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
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

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

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Interface between application program and devices or processors</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Schematic of the transform from the application window to a device dependent world space</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>Schematic of the transform from the application window to a viewpoint on the viewing surface</td>
<td>7</td>
</tr>
<tr>
<td>4.</td>
<td>Filling pattern for &quot;FILTP&quot;</td>
<td>36</td>
</tr>
<tr>
<td>5.</td>
<td>Default color lookup table for Lexidata 8100</td>
<td>52</td>
</tr>
<tr>
<td>6.</td>
<td>Plot geometry used in &quot;SURFAC&quot;; this shows the naming convention used for the skirts and surfaces</td>
<td>53</td>
</tr>
<tr>
<td>7.</td>
<td>Overlapping spheriods with hidden lines removed</td>
<td>58</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Sample of a hardware interface protocol</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>General format and grouping of graphics routines</td>
<td>4</td>
</tr>
<tr>
<td>3.</td>
<td>Device coordinates</td>
<td>8</td>
</tr>
<tr>
<td>4.</td>
<td>Listing from BUILD - corresponding to data file shown in Table 3</td>
<td>10</td>
</tr>
<tr>
<td>5.</td>
<td>Coordinate limits of the hardware</td>
<td>10</td>
</tr>
</tbody>
</table>
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 series1. 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 identification does not imply recommendation or indorsement by the National Bureau of Standards, nor does it imply that the equipment is necessarily the best available for the purpose.
burdensome since applications may require 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.

![Diagram](image)

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.
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 : graphics subroutines
      RETURN

C
  PRINTER SECTION
  2 : graphics subroutines
      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 appendixes.

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**
   - **DEVICE** (n)
   - **NEWFRM**
   - **HDCOPY**
   - **FRAME**
   - **ERASE**
   - **ENDFRM**

2. **Viewing**
   - **SCALNG** (X₁, Y₁, X₂, Y₂, X₁H, Y₁H, X₂H, Y₂H, X₁S, Y₁S, X₂S, Y₂S)
   - **DEFINE** (X₁, Y₁, X₂, Y₂)

3. **Output Primitives**
   - **LINE** (X₁, Y₁, X₂, Y₂)
   - **LINES** (X₁, Y₁, X₂, Y₂, N)
   - **LNPlot** (X, Y, I₁, I₂, I₃)
   - **PLYGON** (X, Y, N)
   - **BOXPLT** (X₁, Y₁, X₂, Y₂)
   - **CIRCLE** (X, Y, R)
   - **SURFAC** (Z, NX, NY, MODE)
   - **CONTUR** (F, TEST, NX, NY, FL)
   - **VOLUME** (MODE, F, NX, NY, NZ, CLEVEL, NCL, T, NT)
   - **SYMBOL** (X, Y, CHAR)
   - **CHPLOT** (X, Y, CHAR, I₁, I₂, I₃)
   - **HHDRAW** (X, Y, SX, SXY, ICHAR, NSET, NSET, IERR)
   - **WDDRAW** (X, Y, DX, DY, SX, SXY, SY, CHARS)

**Function**
- initialize a graphics device
- clear the screen and initialize a new frame
- generate a hard copy
- write out the buffer
- clear the screen (also done in "NEWFRM")
- close the graphics device

**Function**
- define the window and viewport
- define the window (viewport defaults to entire viewing surface)

**Function**
- draw a line between 2 points
- draw a line between points in an array of length n
- draw lines between selected points of an array
- draw a closed polygon
- draw a rectangle
- draw a circle
- performs surface plotting
- performs contour plotting
- performs volume plotting (3 dimensional contouring)
- draws a specified hardware character at a given point
- draws a specified hardware character at selected points in an array
- draws a particular character of the specified character set
- draws a character string
LABEL (CHARS, X1, Y1, X2, Y2, ANGLE)
ALABEL (CHARS, X1, X2, Y1, Y2, ANGLE)
FNUMBR (X, Y, DX, DY, SX, SXY, SY, XNUMBR, WIDTH, DIGITS)
ENUMBR (XNUMBR, X1, Y1, X2, Y2)
GRAFIT (NPLT, X1, X2, X1R, X2R, XX1, XX2, Y1, Y2, Y1R, Y2R, YY1, YY2, XTIT, NDVX, YTTIT, NDVY)
PLYPLT (F, IS)

MAPIN (FW, V, NV, E, ES, NE, P, PS, NP)
MAPOUT (W, V, NV, E, ES, NE, P, PS, NP)
VIEWTR (X, W, V, NV, E, ES, NE, P, PS, NP)

4. Attributes
COLOR (N)
LINWID (N)
FILTYP (N)
CHRSIZ (CHFRZ, GCHFRZ)
CHRSET (N)

5. Auxiliary
IOWAIT (N)
SETDEV (N1, N2)
DELAY (N)
SETLUT

6. Input
DEVINP (I, status, X,Y,Z)

draws a character string
draws a character string with aspect ratio 4/3
draws a real number in FORTRAN F-type format
draws a real number in exponential (E) format (E - type)
sets up a graph (x and y axis) for plotting data
read in a "BUILD" formatted structure file, F and display according to IS
read a "BUILD" file and return the vertices, edges and polygons. Files are appended after the initial call.
display the specified edges as given
display the specified edges and polygons using the transform matrix X.

Function
defines the color for drawing
defines the line width
defines the appearance of the interior of polygons and circles
sets the size of the hardware characters
changes the default character set

Function
puts a pause in the program
changes logical units
delays for hard copy units
sets up the color look-up table

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:

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

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

\[
\begin{align*}
XS &= X \times \text{XYCOORD}(1) + \text{XYCOORD}(2) \\
YS &= Y \times \text{XYCOORD}(3) + \text{XYCOORD}(4)
\end{align*}
\]

Any point within the user's window will be converted into valid world coordinates (i.e., X1S < XS < X2S and Y1S < YS < Y2S) 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](image-url)
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 transformations. 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

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>WORLD</th>
<th>NORMALIZED</th>
<th>PHYSICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>Lexidata</td>
<td>128000</td>
<td>102300</td>
<td>32700</td>
</tr>
<tr>
<td>Tek 40XX</td>
<td>1280</td>
<td>1024</td>
<td>1029</td>
</tr>
<tr>
<td>CALCOMP</td>
<td>-</td>
<td>-</td>
<td>1029</td>
</tr>
<tr>
<td>Printer</td>
<td>1279</td>
<td>1023</td>
<td>1279</td>
</tr>
</tbody>
</table>

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 ith 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 5th 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

\[
\text{EDGE (i,j) or POLY(i,j)},
\]

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

\[
\text{EDGE (1,4) = 6 and EDGE (2,4) = 10}
\]

Polygons are stored in a similar manner. \(\text{POLY(i,j)}\) contains the vertex number of the \(i\)th vertex of the \(j\)th polygon. Polygons have no more than \(\text{NPVERT}\) (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 \(\text{ESPEC (i,j)}\) or \(\text{PSPEC (i,j)}\) for edges and polygons, respectively. Here the \(i\) specifies the attribute and \(j\) indicates the element number.

\[\begin{array}{ccc}
\text{1} & \text{ESPEC (i,j)} & \text{1} \\
\text{1} & \text{line width} & \text{1} \\
\text{2} & \text{color} & \\
\text{3} & \text{line attribute} & \text{3} \\
\text{4} & <\text{unused}> & \text{4} \\
\end{array}\]

\[\begin{array}{ccc}
\text{1} & \text{PSPEC (i,j)} & \\
\text{1} & \text{line width} & \text{2} \\
\text{2} & \text{color} & \\
\text{3} & \text{fill type} & \\
\text{4} & \text{polygon (>0) or polyline (<0)} & \\
\end{array}\]

