, NY, MODE3) - CONSTRUCTS AND PLOTS THE DATA SUR-
C              FACE AND SETS THE HIDDEN LINE ARRAYS WHICH ARE USED IN
C              SURFSK.
C       Z      REAL    - ARRAY DIMENSIONED (NX,NY) CONTAINING THE DATA
C                        TO BE PLOTTED AS A SURFACE
C       NX     INTEGER - DIMENSION AND RANGE OF I IN Z(I,J)
C       NY     INTEGER - DIMENSION AND RANGE OF J IN Z(I,J)
C       MODE3  INTEGER - MODE3 = 1  PLOTS THE UPPER SURFACE
C                              =-1  PLOTS THE LOWER SURFACE
C
          PARAMETER (NPT=101)
          LOGICAL  SW(1024)
          REAL     XX(8), YY(8)
          REAL     X(8), Y(8), H(1280), G(1280)
C         REAL     AJ1(NPT), AJN(NPT), AI1(NPT), AIN(NPT), ZVAL(NPT)
C         REAL     RDZVB(NPT), RDZTV(NPT)
          LOGICAL  SWITCH
          REAL     Z(NX,NY), X1(NPT), X2(NPT), Y1(NPT), Y2(NPT)
          REAL     ZZ(NX, NY)
          COMMON /SURCMN/  X, Y, ZMIN, ZMAX, RDZ, RNXM1, RNYM1, NNX, NNY
C
C      INITIALIZE HIDDEN LINE ARRAYS.
C
          NNX = NX
          NNY = NY
          IF(MAX0(NX,NY).GT.NPT) STOP 76
          YMIN = 1.E+25
          YMAX = -1.E+25
          DO 3 I = 1, 8
          YMIN = AMIN1(YMIN, YY(I))
    3     YMAX = AMAX1(YMAX, YY(I))
          DO 1 I = 1, 1280
          G(I) = YMAX
    1     H(I) = YMIN
C
C      FOR SPECIAL EFFECTS ZBOT AND ZTOP MAY DIFFER FROM ZMIN AND ZMAX.
C
          ZBOT = ZZMIN
          ZTOP = ZZMAX
          ZMIN = ZZMIN
          ZMAX = ZZMAX
          RDZ = 1.0/(ZTOP - ZBOT)
          DO 2 I = 1, 8
          X(I) = XX(I)
    2     Y(I) = YY(I)
          RNXM1 = 1.0/FLOAT(NX - 1)
          RNYM1 = 1.0/FLOAT(NY - 1)
          RETURN
C
C
```

```
C     ----------------------------------------------------------------
C
          ENTRY SFRAME (MODE1)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DESCRIPTION:  SURFRM PLOTS THE FRAME OF THE RECTANGULAR PARALLELO-
C     PIPED PLOTTING REGION.
C
C      ARGUMENTS:
C     MODE1  INTEGER                 MODE1 = 1  PLOTS + AT THE VERTICES.
C                                    MODE1 = 2  ALSO PLOTS LINE SEGMENTS
C                                               CONNECTING THE VERTICES.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
          CALL CHPLOT (X, Y, '+', 1, 1, 8)
C
C     NOW CHECK THE VARIOUS LINES.
C
          IF (MODE1 .LE. 1) GO TO 12
          CALL SURF5 (X(1), Y(1), X(2), Y(2))
          CALL SURF5 (X(6), Y(6), X(2), Y(2))
          CALL SURF5 (X(6), Y(6), X(5), Y(5))
          CALL SURF5 (X(1), Y(1), X(5), Y(5))
          IF (Y(7).GT.Y(5) .OR. X(7).LT.X(5))
     1        CALL SURF5 (X(5), Y(5), X(7), Y(7))
          IF (Y(8).GT.Y(6) .OR. X(8).GT.X(6))
     1        CALL SURF5 (X(8), Y(8), X(6), Y(6))
          IF (Y(3).LT.Y(1) .OR. X(3).LT.X(1))
     1        CALL SURF5 (X(1), Y(1), X(3), Y(3))
          IF (Y(4).LT.Y(2) .OR. X(4).GT.X(2))
     1        CALL SURF5 (X(2), Y(2), X(4), Y(4))
          IF (Y(7).GE.Y(5) .AND. Y(8).GE.Y(6))
     1        CALL SURF5 (X(8), Y(8), X(7), Y(7))
          IF (X(8).GE.X(6) .AND. X(4).GE.X(2))
     1        CALL SURF5 (X(8), Y(8), X(4), Y(4))
          IF (Y(4).LE.Y(2) .AND. Y(3).LE.Y(1))
     1        CALL SURF5 (X(4), Y(4), X(3), Y(3))
          IF (X(3).LE.X(1) .AND. X(7).LE.X(5))
     1        CALL SURF5 (X(7), Y(7), X(3), Y(3))
   12     CONTINUE
          RETURN
C
C
C     ----------------------------------------------------------------
C
      ENTRY SSKIRT (ZZ, NX, NY, MODE2)
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DESCRIPTION:  SURFSK PLOTS ANY OF THE SKIRTS WHICH MAY BE DESIRED.
C     ONLY ONE SKIRT IS PLOTTED PER CALL AND THE HIDDEN LINE ALGORITHM
C     IS INVOKED. THEREFORE, THE SIDE SKIRTS (+2, -2, +4, -4) AND THE
```

```
C     BACK SKIRTS (+3, -3) SHOULD ONLY BE PLOTTED AFTER THE DATA SURFACE
C     HAS BEEN CONSTRUCTED USING SURFACE. THIS ENTRY DOES NOT PLOT THE Z
C     SURFACE (HERE THE DATA ARE CALLED ZZ TO AVOID DECLARATION CON-
C     FLICTS). HOWEVER Z(I,J) ARE NEEDED TO DEFINE THE SKIRT POSITIONS.
C
C      ARGUMENTS:
C     ZZ      REAL ARRAY (NX,NY)  THE DATA TO BE PLOTTED AS A SURFACE. I
C     NX      INTEGER             DIMENSION AND RANGE OF I IN Z(I,J).  I
C     NY      INTEGER             DIMENSION AND RANGE OF J IN Z(I,J).  I
C     MODE2   INTEGER             MODE2 = 1(-1) LOWER (UPPER) FRONT SKIRT
C                                 MODE2 = 2(-2) LOWER (UPPER) RIGHT SKIRT
C                                 MODE2 = 3(-3) LOWER (UPPER) BACK SKIRT
C                                 MODE2 = 4(-4) LOWER (UPPER) LEFT SKIRT
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
          NSKIRT = IABS(MODE2)
          GO TO (21, 22, 23, 22), NSKIRT
C
C     THE FRONT SKIRT IS ADDED.
C
   21     CONTINUE
          XL = X(1)
          XR = X(2)
          YL = Y(1)
          YR = Y(2)
          IF (MODE2 .GT. 0) GO TO 211
          XL = X(5)
          XR = X(6)
          YL = Y(5)
          YR = Y(6)
  211     DO 212 I = 1, NX
          FI1 = FLOAT(I-1)/FLOAT(NX-1)
          FIN = FLOAT(NX-I)/FLOAT(NX-1)
          XA = XR*FI1 + XL*FIN
          YA = YR*FI1 + YL*FIN
          CALL SURF4 (XB,YB, I,1, ZZ(I,1))
  212     CALL SURF5 (XA, YA, XB, YB)
          RETURN
C
C     THE RIGHT OR LEFT SIDE SKIRT IS ADDED. IF IT IS OBSCURED THE
C     HIDDEN LINE ALGORITHMS ARE USED SO CALL ONLY AFTER SURFACE IS USED
   22     NSK = (4 - NSKIRT)/2
          I = 1 + (NX-1)*NSK
          XST = X(NSK+1)
          YST = Y(NSK+1)
          XND = X(NSK+3)
          YND = Y(NSK+3)
          IF (MODE2 .GT. 0) GO TO 221
          XST = X(NSK+5)
          YST = Y(NSK+5)
          XND = X(NSK+7)
          YND = Y(NSK+7)
  221     DO 222 J = 1, NY
          FJ1 = FLOAT(J-1)/FLOAT(NY-1)
```