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

<table>
<thead>
<tr>
<th>BUILD</th>
<th>0.00000E+00</th>
<th>0.00000E+00</th>
<th>1.28000E+03</th>
<th>1.02400E+03</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.00000E+09</td>
<td>-1.00000E+09</td>
<td>-1.00000E+09</td>
<td>1.00000E+09</td>
<td>1.00000E+09</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.55608E+02</td>
<td>7.37531E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4.03576E+02</td>
<td>9.39818E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.00956E+02</td>
<td>7.45070E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.60015E+02</td>
<td>6.44554E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4.36470E+02</td>
<td>3.88245E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9.64768E+02</td>
<td>6.44554E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.30060E+02</td>
<td>3.68136E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.30070E+02</td>
<td>5.88014E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8.03136E+02</td>
<td>5.88014E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8.03136E+02</td>
<td>4.63627E+02</td>
<td>0.00000E+00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>WORLD</th>
<th>WINDOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT, BOTTOM, FRONT:</td>
<td>-1.0E+09 -1.0E+09 -1.0E+09 LEFT BOTTOM:</td>
</tr>
<tr>
<td>RIGHT, TOP, BACK:</td>
<td>1.0E+09 1.0E+09 1.0E+09 RIGHT, TOP:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VERTEX(X,Y,Z)</th>
<th>SPECs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 155.61</td>
<td>737.53 0.00000</td>
</tr>
<tr>
<td>2) 403.58</td>
<td>939.82 0.00000</td>
</tr>
<tr>
<td>3) 500.96</td>
<td>745.07 0.00000</td>
</tr>
<tr>
<td>4) 260.02</td>
<td>644.55 0.00000</td>
</tr>
<tr>
<td>5) 964.77</td>
<td>644.55 0.00000</td>
</tr>
<tr>
<td>6) 330.07</td>
<td>368.14 0.00000</td>
</tr>
<tr>
<td>7) 530.07</td>
<td>588.01 0.00000</td>
</tr>
<tr>
<td>8) 803.14</td>
<td>588.01 0.00000</td>
</tr>
<tr>
<td>9) 803.14</td>
<td>463.63 0.00000</td>
</tr>
</tbody>
</table>

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

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

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 accommodate 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 x 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

\[
\text{MOVE} \begin{cases} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \end{cases} \quad (\text{default} = \text{VERTEX})
\]

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 vertices when moving edges or polygons. In addition, duplicates of elements can be created with

\[
\text{DUPLICATE} \begin{cases} \text{EDGE} \\ \text{POLYGON} \end{cases} \quad (\text{default} = \text{VERTEX})
\]

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

\[
\text{TRANSLATE}
\]

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

\[
\text{ROTATE}
\]

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

\[
\text{SCALE}
\]

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

\[
\text{LIST} \begin{cases} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \\ \text{MATRIX} \end{cases} \quad \text{(default} = \text{VERTEX)}
\]

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

\[
\text{PRINT} \begin{cases} \text{VERTEX} \\ \text{EDGE} \\ \text{POLYGON} \\ \text{ALL} \\ \text{MATRIX} \end{cases} \quad \text{(default} = \text{ALL)}
\]

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

\[
\text{DEVICE} \begin{cases} \text{CALCOMP} \\ \text{PRINTER} \\ \text{LEXIDATA} \\ \text{CONSOLE} \end{cases} \quad \text{(default} = \text{LEXIDATA)}
\]

The specified device becomes the new default for graphical display.

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

\[
\text{COPY} \begin{cases} \text{ON} \\ \text{OFF} \end{cases} \quad \text{(default} = \text{OFF)}
\]

After COPY ON has been entered, a specified number of hard copies of each subsequent graphics display 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

\[
\text{DISPLAY} \begin{cases} 
\text{VERTEX} \\
\text{EDGE} \\
\text{POLYGON} \\
\text{ALL} 
\end{cases} 
\] (default = ALL)

If the output 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 model's edges or polygons, respectively without numbering them. All edges and polygons are shown with DISPLAY ALL. A screen can be cleared with

\text{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

\text{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

\text{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

\text{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 responses 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
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
```

cmdand 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
```
(displayed = 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.
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:

\[
\text{COLOR} = \text{MOD}(N-1, \text{NUMCOLOR}) + 1 \\
\text{COLOR} = \text{MAX}(\text{COLOR}, 1)
\]
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 counterclockwise 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 - coordinate 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.
Purpose: to draw a real decimal number expressed with powers of 10 such that the mantissa is between 1 and 10, i.e., $y \times 10^N$ 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 ENUMBR 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.
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.
FILTP

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

Usage: Call FILTP(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.
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

<table>
<thead>
<tr>
<th>Device</th>
<th>Associated Hard Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calcomp</td>
<td>-</td>
</tr>
<tr>
<td>Printer</td>
<td>-</td>
</tr>
<tr>
<td>Lexidata</td>
<td>Camera</td>
</tr>
<tr>
<td>Tektronix</td>
<td>Activates screen copy unit</td>
</tr>
</tbody>
</table>

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)
ICCHAR - 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.)
LNPLT

Purpose: to draw lines between selected points of an array

Usage: Call LNPLT (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.
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 (XI, 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:

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

*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 parallelepiped 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

1 ≤ X,Y ≤ 1024
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 tessellation is used to find the contour crossing points. The technique 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:
\[ F_1 = \frac{10}{(X-8)^2 + (Y-10.5)^2 + (Z-10.5)^2} \]
\[ F_2 = \frac{20}{(X-13)^2 + (Y-10.5)^2 + (Z-10.5)^2} \]
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

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

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

!"#$%&'()*+,-./0123456789:
:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

[ ]\}{
АБВГДЕЖЗИЙ
КЛМНОПРСТУ
ФХЦЧШЩЪЫЬЭ
ЮЯабвгдежз
ийклмнопрс
tуфхцчшщъы
ьэюя
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'

VERTEX NUMBERS:

Two integers, separated by a comma or space, are 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 degrees) 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:

\[ \begin{align*}
n \\
n-m \\
n- \\
n \text{ BY } X,Y,Z \\
n-m \text{ BY } X,Y,Z \\
n- \text{ BY } X,Y,Z \\
\end{align*} \]

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. \( n- \) represents 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 beginning 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
  \{ PRINTER
  \{ LEXIDATA
  \{ CONSOLE
```

(default = LEXIDATA)

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:

\[ \begin{align*}
    n \\
    n-m \\
    n- \\
    n \text{ BY } X,Y,Z \\
    n-m \text{ BY } X,Y,Z \\
    n- \text{ BY } X,Y,Z
\end{align*} \]

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 \(<\text{RETURN}>\) 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 \(<\text{RETURN}>\) or entering QUIT.

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

\[ \begin{align*}
    \text{e.g.} & & \text{long form} & & \text{short form} \\
    10,8,0 & & 10,8 \\
    0,7,0 & & ,7 \\
    0,0,6 & & ,,6 \\
    1,0,3 & & 1,,3
\end{align*} \]

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

18. PRINT

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

18.1 PRINT VERTEX

All of the model's edges are listed.

18.2 PRINT EDGE

All of the polygons are printed.

18.3 PRINT POLYGON

The transformation matrix is printed.

18.4 PRINT MATRIX

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

18.5 PRINT ALL

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.

<table>
<thead>
<tr>
<th>long form</th>
<th>short form</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,8,0</td>
<td>10,8</td>
</tr>
<tr>
<td>0,7,0</td>
<td>,7</td>
</tr>
<tr>
<td>0,0,6</td>
<td>,,6</td>
</tr>
<tr>
<td>1,0,3</td>
<td>1,,3</td>
</tr>
</tbody>
</table>

#### 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

\[
\text{SET}\{\text{KEYBOARD}}\} \quad (\text{default} = \text{KEYBOARD})
\]

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.

\[
\begin{array}{c|c|c}
\text{e.g.} & \text{long form} & \text{short form} \\
10,8,0 & 10,8 & \\
0,7,0 & ,7 & \\
0,0,6 & ,,6 & \\
1,0,3 & 1,,3 & \\
\end{array}
\]

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 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 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

\[ x, y, z, \text{IDV} = \]

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 specify 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.
<table>
<thead>
<tr>
<th>Program/Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM G2TEST</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE ANGTST</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE CIRC</td>
<td>4</td>
</tr>
<tr>
<td>SUBROUTINE CUBE</td>
<td>5</td>
</tr>
<tr>
<td>SUBROUTINE MM</td>
<td>7</td>
</tr>
<tr>
<td>SUBROUTINE MM3</td>
<td>8</td>
</tr>
<tr>
<td>SUBROUTINE SETMAT</td>
<td>9</td>
</tr>
<tr>
<td>SUBROUTINE TRANS</td>
<td>10</td>
</tr>
<tr>
<td>MAIN PROGRAM</td>
<td>11</td>
</tr>
<tr>
<td>SUBROUTINE GRAFIT1</td>
<td>12</td>
</tr>
<tr>
<td>SUBROUTINE GRAFIT2</td>
<td>13</td>
</tr>
<tr>
<td>SUBROUTINE RDCNL</td>
<td>15</td>
</tr>
<tr>
<td>SUBROUTINE GRAFIT3</td>
<td>16</td>
</tr>
<tr>
<td>SUBROUTINE GRAFIT4</td>
<td>18</td>
</tr>
<tr>
<td>SUBROUTINE PLT</td>
<td>19</td>
</tr>
<tr>
<td>SUBROUTINE PHASE</td>
<td>20</td>
</tr>
<tr>
<td>SUBROUTINE RUMTST</td>
<td>21</td>
</tr>
<tr>
<td>SUBROUTINE VENTV</td>
<td>22</td>
</tr>
<tr>
<td>SUBROUTINE LAYER</td>
<td>23</td>
</tr>
<tr>
<td>SUBROUTINE FIRE</td>
<td>24</td>
</tr>
<tr>
<td>SUBROUTINE PLUME</td>
<td>25</td>
</tr>
<tr>
<td>SUBROUTINE ROOM</td>
<td>26</td>
</tr>
<tr>
<td>SUBROUTINE SADDLE</td>
<td>27</td>
</tr>
<tr>
<td>SUBROUTINE SETCOL</td>
<td>29</td>
</tr>
<tr>
<td>SUBROUTINE SYMTST</td>
<td>30</td>
</tr>
<tr>
<td>SUBROUTINE VOLTST</td>
<td>31</td>
</tr>
<tr>
<td>SUBROUTINE FILNGT</td>
<td>33</td>
</tr>
</tbody>
</table>
PROGRAM G2TEST

INTEGER PB(6)

C  TEST EACH FUNCTION OF THE DEVICE INDEPENDENT PKAGE

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/HERSEY 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  GET THE DEVICE

CALL SYSIO(PB,33,5,'DEVICE=',7,0)
READ(5,2) ID
FORMAT(I)
CALL DEVICE(ID)
GO TO (201,202,203,204,205,206,207,208,209,210,211,212,
     213),I

STOP 'NO SELECTION'

CALL ANGIST
CALL CIRC(ID)
CALL CUBE
CALL GRAFIT1(ID)
CALL GRAFIT2(ID)
CALL GRAFIT3(ID)
CALL GRAFIT4(ID)
CALL PHASE
CALL RUMTST
CALL SADDLE(ID)
CALL SYMTST
CALL VOLIST(ID)
CALL FILNGT(ID)
FORMAT(IX,A50)
FORMAT(I)
STOP
END
SUBROUTINE ANGTST

A SIMPLE TEST OF THE CHARACTER STUFF

CHARACTER*1 TITLE(80)
INTEGER PB(6)
CHARACTER*80 STRING

FIRST A SIMPLE DEMONSTRATION

CALL GRISET (0.,0.,100.,100.)
CALL NEWFRM
CALL WDDRAW (50.,90.,6.,0.,8.,0.,8.,'1LLC02EFT1.' )
CALL WDDRAW (50.,75.,6.,0.,8.,0.,8.,'1M01C1CO2ENTERED1.' )
CALL WDDRAW (50.,60.,6.,0.,8.,0.,8.,'1R01R1CO2IGHT1.' )
STRING='1M01C1C0221C0331C0441C0551C0661C0771C0881C1391C1401.'

CALL WDDRAW (50.,45.,6.,0.,8.,0.,8.,STRING)
STRING='1M01ALS01U1C05SUP£RSCRIPTS101CO1AND1C051DSUBSCRIPTS
1.'

CALL WDDRAW (50.,30.,6.,0.,4.,0.,5.,STRING)
STRING = '1MABC1U1.2341.'
CALL LABEL(STRING,60.,10.,90.,20.,50)
STRING='1M01H04AND1C03YOU1C071H18CAN1H04CHANGE1CHARACTER.
1C031HISSETS1.'

CALL WDDRAW (50.,20.,6.,0.,4.,0.,5.,STRING)
STRING = '1C021H04S1US1US1US1.'