```
            FJN = FLOAT(NY-J)/FLOAT(NY-1)
            XA = XND*FJ1 + XST*FJN
            YA = YND*FJ1 + YST*FJN
            CALL SURF4 (XB,YB, I,J, ZZ(I,J))
C
C     THE SKIRT MAY BE OBSCURED. CHECK ON TOP OR BOTTOM.
C
            K = XB + 0.5
            IF (MODE2 .GT. 0) YB = AMIN1(YB, G(K))
            IF (MODE2 .LT. 0) YB = AMAX1(YB, H(K))
            IF (MODE2.GT.0 .AND. YA.LT.G(K)) CALL SURF5 (XA, YA, XB, YB)
            IF (MODE2.LT.0 .AND. YA.GT.H(K)) CALL SURF5 (XA, YA, XB, YB)
  222       CONTINUE
            RETURN
C
C     THE BACK SKIRT IS ADDED (ASSUMED CALLED AFTER SURFACE).
C
   23       CONTINUE
            XL = X(3)
            YL = Y(3)
            XR = X(4)
            YR = Y(4)
            IF (MODE2.GT.0) GO TO 231
            XL = X(7)
            YL = Y(7)
            XR = X(8)
            YR = Y(8)
  231       DO 232 I = 1, NX
            FI1 = FLOAT(I-1)/FLOAT(NX-1)
            FIN = FLOAT(NX-I)/FLOAT(NX-1)
            XA = XR*FI1 + XL*FIN
            YA = YR*FI1 + YL*FIN
            CALL SURF4 (XB,YB, I,NY, ZZ(I,NY))
            K = XB + 0.5
            IF (MODE2 .GT. 0) YB = AMIN1 (YB, G(K))
            IF (MODE2 .LT. 0) YB = AMAX1 (YB, H(K))
            IF (FLOAT(MODE2)*(YB-YA) .GT. 0.0) CALL SURF5(XA, YA, XB, YB)
  232       CONTINUE
            RETURN
C
C
C     ------------------------------------------------------------------
C
      ENTRY SURFAC (Z, NX, NY, MODE3)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DESCRIPTION: SURFACE CONSTRUCTS AND PLOTS THE DATA SURFACE AND SET
C     THE HIDDEN LINE ARRAYS WHICH ARE USED IN SURFSK.
C
C      ARGUMENTS:
C     Z      REAL ARRAY (NX,NY)  THE DATA TO BE PLOTTED AS A SURFACE.  I
C     NX     INTEGER             DIMENSION AND RANGE OF I IN Z(I,J).   I
C     NY     INTEGER             DIMENSION AND RANGE OF J IN Z(I,J).   I
C     MODE3  INTEGER             MODE3 = 1   PLOTS THE UPPER SURFACE   I
```

```
            YA = AMAX1(Y1(I), H(IXH))
            YB = AMAX1(Y2(I), H(IXH))
            IF (YB.GT.YA) CALL SURF5 (X1(I),YA, X2(I),YB)
            GO TO 35
   39       YA = AMIN1(Y1(I), G(IXH))
            YB = AMIN1(Y2(I), G(IXH))
            IF (YB.LT.YA) CALL SURF5 (X1(I),YA, X2(I),YB)
            GO TO 35
C
C     THIS IS A SLANTED LINE SEGMENT.
C
   38       DIX = 1.0/FLOAT(IXB - IXA)
            SWITCH = .TRUE.
            NSWT = IXB - IXA + 1
            DO 36 K = IXA, IXB
            SW(K) = .TRUE.
            IF (X2(I).GT.X1(I)) YK = Y1(I)*(IXB-K)*DIX + Y2(I)*(K-IXA)*DIX
            IF (X2(I).LT.X1(I)) YK = Y2(I)*(IXB-K)*DIX + Y1(I)*(K-IXA)*DIX
            IF ((MODE3.LT.0 .AND. G(K).GE.YK-0.5)
     1          .OR. (MODE3.GT.0 .AND. H(K).LE.YK+0.5)) GO TO 36
            SWITCH = .FALSE.
            SW(K) = .FALSE.
            NSWT = NSWT - 1
   36       CONTINUE
             IF (SWITCH) CALL SURF5 (X1(I), Y1(I), X2(I), Y2(I))
            IF (SWITCH .OR. NSWT.EQ.0) GO TO 35
C
C     PART OF THE LINE IS OBSCURED SO WE NEED TO PLOT SEGMENTS.
C
            SW(IXA-1) = .FALSE.
            SW(IXB+1) = .FALSE.
            DO 41 K = IXA, IXB
            IF (X2(I).GT.X1(I)) YK = Y1(I)*(IXB-K)*DIX + Y2(I)*(K-IXA)*DIX
            IF (X2(I).LT.X1(I)) YK = Y2(I)*(IXB-K)*DIX + Y1(I)*(K-IXA)*DIX
            IF (.NOT.SW(K) .OR. SW(K-1)) GO TO 51
            XA = K
            YA = YK
            IF (K.EQ.IXA) GO TO 51
            IF (MODE3.GT.0) YA = AMIN1 (H(K), YK)
            IF (MODE3.LT.0) YA = AMAX1 (G(K), YK)
   51       IF (.NOT.SW(K) .OR. SW(K+1)) GO TO 41
            IF (K.EQ.IXB) GO TO 52
            IF (MODE3.GT.0) YK = AMIN1 (H(K), YK)
            IF (MODE3.LT.0) YK = AMAX1 (G(K), YK)
   52       CALL SURF5 (XA,YA,FLOAT(K),YK)
   41       CONTINUE
   35       CONTINUE
   37       CONTINUE
C
C     PLOT THE I TO I+1 LINE SEGMENTS IF NOT HIDDEN.
C
            DO 33 I = 2, NX
            IXB = X2(I) + 0.5
            IXA = X2(I-1)
            DIX = 1.0/FLOAT(IXB - IXA)
```

```
            SWITCH = .TRUE.
            NSWT = IXB - IXA + 1
            DO 34 K = IXA, IXB
            SW(K) = .TRUE.
            YK = Y2(I-1)*(IXB-K)*DIX + Y2(I)*(K-IXA)*DIX
            IF (MODE3.GT.0) H(K) = AMAX1(H(K), YK)
            IF (MODE3.LT.0) G(K) = AMIN1(G(K), YK)
            IF ((MODE3.LT.0 .AND. G(K).GE.YK-0.5)
     1          .OR. (MODE3.GT.0 .AND. H(K).LE.YK+0.5)) GO TO 34
            SWITCH = .FALSE.
            SW(K) = .FALSE.
            NSWT = NSWT - 1
   34       CONTINUE
            IF (SWITCH) CALL SURF5 (X2(I-1), Y2(I-1), X2(I), Y2(I))
            IF (SWITCH .OR. NSWT.EQ.0) GO TO 33
C
C        PART OF THE LINE IS OBSCURED SO WE NEED TO PLOT SEGMENTS.
C
            SW(IXA-1) = .FALSE.
            SW(IXB+1) = .FALSE.
            DO 44 K = IXA, IXB
            YK = Y2(I-1)*(IXB-K)*DIX + Y2(I)*(K-IXA)*DIX
            IF (SW(K) .AND. .NOT.SW(K-1)) XA = K
            IF (SW(K) .AND. .NOT.SW(K-1)) YA = YK
            IF (SW(K) .AND. .NOT.SW(K+1)) CALL SURF5 (XA,YA,FLOAT(K),YK)
   44       CONTINUE
   33       CONTINUE
C
   31       CONTINUE
            RETURN
         END
```