CALL WDDRAW (10.,10.,6.,0.,4.,0.,5.,STRING)
STRING = '1C031D1DD1DD1DD1.'

CALL WDDRAW(20.,10.,6.,0.,4.,0.,5., STRING)
STRING = '1C08NOWTRYONEYOURSELF1.'
CALL WDDRAW(10.,1.,6.,0.,4.,0.,5.,STRING)

CALL FRAME
PAUSE

THE RESTART POSITION - SCART WITH THE ANGLE OF THE TEXT

CALL SYSIO(PB,33,5,'ANGLE=',6,0)
READ(5,1) ANGLE
FORMAT(2F)
IF(ANGLE.LT.0.0) GO TO 5
THETA = ANGLE/57.295
DO 7 I = 1, 80
7 TITLE(I) = ' '  

ERASE THE SCREEN AND PREPARE TO DRAW

CALL NEWFRM
CALL SYSIO(PB,33,5,'TEXT=',5,0)
READ(5,3) TITLE
FORMAT(80A1)
DO 11 I = 1, 80
II = 81 - I
11 IF(TITLE(II).NE.' ') GO TO 12
CONTINUE
SUBROUTINE ANGSTST

106       GO TO 4
107      12  TITLE(II+1) = 'I'
108         TITLE(II+2) = '.
109     C  CALL SYSIO(PB,33,5,'LINWID=',7,0)
110     C  READ(5,2) LW
111         CALL DEFINE (0., 0., 13., 13.)
112     C  CALL COLOR(4)
113     C  CALL LINWID(LW)
114         CALL BOXPLT (0.5, 0.5, 12.5, 12.5)
115         CALL COLOR(5)
116         CALL WDCOUNT(TITLE,NT)
117         IF(NT.EQ.0) GO TO 4
118     C  DRAW THE LABEL
119     C
120     C
121       YB = 5.0
122       XPOS = 5.0
123       XRIGHT = XPOS + MIN(FLOAT(NT),11.)
124       YTOP = YB + 1.
125         CALL LABEL(TITLE, XPOS, YB, XRIGHT, YTOP, THETA)
126     C  CALL FRAME
127     C
128     C  CLOSE THE PLOTTING DEVICE AND ERASE NON-LOCAL SCREENS
129     C
130       CALL ENDFRM
131  5       CALL ENDFRM
132     C
133       STOP
134     C
135       END

-108-
SUBROUTINE CIRC

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 FILTPY(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), IPEDEG(24), NVERT, NEDGS, NSURF, F, SCALE(4)

INTEGER IV(2)/2, 0/

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
3000 CALL TRANS(T)
CALL SETMAT(QM,1,1,0,0,0,0)
SUBROUTINE CUBE

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
XCl = SIN(ANGLE8)
ZCl = COS(ANGLE8) + 1.0
YC1 = FLOAT(I)/360.
QM(1,4) = -(QM(1,1)*XCl + QM(1,2)*YC + QM(1,3)*ZCl) + XCl
QM(2,4) = -(QM(2,1)*XCl + QM(2,2)*YC + QM(2,3)*ZCl) + YC1
QM(3,4) = -(QM(3,1)*XCl + QM(3,2)*YC + QM(3,3)*ZCl) + ZCl
XC = XCl
YC = YC1
ZC = ZCl
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
D(I,J) = D(I,J) + B(I,K) * C(K,J)
CONTINUE
CONTINUE
DO 20 I = 1, 4
DO 20 J = 1, 4
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
D(I) = D(I) + B(I,J)*C(J)
CONTINUE
DO 20 I = 1, 3
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
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)
COMMONTQNORM(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)
DO10 I = 1, NEDGS
10 IEDG(I) = 0
DO20 I = 1, NVERT
CALLMM3(RVERT(1,I),R,VERT(1,I),1)
CONTINUE
IPTS = 1
DO100 I = 1, NSURF
K = ISURF(I)
IPTF = IPTS + K - 1
CALLMM3(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
DO30 J = IPTS, IPTF
JJ = IPEDG(J)
IPTS = IPTF + 1
DO200 I = 1, NVERT
ZPF = RVERT(3,I) + F
XP(I) = F*RVERT(1,I)/ZPF
YP(I) = F*RVERT(2,I)/ZPF
CONTINUE
CALLERASE
DO300 I = 1, NEDGS
IF(IEDG(I).EQ.0) GO TO 300
I1 = IEDGV1(I)
I2 = IEDGV2(I)
CALLLINE(XP(I1),YP(I1),XP(I2),YP(I2))
CONTINUE
CALLFRAME
CALLHDCOPY
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)
348  C
349  DIMENSION X(100), Y(100)
350  CHARACTER*31 YAXIS/* IGNITION TIME (MINUTES)$/'
351  C
352  CALL NEWFRM
353  C
354  CALL SETDEV(0,5)
355  CALL COLOR(2)
356  CALL GRISET (0., -50., 1023., 1023.)
357  PIP = 3.1415
358  PIM = - PIP
359  CALL COLOR(3)
360  CALL GRILAB(1,PIM,-2.,PIP,+2.)
361  CALL GRAFIT (1,PIM,PIP, 200., 900., PIM, PIP, -2.0, +2.0,
362  100., 800., -2.0, +2.0, 'X AXIS$$', 3, YAXIS, 5)
363  CALL COLOR(4)
364  CALL GRILAB(2,PIM,-2.,PIP,+2.)
365  CALL GRAFIT (2,PIM,PIP, 450., 650., PIM, PIP, -2.0, +2.0,
366  200., 500., -2.0, +2.0, 'X AXIS$$', 3, 'Y AXIS$$', 5)
367  XOFF = (PIP - PIM) / 100.
368  DO 2 I = 1, 100
369  XI = PIM + FLOAT(I-1) * XOFF
370  YI = COS(XI)
371    2 XI = XI
372    2 Y(I) = YI
373  CALL COLOR(1)
374  CALL PLOTLN (1, X, Y, 100)
375  CALL PLOTLN (2, X, Y, 100)
376  CALL FRAME
377  IF(ID.EQ.3) PAUSE
378  CALL ENDFRM
379  STOP
380  END
SUBROUTINE GRAFIT2(ID)

DIMENSION X(100),Y(100)

READ IN SOME DATA (X VS Y)

LIN = 3
OPEN(UNIT=1,FILE='G1TEST2.DAT')
CALL RDCNL(Y,Y1,YI,YI)
CALL RDCNL(X,X1,XI,XI)
IZ = MINO(XI, YI)
I3 = IZ/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
SCALE - WE NOW KNOW HOW BIG TO MAKE THE AXES

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))
YMIN = AMIN1(YMIN, 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-I+1))
5 YMAXC = AMAX1(YMAXC, Y(I3-I+1))
NOW ROUND OFF THE AXES TO APPROPRIATE WHOLE VALUES

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
SUBROUTINE GRAFIT2

435  XMAX = IFIX(XMAX+DELTAX/2.)
436  YMAX = IFIX((YMAX+DELTAY)/100.)*100.
437  CALL NEWFRM
438  CALL LINWID(LIN)
439  CALL COLOR(1)
440  CALL GRISET(XMIN, YMIN, XMAX, YMAX)
441  CALL GRILAB(1,XMIN,YMIN,XMAX,YMAX)
442  CALL GRAFIT(1,XMIN,XMAX,YMINA,XMAXA,XMIN,YMIN,YMAX,
443     YMINA,YMAXA,YMIN,YMAX,' TIME(MINUTES)\1.',14,
444     ' TEMPERATURE(K)\1. ',12)
445  CALL GRILAB(2,X(I3),YMINC,X(2*I3),YMAXC)
446  CALL GRAFIT(2,X(I3),X(2*I3),XMINB,XMAXB,X(I3),X(2*I3),
447     YMINC,YMAXC,YMINB,YMAXB,YMINC,YMAXC,
448     ' TIME(MINUTES)\1. ', 5, ' FLUE TEMPERATURE(K)\1. ', 5)