```
      SUBROUTINE SURF4 (XVAL, YVAL, I, J, ZIN)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     THIS IS AN AUXILIARY ROUTINE TO SURFACE. IT PERFORMS A TRILINEAR I
C     TERPOLATION IN THE 3D RECTANGLE OUTLINED BT (X(I), Y(I)) ON THE
C     PLOTTING SURFACE FOR I = 1, ..., 8. THE TEXT WILL BE SUBSTITUTED
C     IN LINE WHEN THE ASC COMPILER BUG HAS BEEN FIXED.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
          REAL     X(8), Y(8)
          COMMON /SURCMN/  X, Y, ZMIN, ZMAX, RDZ, RNXM1, RNYM1, NNX, NNY
C
          FI1 = FLOAT(I-1)*RNXM1
          FJ1 = FLOAT(J-1)*RNYM1
          FIN = FLOAT(NNX-I)*RNXM1
          FJN = FLOAT(NNY-J)*RNYM1
          ZVAL = AMAX1 (ZIN, ZMIN)
          ZVAL = AMIN1 (ZVAL, ZMAX)
          XVAL  = RDZ*(ZVAL-ZMIN)*(FJN*(FI1*X(6) + FIN*X(5))
     1              + FJ1*(FI1*X(8) + FIN*X(7)))
     2           + RDZ*(ZMAX-ZVAL)*(FJN*(FI1*X(2) + FIN*X(1))
     3              + FJ1*(FI1*X(4) + FIN*X(3)))
          YVAL  = RDZ*(ZVAL-ZMIN)*(FJN*(FI1*Y(6) + FIN*Y(5))
     1              + FJ1*(FI1*Y(8) + FIN*Y(7)))
     2           + RDZ*(ZMAX-ZVAL)*(FJN*(FI1*Y(2) + FIN*Y(1))
     3              + FJ1*(FI1*Y(4) + FIN*Y(3)))
          RETURN
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     THIS SUBROUTINE IS A GENERAL-PURPOSE GRAPHICS INTERFACE FOR USE
C     WITH THE SURFACE SUBROUTINE FOR 3D SURFACE PLOTS WITH HIDDEN LINES
C     REMOVED.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      ENTRY SURF5 (X1, Y1, X2, Y2)
C
          IF ( ((X1-X2)**2 + (Y1-Y2)**2) .GT. 2.25)
     1        CALL LINE (X1, Y1, X2, Y2)
          RETURN
      END
```

```
      SUBROUTINE SYMBOL (XCHAR, YCHAR, CHARAC)
C
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      CHARACTER*1 CHARAC
C
C     THIS SUBROUTINE USES THE HARWARE CHARACTERS TO PLOT DATA
C     POINTS AT THE LOCATION SPECIFIED BY (XCHAR,YCHAR) IN THE PLOTTING
C     REGION DETERMINED BY XYCOOD.
C
      IF (IDEVIC.EQ.4) CALL IOWAIT(8)
      X = XYCOOD(1)*XCHAR + XYCOOD(2) - CHSIZE(2)*CHSIZE(3)
      Y = XYCOOD(3)*YCHAR + XYCOOD(4) - CHSIZE(2)*CHSIZE(4)
      CALL PUTCH (X, Y, CHARAC)
      RETURN
      END
```

```
      SUBROUTINE VIEWTR(XDC,WINDOW,VERTEX,NV,EDGE,ESPEC,NE,
     . POLY,PSPEC,NP)
C
C  THIS SUBROUTINE PLOTS EDGES AND POLYGONS WHICH ARE WITHIN THE WINDOW
C  EDGE SEGMENTS WHICH CROSS A PORTION OF THE WINDOW.
C
      REAL VERTEX(3,NV), WINDOW(4), XYZ(3,8)
      INTEGER EDGE(2,NE), ESPEC(4,NE), POLY(8,NP), PSPEC(4,NP)
      REAL U1,V1,U2,V2,X(8),Y(8),XDC(4,4)
C
C  DEFINE THE WINDOW SPACE
C
      CALL DEFINE(WINDOW(1),WINDOW(2),WINDOW(3),WINDOW(4))
C
      DOZ=10000.
      FANGLE=90.
      TANAL = 1.0
      WCX = 0.5 * (WINDOW(1)+WINDOW(3))
      WCY = 0.5 * (WINDOW(2)+WINDOW(4))
C
C  PLOT ONE EDGE AT A TIME
C
      LWD = -1
      LCL = -1
      LFL = -1
      DO 150 J=1,NE
C
C     MOVE VERTICES INTO A LOCAL WORK ARRAY
C
      DO 151 I = 1, 2
      DO 151 K = 1, 3
  151 XYZ(K,I) = VERTEX(K,EDGE(I,J))
C
C     APPLY THE TRANSFORM
C
      DO 152 I = 1, 2
      U1 = XYZ(1,I)
      U2 = XYZ(2,I)
      U3 = XYZ(3,I)
      XYZ(1,I) = XDC(1,1)*U1 + XDC(1,2)*U2 + XDC(1,3)*U3 + XDC(1,4)
      XYZ(2,I) = XDC(2,1)*U1 + XDC(2,2)*U2 + XDC(2,3)*U3 + XDC(2,4)
  152 XYZ(3,I) = XDC(3,1)*U1 + XDC(3,2)*U2 + XDC(3,3)*U3 + XDC(3,4)
C
C  ADD PERSPECTIVE
C
      D = DOZ/((DOZ+XYZ(3,1))*TANAL)
      U1=(XYZ(1,1)-WCX)*D + WCX
      V1=(XYZ(2,1)-WCY)*D + WCY
      D = DOZ/((DOZ+XYZ(3,2))*TANAL)
      U2=(XYZ(1,2)-WCX)*D + WCX
      V2=(XYZ(2,2)-WCY)*D + WCY
C
C  CLIP LEFT EDGE
C
      IF (U1.GE.WINDOW(1).AND.U2.GE.WINDOW(1)) GO TO 50
```

```
      IF (U1.LT.WINDOW(1).AND.U2.LT.WINDOW(1)) GO TO 150
      IF (U1.GT.WINDOW(1)) GO TO 40
      V1 = (V1-V2)*U1/(U2-U1)+V1
      U1 = WINDOW(1)
      GO TO 50
40    V2 = (V2-V1)*U2/(U1-U2)+V2
      U2 = WINDOW(1)
C
C  CLIP RIGHT EDGE
C
50    IF (U1.LE.WINDOW(3).AND.U2.LE.WINDOW(3)) GO TO 70
      IF (U1.GT.WINDOW(3).AND.U2.GT.WINDOW(3)) GO TO 150
      IF (U1.GT.WINDOW(3)) GO TO 51
      V2 = (V2-V1)*(WINDOW(3)-U1)/(U2-U1)+V1
      U2 = WINDOW(3)
      GO TO 70
51    V1 = (V1-V2)*(WINDOW(3)-U2)/(U1-U2)+V2
      U1 = WINDOW(3)
C
C  CLIP BOTTOM EDGE
C
70    IF (V1.GE.WINDOW(2).AND.V2.GE.WINDOW(2)) GO TO 90
      IF (V1.LT.WINDOW(2).AND.V2.LT.WINDOW(2)) GO TO 150
      IF (V1.GT.WINDOW(2)) GO TO 80
      U1 = (U1-U2)*V1/(V2-V1)+U1
      V1 = WINDOW(2)
      GO TO 90
80    U2 = (U2-U1)*V2/(V1-V2)+U2
      V2 = WINDOW(2)
C
C  CLIP TOP EDGE
C
90    IF (V1.LE.WINDOW(4).AND.V2.LE.WINDOW(4)) GO TO 110
      IF (V1.GT.WINDOW(4).AND.V2.GT.WINDOW(4)) GO TO 150
      IF (V1.GT.WINDOW(4)) GO TO 100
      U2 = (U2-U1)*(WINDOW(4)-V1)/(V2-V1)+U1
      V2  = WINDOW(4)
      GO TO 110
100   U1 = (U1-U2)*(WINDOW(4)-V2)/(V1-V2)+U2
      V1 = WINDOW(4)
C
C  PLOT THE EDGE
C
110    IF(LWD.NE.ESPEC(1,J)) THEN
        LWD = ESPEC(1,J)
        CALL LINWID(LWD)
      ENDIF
      IF(LCL.NE.ESPEC(2,J)) THEN
        LCL = ESPEC(2,J)
        CALL COLOR(LCL)
      ENDIF
      CALL LINE(U1, V1, U2, V2)
150    CONTINUE
C
C     PLOT POLYGONS ON TOP OF EDGES
```

```
C
      DO 10 J = 1, NP
      NUM=0
C
C     DETERMINE THE NUMBER OF VERTICES (NON-ZERO ENTRIES) IN THE
C     POLYGON
C
      DO 15 I=1,8
      IF(POLY(I,J).NE.0) NUM=NUM+1 ;NUMBER OF VERTICES IN POLYGON
   15 CONTINUE
C
C     MOVE THE VERTICES INTO A LOCAL WORK ARRAY
C
      DO 61 I = 1, NUM
      DO 61 K = 1, 3
   61 XYZ(K,I) = VERTEX(K,POLY(I,J))
C
C     APPLY THE TRANSFORM
C
      DO 62 I = 1, NUM
      U1 = XYZ(1,I)
      U2 = XYZ(2,I)
      U3 = XYZ(3,I)
      XYZ(1,I) = XDC(1,1)*U1 + XDC(1,2)*U2 + XDC(1,3)*U3 + XDC(1,4)
      XYZ(2,I) = XDC(2,1)*U1 + XDC(2,2)*U2 + XDC(2,3)*U3 + XDC(2,4)
   62 XYZ(3,I) = XDC(3,1)*U1 + XDC(3,2)*U2 + XDC(3,3)*U3 + XDC(3,4)
C
C     ADD PERSPECTIVE
C
      DO 60 IV=1,NUM
      D = DOZ/((DOZ+XYZ(3,IV))*TANAL)
      X(IV) = (XYZ(1,IV) - WCX)*D  +  WCX
   60 Y(IV) = (XYZ(2,IV) - WCY)*D  +  WCY
C
C  PLOT THE POLYGON
C
      IF(PSPEC(1,J).NE.LWD) THEN
        LWD = PSPEC(1,J)
        CALL LINWID(LWD)
      ENDIF
      IF(PSPEC(2,J).NE.LCL) THEN
        LCL = PSPEC(2,J)
        CALL COLOR(LCL)
      ENDIF
      IF(PSPEC(3,J).NE.LFL) THEN
        LFL = PSPEC(3,J)
        CALL FILTYP(LFL)
      ENDIF
      IF(PSPEC(4,J).GE.0) THEN
        CALL PLYGON(X, Y, NUM)
      ELSE
        CALL LINES(X(1), Y(1), X(2), Y(2), NUM-1)
      ENDIF
   10 CONTINUE                     ;GET NEXT POLYGON
      RETURN
```

```
      END
C
```

```
      SUBROUTINE VOLFRM (MODE1)
C
C     THIS ENTRY PUTS A FRAME AROUND THE REGION IF DESIRED.
C     FIRST PLOT + AT ALL OF THE VERTICES.
C
         COMMON/VOLCOM/XR,YR,A,TP,AP,DX,DY,DZ,X,Y,Z,SC1,FSEG,NTT
         LOGICAL A(1)
         INTEGER TP,AP
         DIMENSION XR(8),YR(8),DX(4),DY(4),DZ(4),X(4),Y(4),Z(4),SC1(4)
         DO 101 I = 1, 8
  101    CALL CHPLOT (SC1, XR(I), YR(I), '+¶.', 1,1,1)
C
C     NOW CHECK THE VARIOUS LINES.
C
         IF (MODE1 .LE. 1) GO TO 102
         CALL LINE (XR(1),YR(1), XR(2),YR(2))
         CALL LINE (XR(6),YR(6), XR(2),YR(2))
         CALL LINE (XR(6),YR(6), XR(5),YR(5))
         CALL LINE (XR(1),YR(1), XR(5),YR(5))
         CALL VOLUM8 (XR(4),YR(4), XR(3),YR(3), T, NT)
         CALL VOLUM8 (XR(4),YR(4), XR(8),YR(8), T, NT)
         CALL VOLUM8 (XR(7),YR(7), XR(8),YR(8), T, NT)
         CALL VOLUM8 (XR(7),YR(7), XR(3),YR(3), T, NT)
         CALL VOLUM8 (XR(4),YR(4), XR(2),YR(2), T, NT)
         CALL VOLUM8 (XR(1),YR(1), XR(3),YR(3), T, NT)
         CALL VOLUM8 (XR(6),YR(6), XR(8),YR(8), T, NT)
         CALL VOLUM8 (XR(7),YR(7), XR(5),YR(5), T, NT)
  102    CONTINUE
         RETURN
      END
C
```

```
      SUBROUTINE VOLUMO (Q, R, NX, NY, NZ)
C
         REAL     X(8), Y(8), Q(8), R(8)
         RNX = 1.0/FLOAT(NX-1)
         RNY = 1.0/FLOAT(NY-1)
         RNZ = 1.0/FLOAT(NZ-1)
         FNX = NX
         FNY = NY
         FNZ = NZ
         DO 1 I = 1, 8
         X(I) = Q(I)
    1    Y(I) = R(I)
         RETURN
C
C
      ENTRY VOLUM6 (XA,YA, XG,YG,ZG)
C
         DI1 = (FNX - XG)*RNX
         DJ1 = (FNY - YG)*RNY
         DK1 = (FNZ - ZG)*RNZ
         DIN = 1.0 - DI1
         DJN = 1.0 - DJ1
         DKN = 1.0 - DK1
         XA = ((X(1)*DI1+X(2)*DIN)*DJ1 + (X(3)*DI1+X(4)*DIN)*DJN)*DK1
    1       + ((X(5)*DI1+X(6)*DIN)*DJ1 + (X(7)*DI1+X(8)*DIN)*DJN)*DKN
         YA = ((Y(1)*DI1+Y(2)*DIN)*DJ1 + (Y(3)*DI1+Y(4)*DIN)*DJN)*DK1
    1       + ((Y(5)*DI1+Y(6)*DIN)*DJ1 + (Y(7)*DI1+Y(8)*DIN)*DJN)*DKN
         RETURN
      END
C
```