449  CALL COLOR(2)
450  CALL LINWID(4)
451  CALL PLOTLN(1,X,Y,I3)
452  CALL COLOR(3)
453  CALL PLOTLN(1,X(I3-1),Y(I3-1),I3)
454  CALL PLOTLN(2,X(I3),Y(I3),I3)
455  CALL COLOR(2)
456  CALL PLOTLN(1,X(2*(I3-1)),Y(2*(I3-1)),IZ-2*(I3-1))
457  CALL FRAME
458  IF(ID.EQ.3) PAUSE
459  CALL ENDFRM
460  STOP
461  END
SUBROUTINE RDCNL (REED,ICNL,MAXR,IEOF)

DIMENSION REED(IOO)
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

FORMAT (2I6,T78,I3)
FORMAT (7E11.5)
END
SUBROUTINE GRAFIT3(ID)

DIMENSION X(100), Y(100)
CHARACTER*31 YAXIS/'TIME SINCE IGNITION1. MINUTES)'
CHARACTER IC/'Z'/

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)
PIM = 3.1415
PIP = -PI
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 AXIS1.', 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 AXIS1.', 3, 'Y AXIS1.', 5)
XOFF = (PIP - PIM) / 10.
DO 2 I = 1, 10
XI = PIM + FLOAT(I-1) * XOFF
YI = COS(XI)
2 X(I) = XI
Y(I) = YI
CALL COLOR(1)
CALL PLOTLN (1, X, Y, 10)
CALL CHRSIZ(0., .05)
CALL PLOTCH (1, X, Y, 10, 'X'
CALL PLOTLN (2, X, Y, 10)
CALL PLOTCH (2, X, Y, 10, IC)
CALL FRAME
IF(ID.EQ.3) PAUSE
CONTINUE
CALL ENDFRM
STOP
END
SUBROUTINE GRAFIT4

DIMENSION Q(550), VS(550), X(500), Y(500), TM(500), SQT(500)
CHARACTER*16 FILE
CHARACTER*40 TITLE/GRAFIT(4)/

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
CONTINUE

TYPE *, TITLE
NP=N2
CALL PLT(NP,X,Y)
IF(ID.EQ.1) GO TO 50
PAUSE
CONTINUE
50 CALL ENDFRM
STOP
END
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/cm²U210)'
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.,0.,900.,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./

DO 1 I = 1,30
DO 1 J = I, 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
4 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)

CALL LINWID(1)

SET UP A REASONABLE CONTOURING INTERVAL

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
3 CALL COLOR(I)

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)

CONTINUE

PAUSE

CALL ENDFRM

STOP

END
SUBROUTINE VENTV

COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ

REAL BOTM, TOP, SIDE

C

DRAW VERTICAL LINES IN BLACK AND HORIZONTAL LINES IN WHITE

Y1 = BOTM
Y2 = TOP
DO 2 I = 1, 2
X1 = SIDE + (I-1) * WALLSZ * THICK
X2 = X1
2 CALL LINE (XI, 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)

DRAW SQUIGGLE FROM LEFT TO RIGHT TO REPRESENT THE LAYER (HOT) ABOVE THE PLUME

INTEGER DIVCNT

REAL LEFT, RIGHT, BOTTOM, TOP, X(4), Y(4)

NOW DRAW THE SQUIGGLE

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

COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ

C PLOT THE FIRE SOURCE

CALL COLOR(3)
CALL VBXPLT (CENTER-.5*WIDTH, 0.0, CENTER+.5*WIDTH,HEIGHT)

C START AT HEIGHT [HP=(1+EPS)*HEIGHT] AND TRACE THE PARABOLA

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

TO PUT THE PLUME IN, NORMALLY ABOVE THE FIRE

THIS ROUTINE ASSUMES A POINT SOURCE PLUME AT A
VIRTUAL DISTANCE BELOW THE FIRE SUCH THAT THE
PLUME SUBTENDS AN ANGLE OF 11 DEGREES AT THE FIRE
SOURCE.

REAL CENTER, WIDTH, PBOTM, PTOP, FSOURC
COMMON/ROOMSZ/RML,RMR,RMB,RMT,SCALE(4),WALLSZ,CEILSZ

PLOT THE PLUME

RADIUS = 0.5 * WIDTH
THETA = 11. / 57.1
VIRTUAL = RADIUS / ATAN(THETA)
RTAN = ATAN (THETA)

DRAW TWO LINES FROM THE VIRTUAL POINT WITH AN ANGLE
OF THETA AND VISIBLE SEGMENTS
FROM PBOTM TO PTOP

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)

PROGRAM TO TEST THE SURFACE PLOTTING AND CONTOUR ROUTINES.

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., 900., 200., 800., 100., 900., 200., 800.,
5 200., 900., 100., 750., 200., 900., 100., 750.,
6 100., 900., 200., 800., 100., 900., 200., 800.,
7 200., 900., 100., 750., 200., 900., 100., 750.,
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./

INITIALIZE THE DATA

DO 20 J = 1, NY
  DO 20 I = 1, NX
    Z(I, J) = ABS(FLOAT(J) - 7.)**2 + (144. - ABS(FLOAT(I) - 13.)**2)
  CONTINUE

DO 2 I = 1, 8
  FL(I) = -1.0 + FLOAT(I)*10.
  FL(I+8) = 0.2 + FLOAT(I)*10.

INITIALIZE THE GRAPHICS PACKAGE

CALL DEFINE (0., 0., 1023., 1023.)
CALL NEWFRM

LOOP OVER THE VARIOUS PERSPECTIVE PLOTS.

IPP = 3
DO 100 IPP = 1, NP
  IP = INDX(IPP)

PLOT THE UPPER PART OF THE SURFACE WITH FRONT SKIRT.

CALL SRFSET (XP(I, IP), YP(I, IP), ZMIN, ZMAX, NX, NY)
CALL ERASE
SUBROUTINE SADDLE

CALL SETLUT
CALL COLOR(5)
CALL SURFAC(Z, NX, NY, +1)
CALL COLOR(4)
CALL SFRAME(2)
CALL FRAME
CALL IOWAIT(2000)

C PLOT THE LOWER PART OF THE SURFACE.

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 PLOT BOTH PARTS OF THE SURFACE WITH SIDE SKIRTS.

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
  CALL IOWAIT(2000)
ENDIF
CONTINUE

CALL ENDFRM

STOP
END
SUBROUTINE SETCOL

SUBROUTINE SETCOL(IC)
DIMENSION LUT(16,3)

C
DO 1 I = 1, 3
DO 1 J = 1, 16
LUT(J,I) = 0

DO 2 I = 1, 8
LUT(I+1,2) = I*2 - 1
LUT(I+1,3) = I*2 - 1'

DO 3 I = 1, 7
LUT(I+9,1) = I*2 + 1
LUT(I+9,2) = I*2 + 1

CALL DSLWT (16, 48, LUT)

RETURN
END
SUBROUTINE SYMTST

C  A SIMPLE TEST OF THE CHARACTER STUFF
C
CHARACTER*28 TITLE1/'A SET OF HARDWARE CHARACTERS'/
CHARACTER*19 TITLE2/'SET FROM SET 1'/
CHARACTER*32 TITLE3/'A SET ([/$/*]) FROM SET #10'/

ISC=0

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:1))