```
      SUBROUTINE VOLUM7 (CLEVEL, F00,F10,F11,F01, NSEG, X, Y)
C
         REAL     X(4), Y(4)
C
C     AT LEAST ONE CROSSING WILL BE FOUND. TEST THE LOWER SEGMENT FIRST.
C
         ISEG = 0
         IF (F00.NE.F10) DX = (CLEVEL-F00)/(F10-F00)
         IF (F00.EQ.F10) DX = 0.0
         IF (DX.GE.1.0 .OR. DX.LE.0.0) GO TO 2
C
C     RECORD THE LOWER CROSSING.
C
         ISEG = ISEG + 1
         X(ISEG) = DX
         Y(ISEG) = 0.0
C
C     TEST THE RIGHT SEGMENT.
C
    2    IF (F10.NE.F11) DY = (CLEVEL-F10)/(F11-F10)
         IF (F10.EQ.F11) DY = 0.0
         IF (DY.GE.1.0 .OR. DY.LE.0.0) GO TO 3
C
C     RECORD THE RIGHT SIDE CROSSING.
C
         ISEG = ISEG + 1
         X(ISEG) = 1.0
         Y(ISEG) = DY
C
C     TEST THE TOP SEGMENT.
C
    3    IF (F11.NE.F01) DX = (CLEVEL-F01)/(F11-F01)
         IF (F11.EQ.F01) DX = 1.0
         IF (DX.GE.1.0 .OR. DX.LE.0.0) GO TO 4
C
C     RECORD THE TOP CROSSING.
C
         ISEG = ISEG + 1
         X(ISEG) = DX
         Y(ISEG) = 1.0
C
C     TEST THE LEFT SIDE SEGMENT.
C
    4    IF (F01.NE.F00) DY = (CLEVEL-F00)/(F01 - F00)
         IF (F01.EQ.F00) DY = 1.0
         IF (DY.GE.1.0 .OR. DY.LE.0.0) GO TO 5
C
C     RECORD THE LEFT SIDE CROSSING.
C
         ISEG = ISEG + 1
         X(ISEG) = 0.0
         Y(ISEG) = DY
C
C     SAVE ANY LINE SEGMENTS FOUND
C
```

```
      5     NSEG = ISEG
            RETURN
          END
```

```
      SUBROUTINE VOLUM8 (XA, YA, XB, YB, T, NT)
C
         COMMON/VOLCOM/XR,YR,A,TP,AP,DX,DY,DZ,P,Q,R,SC1,FSEG,NTT
         DIMENSION XR(8),YR(8),DX(4),DY(4),DZ(4),P(4),Q(4),R(4),SC1(4)
         LOGICAL T(NT, NT)
         DATA    KINC / 1 /
         LOGICAL  SW1
C
         IF (SQRT((XB-XA)**2 + (YB-YA)**2) .LT. 1.9) RETURN
C
C     CHECK FOR HIDDEN PORTIONS OF THE LINE.
C
         RSEG=FLOAT(NTT)/1024.
         IAT = RSEG*XA
         IBT = RSEG*XB
         JAT = RSEG*YA
         JBT = RSEG*YB
         KI = ABS(XB-XA)
         KJ = ABS(YB-YA)
         IF (KI.GT.120 .OR. KJ.GT.120) GO TO 1
         IF (.NOT.T(IAT,JAT) .OR. .NOT.T(IBT,JBT)) GO TO 1
         CALL LINE (XA,YA, XB,YB)
         RETURN
C
C     AT LEAST PART OF THE LINE IS HIDDEN. FIND REASONABLE LIMITS.
C
    1    KMAX = (MAX0(KI,KJ)/KINC)*KINC + 1
         IF (KMAX .LT. 2) RETURN
C
C     NOW CHECK EACH SEGMENT ALONG THE LINE.
C
         SW1 = .FALSE.
         DO 2 K = 1, KMAX, KINC
         FA = FLOAT(K-1)/FLOAT(KMAX-1)
         X = XA + FA*(XB - XA)
         Y = YA + FA*(YB - YA)
         IX = RSEG*X
         IY = RSEG*Y
         IF (SW1) GO TO 3
C
C     UPDATE THE FIRST POINT IF SW1 IS .FALSE.
C
         IF (.NOT.T(IX,IY)) GO TO 3
         XST = X
         YST = Y
         SW1 = .TRUE.
C
C     SET THE END POINT.
C
    3    IF (T(IX,IY) .AND. K.NE.KMAX) GO TO 2
         IF (.NOT.SW1) GO TO 2
         FB = FLOAT(K-1-KINC)/FLOAT(KMAX-1)
         XND = XA + FB*(XB - XA)
         YND = YA + FB*(YB - YA)
         CALL LINE (XST,YST, XND,YND)
```

```
            SW1 = .FALSE.
     2      CONTINUE
            RETURN
        END
C
```

```
      SUBROUTINE VOLUME (MODE, F, NX,NY,NZ, CLEVE, NCL, T, NT)
C
C
C      THIS SET OF ROUTINES IS DESIGNED TO DRAW ON A DEVICE, SUCH AS
C      A PRINTER, A PLOTTER, THE SC4020 OR THE TEKTRONICS GRAPHICS
C      TERMINAL,  A TWO-DIMENSION PROJECTION OF A THREE DIMENSIONAL
C      ARRAY (FIGURE).  THE HIDDEN LINES ARE ELIMINATED TO SIMULATE
C      THE EFFECT OF "3D".
C
C      THE COLOR OF THE FIGURES (FOR THE SC4020 ONLY) MUST BE SET
C      BY THE USER IN THE CALLING PROGRAM USING THE APPROPRIATE CALLS
C      TO THE GRAPHICS PACKAGE.
C
C    . THE AXIS ARE LABELED AS FOLLOWS: X IS LEFT TO RIGHT
C                                       Y IS INTO THE PAGE
C                                       Z IS BOTTOM TO TOP
C
C      TO INITIALIZE THE PLOTTING PACKAGE, "VOLSET" MUST BE CALLED.
C      IF A FRAME SURROUNDING THE DRAWING IS WANTED, CALL "VOLFRM".
C      THE ROUTINE LOOPS OVER THE Y AXIS FROM FRONT TO BACK, PLOTTING
C      EACH SUCCESSIVE (X,Z) PLANE.  AT THE END OF THE (X,Z) CYCLE,
C      THE ARRAY "T" IS SET TO INDICATE WHAT WILL BE HIDDEN BY THE
C      CURRENT PLANE.
C
C    . CALLING SEQUENCE OR OPERATIONAL PROCEDURE:
C
C      CALL VOLSET(XP, YP, T, NT)
C      CALL VOLUME (MODE, F, NX, NY, NZ, CLEVE, NCL)
C      CALL VOLFRM(MODE1)
C
C    . ARGUMENTS (TYPE AND SIGNIFICANCE) AND/OR INITIAL CONDITIONS:
C
C      XP CONTAINS THE LOCATIONS OF THE X POSITION OF THE VERTICES (8)
C      YP CONTAINS THE CORRESPONDING Y VALUES (8)
C      T IS THE PLOTTING ARRAY (BOOLEAN) TO ELIMINATE THE HIDDEN LINES
C        WHICH IS NT X NT IN SIZE (THIS ARRAY TOGETHER WITH
C        NX,NY,NZ DETERMINE THE FINESSE OF THE PLOT).
C      MODE DETERMINES WHICH SURFACE IS PLOTTED (USED WITH CLEVE)
C       ABS(MODE)=1 WILL PLOT THE CONTOUR LEVEL "CLEVE" IN EACH PLANE
C                   AND THEN FIXES THE HIDDEN LINE MATRIX "T".
C       ABS(MODE)=2 SIMPLY FIXES THE HIDDEN LINE MATRIX "T" WITHOUT
C                   PLOTTING.  THIS IS USEFUL IN AVOIDING MULTIPLE
C                   PLOTS OF THE SAME FIXED FIGURE.
C       MODE<0      IS USED WHEN THE VALUE OF THE FUNCTION ON THE
C                   "OUTSIDE" IS LESS THAN "CLEVE".
C       MODE>0      IS USED WHEN THE VALUE OF THE FUNCTION ON THE
C                   "OUTSIDE" IS GREATER THAN "CLEVE".
C      F IS THE THREE-DIMENSIONAL FIGURE TO BE PLOTTED.
C      NX,NY,NZ ARE THE DIMENSIONS OF THIS ARRAY <F(NX,NY,NZ)>
C      CLEVE IS THE CONTOUR SURFACE LEVEL FOR PLOTTING.
C      NCL IS THE NUMBER OF PAIRS OF (MODE,CLEVE) TO BE PLOTTED.
C      MODE1 IS THE MODE OF FRAMING 0=> "+" AT THE EIGHT VERTICES;
C                                   1=> A BOX FORMED BY LINES.
C
```