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 VOLST(ID)

VOLUME TEST PROGRAM WITH REDUCED RESOLUTION HIDDEN LINES.

THIS PROGRAM DEMONSTRATES THE USE OF THE "VOLUME" PLOTTER.
IT USES A SIMPLE SOLID ( TWO OVERLAPPING SPHERES) WITH A SOLID PILLAR PLACED IN FRONT TO OBSCURE PART OF THE SPHERES AND DEMONSTRATE THE HIDDEN LINE FEATURE. THIS IS DONE TOGETHER WITH THE NECESSARY INTERACTION WITH THE GRAPHICS PACKAGE. THIS SHOULD BE USED AS A MODEL FOR DESIGNING PROGRAMS UN THE USER IS FAMILIAR WITH VOLUME.

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/
DATA 1 100., 800., 250., 900., 100., 800., 250., 900.,
DATA 2 100., 900., 200., 800., 100., 900., 200., 800.,
DATA 3 200., 900., 100., 750., 200., 900., 100., 750./
DATA YP/
DATA 1 100., 100., 250., 250., 800., 800., 900., 900.,
DATA 2 100., 100., 250., 250., 800., 800., 900., 900.,
DATA 3 100., 100., 250., 250., 800., 800., 900., 900./

INITIALIZE THE PACKAGE

CALL DEFINE(0., 0., 1024., 1024.)

LOOP OVER FOUR SIZES OF THE SPHERES

DO 100 IPIC = 1, 901, 300
NP = NP
IP = 1
FAC = 1.0 + 0.01*FLOAT(IPIC)

SET UP THE VALUE OF THE FUNCTION.
IT IS AN NRL PEANUT ( 2 OVERLAPPING SPHERES)

(X-13)**2 + (Y-10.5)**2 + (Z-10.5)**2
(X-8)**2 + (Y-10.5)**2 + (Z-10.5)**2

DO 1 I = 1, NX
DO 1 J = 1, NY
DO 1 K = 1, NZ
RISQ = (I-8.0)**2 + (J-10.5)**2 + (K-10.5)**2
R2SQ = (I-13.0)**2 + (J-10.5)**2 + (K-10.5)**2
F(I,J,K) = FAC/RISQ + 2.0*FAC/R2SQ

CONTINUE

CONSTRUCT THE PILLAR
SUBROUTINE VOLTST

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 INITIALIZE THE PLOTTING PACKAGE
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.).

CALL NEWFRM
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)

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)

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)

CALL COLOR(4)
CALL VOLUME (-2, F, NX,NY,NZ, 0.5, 1, T, NT)
CALL VOLUME (+1, F, NX,NY,NZ, 1.0, 1, T, NT)
CALL VOLUME (-2, F, NX,NY,NZ, 1.0, 1, T, NT)
CALL FRAME
CALL IOWAIT(2000)

CONTINUE

CLOSE-OUT GRAFIT.

IF(ID.EQ.3) PAUSE
CALL ENDFRM
STOP
END
SUBROUTINE FILNGT

1060 SUBROUTINE FILNGT
1061 C
1062 CALL NEWFRM
1063 CALL DEFINE(-11.,-11.,11.,11.)
1064 DO 2 I = 1, 9
1065 CALL FILTYP(I-1)
1066 X = -11. + FLOAT(I)*2.1
1067 Y = -11. + FLOAT(I)*2.1
1068 CALL CIRCLE(X, Y, 2.0)
1069 2 CONTINUE
1070 CALL FRAME
1071 PAUSE
1072 CALL ENDFRM
1073 STOP
1074 END
<table>
<thead>
<tr>
<th>SUBROUTINE NAME</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALABEL</td>
<td>1</td>
</tr>
<tr>
<td>FUNCTION AND</td>
<td>2</td>
</tr>
<tr>
<td>BOXPLT</td>
<td>3</td>
</tr>
<tr>
<td>ENTRY VBXPLT</td>
<td>3</td>
</tr>
<tr>
<td>CHPLOT</td>
<td>4</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>5</td>
</tr>
<tr>
<td>CONTUR</td>
<td>6</td>
</tr>
<tr>
<td>DEVICE</td>
<td>10</td>
</tr>
<tr>
<td>SETDEV</td>
<td>12</td>
</tr>
<tr>
<td>NEWFRM</td>
<td>12</td>
</tr>
<tr>
<td>FRAME</td>
<td>12</td>
</tr>
<tr>
<td>ERASE</td>
<td>12</td>
</tr>
<tr>
<td>HDCOPY</td>
<td>12</td>
</tr>
<tr>
<td>COLOR</td>
<td>12</td>
</tr>
<tr>
<td>FILTYP</td>
<td>12</td>
</tr>
<tr>
<td>CHRSIZ</td>
<td>13</td>
</tr>
<tr>
<td>LINWID</td>
<td>13</td>
</tr>
<tr>
<td>SCALNG</td>
<td>13</td>
</tr>
<tr>
<td>DEFINE</td>
<td>13</td>
</tr>
<tr>
<td>ENDFRM</td>
<td>14</td>
</tr>
<tr>
<td>DELAY</td>
<td>20</td>
</tr>
<tr>
<td>IOWAIT</td>
<td>20</td>
</tr>
<tr>
<td>LEXDID</td>
<td>20</td>
</tr>
<tr>
<td>TABLET</td>
<td>20</td>
</tr>
<tr>
<td>DEVINP</td>
<td>22</td>
</tr>
<tr>
<td>ENUMBR</td>
<td>24</td>
</tr>
<tr>
<td>FNUMBR</td>
<td>25</td>
</tr>
<tr>
<td>GRAFIT</td>
<td>28</td>
</tr>
<tr>
<td>PLOTCH</td>
<td>31</td>
</tr>
<tr>
<td>PLOTLN</td>
<td>31</td>
</tr>
<tr>
<td>PLOTBAR</td>
<td>32</td>
</tr>
<tr>
<td>PLOTBAR</td>
<td>32</td>
</tr>
<tr>
<td>GRISET</td>
<td>32</td>
</tr>
<tr>
<td>GRILAB</td>
<td>32</td>
</tr>
<tr>
<td>REDRAW</td>
<td>34</td>
</tr>
<tr>
<td>ESETS</td>
<td>36</td>
</tr>
<tr>
<td>IDCEAR</td>
<td>38</td>
</tr>
<tr>
<td>IREVCH</td>
<td>38</td>
</tr>
<tr>
<td>LABEL</td>
<td>39</td>
</tr>
</tbody>
</table>
SUBROUTINE LINE .................................................. page 40
SUBROUTINE LINES ................................................ page 41
SUBROUTINE LNPLT ................................................ 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 PUTC ................................................ 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 VOLLFRM .......................................... page 83
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBROUTINE VOLUMO</td>
<td>84</td>
</tr>
<tr>
<td>ENTRY VOLUM6</td>
<td>84</td>
</tr>
<tr>
<td>SUBROUTINE VOLUM7</td>
<td>85</td>
</tr>
<tr>
<td>SUBROUTINE VOLUM8</td>
<td>87</td>
</tr>
<tr>
<td>SUBROUTINE VOLUME</td>
<td>89</td>
</tr>
<tr>
<td>ENTRY VOLSET</td>
<td>92</td>
</tr>
<tr>
<td>SUBROUTINE WDCOUNT</td>
<td>94</td>
</tr>
<tr>
<td>SUBROUTINE WDDRAW</td>
<td>96</td>
</tr>
<tr>
<td>ENTRY CHRSET</td>
<td>99</td>
</tr>
</tbody>
</table>
SUBROUTINE ALABEL

CHARACTER*1 CHARS(*)

CALL WDCOUNT(CHARS, NC)

IF(NC.LE.0) THEN
  IF(LUDIAG.GT.0) WRITE(LUDIAG,3) (CHARS(I),I=1,132)
  FORMAT(' NO ESCAPE SEQUENCE IN ALABEL -',/1X,132A1)
  RETURN
ENDIF

DX = (XR - XL) / MAX(1.,FLOAT(NC))
DY = 4./3. * DX * XYCOORD(1)/XYCOORD(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 BOXPLT
C******************************************************************************
C
REAL XA(4), XB(4), YA(4), YB(4)

CENTRY 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)

XARRAY REAL - AN ARRAY, DIMENSIONED N, OF X COORDINATES OF
    THE DATA (MATH SPACE)
YARRAY REAL - AN ARRAY, DIMENSIONED N, OF Y COORDINATES OF
    THE DATA (MATH SPACE)
CHARAC LITERAL - CHARACTER TO BE PLOTTED
I1 INTEGER - INDEX OF FIRST POINT TO BE PLOTTED
I2 INTEGER - INCREMENT AT WHICH POINTS ARE TO BE SELECTED
I3 INTEGER - INDEX OF LAST POINT IN DATA ARRAYS

COMMON /DEVTYP/ IDEVIC,LSW,LTSW,XYCOORD(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)

DRAW A CIRCLE OF WIDTH LINWD and FILL WITH LFLMAT.

COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOORD(4),LUOUT,LPAGE
COMMON/GRFMOD/LFLMAT(5),LINWD(5),LCLMAT(5)
REAL XO, YO, XN, YN, DX, DY, THETA, ARC

IF(IDEVIC.EQ.0) RETURN
GO TO (1,1,2,1,1), IDEVIC

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
CONTINUE

RETURN

XL = MAX(X * XYCOORD(1) + XYCOORD(2), 0.)
YL = MAX(Y * XYCOORD(3) + XYCOORD(4), 0.)
RL = R * MIN(XYCOORD(1), XYCOORD(3))
CALL GMOVAXL, 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), YT3(NPT),
     FT1(NPT), FT2(NPT), FT3(NPT)

REAL TOO(NPT), TO1(NPT), TO1(NPT), T11(NPT)

SUBARRAYS OUT OF THE DATA CANNOT BE CONTOURED. THE
ENTIRE F(I,J) ARRAY IS CONTOURED SO A COPY OVER INTO SCRATCH SPACE
IS REQUIRED IF ONLY A PORTION OF F IS TO BE PLOTTED.

F REAL ARRAY (NX,NY) THE REAL VALUES OF THE FUNCTION TO
CONTOURED. A TOPOLOGICALLY SQUARE
GRID CELL IS ASSUMED AND DX = DY = 1.

TEST REAL ARRAY (NX,NY) A USER SUPPLIED SCRATCH ARRAY OF
THE SAME DIMENSIONALITY AS F.

NX INTEGER RANGE AND DIMENSION OF I IN F(I,J).
NY INTEGER RANGE AND DIMENSION OF J IN F(I,J).

FL REAL THE VALUE OF F(I,J) FOR CONTOURING.

CALCULATE THE AVERAGE VALUE OF F AT THE CENTERS OF THE CELLS.

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)

   TEST(I,J) = 0.25*TEST(I,J)

NOW CALCULATE THE CROSSINGS WHICH PARELLEL THE "I" AXIS.

DO 300 J = 1, NY
   DO 300 I = 1, NXM

   TOO(I) = (F(I+1,J) - FL)*(FL - F(I,J))

   A CROSS OCCURS IF THE CONTOUR PASSES THRU F(I,J) IN THE BOX I->I+1

   DO 309 I = 1, NXM
      IF (TOO(I) .LT. 0.0) GO TO 309

   THE LINE SEGMENT IS A HIT. TREAT IN A SCALAR WAY FIRST THE UPPER
   AND THEN THE LOWER TRIANGLE.

   IF (J.EQ.NY) GO TO 305
   IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
           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)
SUBROUTINE CONTUR

305 IF (J.EQ.1) GO TO 309
IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1, YT1, FT1, XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)

310 CONTINUE

NEXT CALCULATE THE CROSSINGS ALONG THE "J" AXIS.

DO 360 J = 1, NYM
DO 350 I = 1, NX
T11(I) = (F(I,J+1) - FL)*(FL - F(I,J))

CORSS OCCURS IF THE CONTOUR VALUE PASSES THRU THE BOX IN F(I,J)
IN THE RANGE J->J+1

DO 359 I = 1, NX
IF (T11(I) .LT. 0.0) GO TO 359
THE LINE SEGMENT IS A HIT. TREAT IN A SCALAR WAY FIRST THE RIGHT
AND THEN THE LEFT TRIANGLE.

IF (I.EQ.NX) GO TO 355
IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1, YT1, FT1, XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)

NTRIA = NTRIA + 1
SUBROUTINE CONTUR

219  FT1(NTRIA) = F(I,J)
220  XT2(NTRIA) = FLOAT(I)
221  YT2(NTRIA) = FLOAT(J + 1)
222  FT2(NTRIA) = F(I,J+1)
223  XT3(NTRIA) = FLOAT(I) - 0.5
224  YT3(NTRIA) = FLOAT(J) + 0.5
225  FT3(NTRIA) = TEST(I-1,J)
226 CONTINUE
227 CONTINUE
228
229 C NOW SEEK ALL TRIANGLES WITH TWO DIAGONAL CROSSINGS.
230 C
231 DO 490 J = 1, NYM
232 DO 420 I = 1, NxM
233 T00(I) = (TEST(I,J) - FL)*(FL - F(I,J))
234 T10(I) = (FL - TEST(I,J))*(F(I+1,J) - FL)
235 T01(I) = (FL - TEST(I,J))*(F(I,J+1) - FL)
236 420 T11(I) = (TEST(I,J) - FL)*(FL - F(I+1,J+1))
237 C
238 C CONSIDER THE LOWER TRIANGLE IN THE SQUARE.
239 C
240 DO 430 I = 1, NxM
241 IF (AMIN1(T00(I), T10(I)) .LT. 0.0) GO TO 430
242 IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1, YT1, FT1,
243 1 XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)
244 NTRIA = NTRIA + 1
245 XT1(NTRIA) = FLOAT(I) + 0.5
246 YT1(NTRIA) = FLOAT(J) + 0.5
247 FT1(NTRIA) = TEST(I,J)
248 XT2(NTRIA) = FLOAT(I)
249 YT2(NTRIA) = FLOAT(J)
250 FT2(NTRIA) = F(I,J)
251 XT3(NTRIA) = FLOAT(I + 1)
252 YT3(NTRIA) = FLOAT(J)
253 FT3(NTRIA) = F(I+1,J)
254 CONTINUE
255 C
256 C CONSIDER THE RIGHT TRIANGLE IN THE SQUARE.
257 C
258 DO 440 I = 1, NxM
259 IF (AMIN1(T10(I), T11(I)) .LT. 0.0) GO TO 440
260 IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1, YT1, FT1,
261 1 XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)
262 NTRIA = NTRIA + 1
263 XT1(NTRIA) = FLOAT(I) + 0.5
264 YT1(NTRIA) = FLOAT(J) + 0.5
265 FT1(NTRIA) = TEST(I,J)
266 XT2(NTRIA) = FLOAT(I + 1)
267 YT2(NTRIA) = FLOAT(J)
268 FT2(NTRIA) = F(I+1,J)
269 XT3(NTRIA) = FLOAT(I + 1)
270 YT3(NTRIA) = FLOAT(J + 1)
271 FT3(NTRIA) = F(I+1,J+1)
272 CONTINUE
273
CONSIDER THE UPPER TRIANGLE IN THE SQUARE.