```
C
      COMMON/VOLCOM/XR,YR,A,TP,AP,DX,DY,DZ,X,Y,Z,SC1,FSEG,NTT
      LOGICAL T(NT,NT)
      DIMENSION XR(8),YR(8),DX(4),DY(4),DZ(4),X(4),Y(4),Z(4),SC1(4)
      INTEGER  MODE(NCL)
      REAL     F(NX,NY,NZ), CLEVE(NCL)
      REAL XP(8), YP(8)
C
      CALL VOLUM0 (XR,YR, NX,NY,NZ)
C
C     THE OUTER LOOP IS OVER THE I,K SURFACES MOVING BACK.
C
      DO 1 J = 1, NY
      IF (J.EQ.1) GO TO 2
C
C     THEN LOOP OVER THE VARIOUS CONTOUR LEVELS.
C
      DO 20 LL = 1, NCL
      CLEVEL = CLEVE(LL)
      MODEL = MODE(LL)
      IF (IABS (MODEL) .EQ. 2) GO TO 20
C
C     FIRST COMPUTE THE FRONT-TO-BACK LINE SEGMENTS.
C
      DO 3 I = 1, NX
      DO 4 K = 2, NZ
      FMAX = AMAX1 (F(I,J-1,K-1), F(I,J,K-1), F(I,J,K), F(I,J-1,K))
      FMIN = AMIN1 (F(I,J-1,K-1), F(I,J,K-1), F(I,J,K), F(I,J-1,K))
      IF (FMAX.LT.CLEVEL .OR. FMIN.GT.CLEVEL) GO TO 4
      CALL VOLUM7 (CLEVEL, F(I,J-1,K-1), F(I,J,K-1),
     1       F(I,J,K), F(I,J-1,K), NSEG, DY, DZ)
      IF (NSEG .LE. 1) GO TO 4
      DO 5 L = 1, NSEG
      X(L) = FLOAT(I)
      Y(L) = FLOAT(J-1) + DY(L)
    5 Z(L) = FLOAT(K-1) + DZ(L)
      CALL VOLUM6 (XA,YA, X(1),Y(1),Z(1))
      CALL VOLUM6 (XB,YB, X(2),Y(2),Z(2))
      CALL VOLUM8 (XA,YA, XB,YB, T, NT)
      IF (NSEG.NE.4) GO TO 4
      CALL VOLUM6 (XA,YA, X(3),Y(3),Z(3))
      CALL VOLUM6 (XB,YB, X(4),Y(4),Z(4))
      CALL VOLUM8 (XA,YA, XB,YB, T, NT)
    4 CONTINUE
    3 CONTINUE
      DO 6 K = 1, NZ
      DO 7 I = 2, NX
      FMAX = AMAX1 (F(I-1,J-1,K), F(I,J-1,K), F(I,J,K), F(I-1,J,K))
      FMIN = AMIN1 (F(I-1,J-1,K), F(I,J-1,K), F(I,J,K), F(I-1,J,K))
      IF (FMAX.LT.CLEVEL .OR. FMIN.GT.CLEVEL) GO TO 7
      CALL VOLUM7 (CLEVEL, F(I-1,J-1,K), F(I,J-1,K),
     1       F(I,J,K), F(I-1,J,K), NSEG, DX, DY)
      IF (NSEG .LE. 1) GO TO 7
      DO 8 L = 1, NSEG
      X(L) = FLOAT(I-1) + DX(L)
```

```
            Y(L) = FLOAT(J-1) + DY(L)
      8     Z(L) = FLOAT(K)
            CALL VOLUM6 (XA,YA, X(1),Y(1),Z(1))
            CALL VOLUM6 (XB,YB, X(2),Y(2),Z(2))
            CALL VOLUM8 (XA,YA, XB,YB, T, NT)
            IF (NSEG.NE.4) GO TO 7
            CALL VOLUM6 (XA,YA, X(3),Y(3),Z(3))
            CALL VOLUM6 (XB,YB, X(4),Y(4),Z(4))
            CALL VOLUM8 (XA,YA, XB,YB, T, NT)
      7     CONTINUE
      6     CONTINUE
     20     CONTINUE
C
C     NOW PLOT THE X-Z SURFACE CONTOURS BEFORE CORRECTING THE HIDDEN
C     LINE ARRAY.
C
      2     DO 21 LL = 1, NCL
            CLEVEL = CLEVE(LL)
            MODEL = MODE(LL)
            IF (IABS (MODEL) .EQ. 2) GO TO 21
            DO 9 K = 2, NZ
            DO 10 I = 2, NX
            FMAX = AMAX1 (F(I-1,J,K-1), F(I,J,K-1), F(I,J,K), F(I-1,J,K))
            FMIN = AMIN1 (F(I-1,J,K-1), F(I,J,K-1), F(I,J,K), F(I-1,J,K))
            IF (FMAX.LT.CLEVEL .OR. FMIN.GT.CLEVEL) GO TO 10
            CALL VOLUM7 (CLEVEL, F(I-1,J,K-1), F(I,J,K-1),
      1         F(I,J,K), F(I-1,J,K), NSEG, DX, DZ)
            IF (NSEG .LE. 1) GO TO 10
            DO 11 L = 1, NSEG
            X(L) = FLOAT(I-1) + DX(L)
            Y(L) = FLOAT(J)
     11     Z(L) = FLOAT(K-1) + DZ(L)
            CALL VOLUM6 (XA,YA, X(1),Y(1),Z(1))
            CALL VOLUM6 (XB,YB, X(2),Y(2),Z(2))
            CALL VOLUM8 (XA,YA, XB,YB, T, NT)
            IF (NSEG.NE.4) GO TO 10
            CALL VOLUM6 (XA,YA, X(3),Y(3),Z(3))
            CALL VOLUM6 (XB,YB, X(4),Y(4),Z(4))
            CALL VOLUM8 (XA,YA, XB,YB, T, NT)
     10     CONTINUE
      9     CONTINUE
     21     CONTINUE
C
C     FILL THE HIDDEN LINE ARRAY AFTER PLOTTING IN THE NEW PLANE.
C
            DO 22 LL = 1, NCL
            CLEVEL = CLEVE(LL)
            MODEL = MODE(LL)
            DO 12 K = 2, NZ
            DO 13 I = 2, NX
            IF (MODEL.GT.0 .AND. AMIN1(F(I-1,J,K-1), F(I,J,K-1),
      1         F(I-1,J,K), F(I,J,K)).GT.CLEVEL) GO TO 13
            IF (MODEL.LT.0 .AND. AMAX1(F(I-1,J,K-1), F(I,J,K-1),
      1         F(I-1,J,K), F(I,J,K)).LT.CLEVEL) GO TO 13
C
```

```
C     THERE IS HIDDEN STUFF SOMEWHERE IN THE CELL. FIRST FIND THE LIMITS
C     OF THE CELL.
C
          CALL VOLUM6 (X00,Y00, FLOAT(I-1), FLOAT(J), FLOAT(K-1))
          CALL VOLUM6 (X10,Y10, FLOAT(I  ), FLOAT(J), FLOAT(K-1))
          CALL VOLUM6 (X11,Y11, FLOAT(I  ), FLOAT(J), FLOAT(K  ))
          CALL VOLUM6 (X01,Y01, FLOAT(I-1), FLOAT(J), FLOAT(K  ))
          DXMAX = AMAX1 (ABS(X10-X00),ABS(X11-X01))
          DYMAX = AMAX1 (ABS(Y01-Y00),ABS(Y11-Y10))
          NSEGX = (DXMAX + FSEG)/FSEG
          NSEGZ = (DYMAX + FSEG)/FSEG
          DII = 1.0/NSEGX
          DKK = 1.0/NSEGZ
          DO 14 II = 1, NSEGX
          XGRID = (FLOAT(II) - 0.5)*DII
          DO 15 KK = 1, NSEGZ
          ZGRID = (FLOAT(KK) - 0.5)*DKK
          CALL VOLUM6 (XA,YA,XGRID+FLOAT(I-1),FLOAT(J),ZGRID+FLOAT(K-1))
          IXA = XA/FSEG
          IYA = YA/FSEG
          IF (.NOT.T(IXA,IYA)) GO TO 15
C
C     FIND THE VALUE OF F AT THE POINT UNDER SCRUTINY.
C
          FVAL = (F(I-1,J,K-1)*(1.0-XGRID)+F(I,J,K-1)*XGRID)*(1.0-ZGRID)
     1         +  (F(I-1,J,K)*(1.0-XGRID) +F(I,J,K)*XGRID)*ZGRID
          IF (MODEL.GT.0.AND.FVAL.LT.CLEVEL) T(IXA,IYA) = .FALSE.
          IF (MODEL.LT.0.AND.FVAL.GT.CLEVEL) T(IXA,IYA) = .FALSE.
   15     CONTINUE
   14     CONTINUE
C
   13     CONTINUE
   12     CONTINUE
   22     CONTINUE
    1     CONTINUE
          RETURN
C
      ENTRY VOLSET (XP, YP, T, NT)
C
C         THIS FIRST ENTRY INITIALIZES THE PLOT GEOMETRY AND FILLS THE
C     HIDDEN LINE ARRAY T WITH THE INITIAL .TRUE. FOR TRANSPARENT. THE
C     ARRAY T IS NT X NT (OR 16K WORDS MAX) AND IS PASSED IN FROM THE
C     OUTSIDE. THIS CORRESPONDS TO 1024/NT RASTER UNITS ON THE SC4020.
C
C
          SC1(1) = 1.0
          SC1(2) = 0.0
          SC1(3) = 1.0
          SC1(4) = 0.0
          DO 100 I = 1, NT
          DO 100 J = 1, NT
  100     T(I,J) = .TRUE.
          FSEG = 1024.0/FLOAT(NT)
          DO 103 I = 1, 8
          XR(I) = XP(I)
```

```
  103     YR(I) = YP(I)
          NTT = NT
          RETURN
      END
```

```
           SUBROUTINE WDCOUNT(W,NC)
C
C      COUNT THE NUMBER OF CHARACTERS (LEGITIMATE) IN A STRING
C
           CHARACTER*1 W(*), E, P, B,  H, U, D, L, C, R, O, M
           DATA E/'¶'/,P/'.'/,B/'B'/,H/'H'/,U/'U'/,D/'D'/,L/'L'/
           DATA C/'C'/,R/'R'/,O/'O'/,M/'M'/
C
           IP = 0 ; NO CHARACTERS
           I = 1
  100      IF(W(I).EQ.E) THEN
             IF(W(I+1).EQ.H) THEN
               I = I + 2
             ELSE
               IF(W(I+1).EQ.U) THEN
               ELSE
                 IF(W(I+1).EQ.D) THEN
                 ELSE
                   IF(W(I+1).EQ.L.OR.W(I+1).EQ.R.OR.W(I+1).EQ.M) THEN
                   ELSE
                     IF(W(I+1).EQ.B) THEN
                     ELSE
                       IF(W(I+1).EQ.O) THEN
                       ELSE
                         IF(W(I+1).EQ.C) THEN
                           I = I + 2
                         ELSE
                           IF(W(I+1).EQ.E) THEN
                             GO TO 200
                           ELSE
                             IF(W(I+1).EQ.P) THEN
                               GO TO 300
                             ELSE
                               GO TO 101
                             ENDIF
                           ENDIF
                         ENDIF
                       ENDIF
                     ENDIF
                   ENDIF
                 ENDIF
               ENDIF
             ENDIF
           ELSE
             GO TO 200
           ENDIF
C
  101      I = I + 2
           IF(I.GT.132) GO TO 400 ; QUIT ANYHOW
           GO TO 100
C
C      COUNT A CHARACTER
C
  200      I = I + 1
           IP = IP + 1
```

```
            GO TO 100
      C
      C     TERMINATE
      C
        300 NC = IP
            RETURN
      C
      C     TERMINATE
      C
        400 NC = 0
            RETURN
            END
```

```
      SUBROUTINE WDDRAW(XSTART, YSTART, DX, DY, SX, SXY, SY, W)
C
C     XSTART REAL     - X COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
C                       CHARACTER TO BE DRAWN
C     YSTART REAL     - Y COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
C                       CHARACTER TO BE DRAWN
C     DX     REAL     - INCREMENT TO BE ADDED TO THE X COORDINATE FOR
C                       EACH CHARACTER DRAWN
C     DY     REAL     - INCREMENT TO BE ADDED TO THE Y COORDINATE FOR
C                       EACH CHARACTER DRAWN
C     SX     REAL     - X MATH SPACE SIZE FOR CHARACTERS
C     SXY    REAL     - SLANT MODIFIER FOR CHARACTERS
C     SY     REAL     - Y MATH SPACE SIZE FOR CHARACERS
C     W               - CHARACTER STRING TO BE DRAWN.
C
C    CONTROL CHARACTERS¶L - TEXT IS DRAWN TO THE LEFT OF XSTART,YSTART
C                      ¶M - TEXT IS CENTERED AT XSTART,YSTART
C                      ¶R - TEXT IS DRAWN TO THE RIGHT OF XSTART,YSTART
C
C                      ¶H - CHANGE CHARACTER SETS ... ¶H01, ¶H02,
C                           ... ¶H20, ¶H21 FOR CHARACTER SETS
C                           1, 2, ... 20, AND 21.
C
C                      ¶U - DRAW THE FOLLOWING CHARACTERS IN
C                           SUPERSCRIPT.
C                      ¶D - DRAW THE FOLLOWING CHARACTERS IN SUBSCRIPT.
C                      ¶O - RESET CHARACTER SIZE AND PLACEMENT FROM
C                           SUPERSCRIPT OR SUBSCRIPT TO ORIGINAL
C                           SIZE.
C
C                      ¶B - BACKSPACE OVER LAST CHARACTER DRAWN.
C                           WORKS ONLY FOR ONE CHARACTER.  MULTIPLE
C                           BACKSPACES WILL PRODUCE UNPREDICTABLE
C                           RESULTS.
C
C                      ¶C - CHANGE COLOR, ¶C00, ¶C01, ... TO CHANGE
C                           TO COLOR 0, 1, ...
C
C                      ¶. - END OF CHARACTER STRING.
C***********************************************************************
C    WDDRAW
C***********************************************************************
C
      INTEGER CHNUMB, SDFLT
      CHARACTER*1 W(1), E, P, B, CC(4), H, U, D, L, C
      CHARACTER*1 CENTER, R, O, M
      DATA E/'¶'/,P/'.'/,B/'B'/,H/'H'/,U/'U'/,D/'D'/,L/'L'/
      DATA C/'C'/,R/'R'/,O/'O'/,M/'M'/, SDFLT/4/, SE/4/
      REAL   X(132),Y(132),SCL(132),XLL(125),YLL(125),PEN(125)
      COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
      COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
      EQUIVALENCE  (CC,CHNUMB)
      INTEGER  SETL(132), SE, CH(132), ICL(132)
C
C
```

```
C     SET THE PLOTTING SPACE PARAMETERS
C
            CHSIZE(2) = SY * XYCOOD(3) * CHSIZE(6)
            CHSIZE(7) = CHSIZE(2)*(CHSIZE(3)+.5*(1.8-CHSIZE(6)))/XYCOOD(1)
            CHSIZE(8) = CHSIZE(2)*(CHSIZE(4)+.5*(1.8-CHSIZE(6)))/XYCOOD(3)
C
            XL = 0.0
            YL = 0.0
            IP = 0
            ICOL = 0
            XLAST = XL
            WIDTH = 0.0
            SC = 1.0
            IC = 1
            I = 1
            CENTER = 'R'
C
C     SCAN THE INPUT STRING FOR JUSTIFICATION AND END OF TEXT
C
  100       IF(W(I).EQ.E) THEN
              IF(W(I+1).EQ.H) THEN
                SE = CTOI(W(I+2),IC)*10 + CTOI(W(I+3),IC)
                IF (SE.LT.1.OR.SE.GT.24) SE = SDFLT
                I = I + 2
              ELSE
                IF(W(I+1).EQ.U) THEN
                  XL = XL + SX * 0.20 * SC
                  YL = YL + SY * 0.8 * SC
                  SC = SC * 0.6
                ELSE
                  IF(W(I+1).EQ.D) THEN
                    XL = XL + SX * 0.06 * SC
                    YL = YL - SY * 0.20 * SC
                    SC = SC * 0.6
                  ELSE
                    IF(W(I+1).EQ.L.OR.W(I+1).EQ.R.OR.W(I+1).EQ.M) THEN
                      CENTER = W(I+1)
                    ELSE
                      IF(W(I+1).EQ.B) THEN
                        XL = XL - WLAST
                      ELSE
                        IF(W(I+1).EQ.O) THEN
                          YL = 0.0
                          SC = 1.0
                        ELSE
                          IF(W(I+1).EQ.C) THEN
                            ICOL = CTOI(W(I+2),IC)*10 + CTOI(W(I+3),IC)
                            I = I + 2
                          ELSE
                            IF(W(I+1).EQ.E) THEN
                              GO TO 200
                            ELSE
                              IF(W(I+1).EQ.P) THEN
                                GO TO 300
                              ELSE
```

```
                                 I = I - 1
                                 GO TO 101
                               ENDIF
                             ENDIF
                           ENDIF
                         ENDIF
                       ENDIF
                     ENDIF
                   ENDIF
                 ENDIF
               ENDIF
             ELSE
               GO TO 200
             ENDIF
C
   101       I = I + 2
             IF(I.GT.132) GO TO 300 ; QUIT ANYHOW
             GO TO 100
C
C       WE HAVE A CHARACTER
C
   200       IF(SE.EQ.1) THEN
               CC(1) = W(I)
               CWMIN = 0.
               CWMAX = SX
             ELSE
               CHNUMB = IDCHAR(W(I))
               CALL HSETS (CHNUMB,XLL,YLL,PEN,NR,SE-1,IER)
               IF(IER.NE.0) THEN
                 IF(LUDIAG.GT.0) WRITE(LUDIAG,201) CHNUMB,SE,W(I)
   201           FORMAT(' CHARACTER ',Z10,' NOT FOUND IN SET ',I4,Z5)
                 GO TO 203
               ENDIF
               IF(NR.EQ.0) THEN
                 CWMIN  = 0.0
                 CWMAX = SX * 0.6
               ELSE
                 CWMIN = 1.0E+10
                 CWMAX = 0.
                 DO 202  J = 1, NR
                 XLL(J) = XLL(J) * 0.216 * SX
                 CWMIN = MIN (CWMIN, XLL(J))
   202           CWMAX = MAX (CWMAX, XLL(J))
               ENDIF
             ENDIF
C
C       STORE ITS PARAMETERS IN LOCAL SPACE
C
             XLAST = XL
             IP = IP + 1
             X(IP) = XL - CWMIN
             Y(IP) = YL
             ICL(IP) = ICOL
             SCL(IP) = SC
             CH(IP) = CHNUMB
```

```
          SETL(IP) = SE
          WLAST = (0.500*SC+0.5)*(CWMAX-CWMIN) + 0.20*SC*DX
          XL = XL + WLAST
          YL = YL + DY
  203     I = I + 1
          GO TO 100
C
C     DRAW IT - FIRST SET THE CORRECT CENTERING
C
  300     XOFF = X(1)
          WIDTH = XL - XOFF
          IF (CENTER.EQ.R) THEN
            DXX = 0.0
          ELSE IF (CENTER.EQ.M) THEN
            DXX = WIDTH / 2.
          ELSE IF (CENTER.EQ.L) THEN
            DXX = WIDTH
          END IF
          DO 301 I = 1, IP
  301     X(I) = X(I) - XOFF - DXX
C
C     DO ANY SHIFTING AND NECESSARY ROTATIONS
C
          COSX = COS(ANGLE)
          SINX = SIN(ANGLE)
          SXX = ABS(XYCOOD(1))
          SYY = ABS(XYCOOD(3))
          OSX = 1. / SXX
          OSY = 1. / SYY
          DO 304 I = 1, IP
          XI = X(I) * COSX * SXX - Y(I) * SINX * SYY
          YI = X(I) * SINX * SXX + Y(I) * COSX * SYY
          X(I) = XI * OSX + XSTART
  304     Y(I) = YI * OSY + YSTART
C
C     NOW DO THE DRAW
C
          IF(IP.LE.0) GO TO 303 ; NOTHING TO DO
          ICOL = 0
          DO 305 J = 1, IP
          IF(ICL(J).NE.0.AND.ICL(J).NE.ICOL) THEN
            CALL COLOR(ICL(J))
            ICOL = ICL(J)
          ENDIF
  305     CALL HHDRAW (X(J), Y(J), SX*SCL(J), SXY, SY*SCL(J),
     .    CH(J), SETL(J), IER)
  303     CHSIZE(2) = CHSIZE(1)
          ANGLE = 0.0
          RETURN
C
C     CHANGE THE DEFAULT CHARACTER SET - BYPASS THE ESCAPE SEQUENCE
C
          ENTRY CHRSET (ISET)
          IF(ISET.GT.0) SE = ISET
          IF(LUDIAG.GT.0) WRITE(LUDIAG,302) ISET
```

```
          RETURN
    302   FORMAT(' CHANGE THE DEFAULT CHARACTER SET TO #',I3)
          RETURN
          END
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 85-3235	2. Performing Organ. Report No.	3. Publication Date October 1985

4. TITLE AND SUBTITLE

A Device Independent Graphics Kernel

5. AUTHOR(S)

Walter W. Jones and Alicia B. Fadell

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
~~WASHINGTON, D.C. 20234~~ Gaithersburg, Maryland 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This paper describes an interface for programs which allows one to write graphics primitives to several devices without regard for the type of device. The most salient features are that it has low overhead, is transportable and can be expanded as the nature of the input/output devices changes. A conscious effort has been made to include all normal graphics primitives together with the most useful high level routines without compromising the use of special features of custom display units.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

device independence; display devices; graphics

13. AVAILABILITY

[X] Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
[X] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

241

15. Price