```
274  C    DO 450 I = 1, NXM
275   C    IF (AMIN1(T1(I), T0(I)) .LT. 0.0) GO TO 450
276     C    IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
277       1       XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
278  C    NTRIA = NTRIA + 1
279     C    XT1(NTRIA) = FLOAT(I) + 0.5
280     C    YT1(NTRIA) = FLOAT(J) + 0.5
281     C    FT1(NTRIA) = TEST(I,J)
282     C    XT2(NTRIA) = FLOAT(I + 1)
283     C    YT2(NTRIA) = FLOAT(J + 1)
284     C    FT2(NTRIA) = F(I+1,J+1)
285     C    XT3(NTRIA) = FLOAT(I)
286     C    YT3(NTRIA) = FLOAT(J + 1)
287     C    FT3(NTRIA) = F(I,J+1)
288  C    CONTINUE
289  450   CONTINUE
290     C    CONSIDER THE LEFT TRIANGLE IN THE SQUARE.
291     C
292     C  DO 460 I = 1, NXM
293     C  IF (AMIN1(T0(I), T00(I)) .LT. 0.0) GO TO 460
294     C  IF (NTRIA .GE. NTMAX) CALL PROPOL (NTRIA, XT1,YT1,FT1,
295       1       XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
296     C  NTRIA = NTRIA + 1
297     C  XT1(NTRIA) = FLOAT(I) + 0.5
298     C  YT1(NTRIA) = FLOAT(J) + 0.5
299     C  FT1(NTRIA) = TEST(I,J)
300     C  XT2(NTRIA) = FLOAT(I)
301     C  YT2(NTRIA) = FLOAT(J + 1)
302     C  FT2(NTRIA) = F(I,J+1)
303     C  XT3(NTRIA) = FLOAT(I)
304     C  YT3(NTRIA) = FLOAT(J)
305     C  FT3(NTRIA) = F(I,J)
306  460   CONTINUE
307  490   CONTINUE
310    C
311    C    IF THE BUFFER IS ONLY PARTIALLY FULL, FLUSH AT THE END.
312    C
313    C  IF (NTRIA .GT. 0) CALL PROPOL (NTRIA, XT1,YT1,FT1,
314       1       XT2,YT2,FT2, XT3,YT3,FT3, FL, NX, NY)
315    C
316    C    RETURN
317    C    END
```
SUBROUTINE DEVICE (N)

DEVICE
1. CALCOMP PLOTTER
2. PRINTER PLOT
3. LEXIDATA DISPLAY UNIT WITH MATRIX CAMERA
4. ADM WITH RETROGRAPHICS OR TEK 40XX SERIES
5. EMPTY SLOT

COMMANDS
DEVICE - SELECT DEVICE (1-5)
SETDEV - SET THE LOGICAL UNITS FOR OUTPUT
LINWID - NUMBER OF STROKES IN A LINE
FILTY - TYPE OF FILLING FOR POLYGONS AND CIRCLE
CHRSIZ - HEIGHT OF CHARACTERS (IN RASTER UNITS)

1. NEWFRM - SET UP DEVICE FOR A NEW PLOT
2. FRAME - FORCE A FLUSH OF ALL BUFFERS
3. ERASE - ERASE THE SCREEN
4. HDCOPY - HARD COPY COMMAND - SHUTTER ON MATRIX
5. COLOR - SELECT THE COLOR TO DRAW
6. SCALE - SCALE FOR PHYSICAL DEVICE
7. ENDFRM - TERMINATE THE DEVICE AND ADVANCE

DEFAULT DEVICE ASSIGNMENTS

<table>
<thead>
<tr>
<th>TYPE</th>
<th>LU</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAGNOSTICS</td>
<td>0</td>
<td>NO OUTPUT</td>
</tr>
<tr>
<td>GRAPHICS</td>
<td>7</td>
<td>NORMAL - ASSIGNED</td>
</tr>
<tr>
<td>CAMERA</td>
<td>8</td>
<td>ASSIGNED WITH LEXIDATA</td>
</tr>
<tr>
<td>CHARACTER SET</td>
<td>0</td>
<td>UNIT 0 - ASSIGNED</td>
</tr>
<tr>
<td>CALCOMP</td>
<td>L7</td>
<td>RS232 LINE</td>
</tr>
<tr>
<td>PRINTER</td>
<td>PR</td>
<td>WHATEVER</td>
</tr>
<tr>
<td>LEXIDATA</td>
<td>LE</td>
<td>RS232</td>
</tr>
<tr>
<td>MATRIX CAMERA</td>
<td>TEKTRONIX</td>
<td>CONSOLE</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>NULL: BIT BUCKET</td>
</tr>
<tr>
<td>CONSOLE INPUT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>CHARACTER SET</td>
<td>9</td>
<td>(CLOSED THEN OPENED)</td>
</tr>
<tr>
<td>TABLET INPUT</td>
<td>8</td>
<td>(CLOSED THEN OPENED -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAMERA AND TABLET CAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT BE ACTIVE SIMULTANEOUSLY)</td>
</tr>
</tbody>
</table>

INTEGER LB, COUNT(5), NFRAME(5), PEN, COMMAND, 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, OPNCAL, OPNLEX, IRVRSE

LOGICAL OPNMAT, OPNTAB, OPENTB

COMMON/DEVTYPE/IDEVIC, LSW, LTSW, XYGOOD(4), LUOUT, LPAGE

COMMON/GRFTYP/Angle, IRVRSE, CHSIZE(9), ITKWIT, LUDIAG, LUHSET

COMMON/DEVPOS/XIR(5), YIR(5), X2R(5), Y2R(5)

COMMON/GRFMOD/LFLMAT(5), LINWDM(5), LCLMAT(5)
subroutine device

373 equivalence (hwsizE(1),calras),(hwsizE(9),lexras)
374 equivalence (hwsizE(13),tekras)
375 data start,line,slash,endit/4h( 4h108h, 4h /, 4h )/
376 data hblank, hminus, haapos, haaposb, hastrA, hastrb
377 1 /4h , 4h---- , 4h 1 , 4h*--- , 4h---* /
378 data openany/.false./, opnlex/.true./
379 data itimet/20/, idletm/50/, ludiaG/0/
380 data count/5*0/, nframe/5*0/, angle/0.0/
381 data tekras/0,0,1022,768/,lexras/50,50,32700,26160/
382 data calras/0,0,10750,7900/
383 data idflto/l7: ', 'pr: ', 'lex: ', 'c: ', 'null: '/
384 data xir/0.0.,1.e+7,0.0., x2r/2*1279.,11279000.,2*1279./
385 data yir/0.0.,1.e+7,0.0., y2r/2*1023.,11023000.,2*1023./
386 data luset/7/, icodeT/27,23/, ictrlx/31,24/
387 data ichanx/27,97,27,127/, opncal/.true./
388 data nframe/5*0/, count/5*0/, luhsset/9/, opnmat/.true./
389 data lplmat/0,0,1,0,0/, linwdm/1,1,2,1,1/, lclmat/5*1/
390 data matunt/8/, matfile/1l4: '/, tabfile/1l9: '/, tabunt/8/
391 data xycood/1.,0.,1.,0./, opntab/.true./
392 data ghsizE/31.,31.,0.3,0.5,0.04,0.58,0.0.,0.,0.03/
393 data uspat/3*3640,3*455,3*3640,3*455,0,0,778,
394 . 992,504,240,240,504,992,778,0,0,2*0,220,270,2f8,
395 . zifc,23fe,27ff,4*0,
396 . 0,96,240,504,1020,2046,4095,3999,3855,3591,3075,2049/
397 c
398 c *** start of the section which handels functions
399 c
400 c device
401 c
402 c if(openany) go to 1100
403 c
devic = n
404 c
405 c lout = luset
406 c
407 c if (idevic.lt.1.or.idevic.gt.5) idevic = 4
408 c
409 c close(lout)
410 c
411 c open(unit=lout, iostat=ios, err=901, file=idflto(idevic),
412 c type='device', share='erw', rkey=0, wkey=0)
413 c
414 c openany = .true.
415 c
416 c if (ludiag.ne.0) write (ludiag,1003) idevic
417 c
418 c return
419 c
420 c error on call to device
421 c
422 c
423 c close(lout)
424 c
425 c if (ludiag.ne.0) write (ludiag,1014) idevic, ios
426 c
427 c setdev - change the logical unit
428 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

NEWFRM
ENTRY NEWFRM
MODEP = 1
IF (.NOT.OPENANY) GO TO 1000
COUNT(IDEVIC) = COUNT(IDEVIC) + 1
NFRAME(IDEVIC) = 1
IRVERSE = .FALSE.
IF (LUDIAG.NE.0) WRITE (LUDIAG, 1101) COUNT(IDEVIC)
GO TO (1,2,3,4,5), IDEVIC

FRAME - FORCE A WRITE OF THE BUFFER
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

ERASE
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

HARD COPY
ENTRY HDCPY
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

COLOR
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

SET THE TYPE FOR POLYGON AND CIRCLE FILLING 0, 1, 2, OR 3
ENTRY FILTP(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 SET THE SIZE OF THE HARDWARE CHARACTERS
C
ENTRY CHRSIZ (CHFRZ, GCHFRZ)
IF(.NOT.OPENANY) GO TO 1000
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 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 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.GE.0.0) HWSIZE(IND+2) = Y1H * SCLRAS
IF(X2H.GE.0.0) HWSIZE(IND+3) = X2H * SCLRAS
IF(Y2H.GE.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
C ENTRY DEFINE (X1, Y1, X2, Y2)
C MODEP = 6
C
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
XYC00D(4) = Y1R(IDEVIC) - XYC00D(3)*Y1
IF (LUDIAG.NE.0) WRITE (LUDIAG, 1400) XYC00D
CHSIZE(1) = CHSIZE(9) * (Y2R(IDEVIC)-Y1R(IDEVIC))
CHSIZE(2) = CHSIZE(1)
GO TO (1,2,3,4,5), IDEVIC
ENDFRM - TO CLOSE THE LOGICAL UNIT
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
CLOSE(7)
OPENANY = .FALSE.
IDEVIC = 0
RETURN

*** START OF THE SECTION WHICH HANDLES THE HARDWARE ***
CALCOMP SECTION
GO TO (101,102,103,105,106,107), MODEP
CALCOMP INITIALIZATION
CALL CALBUF (LOUT,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
CALCOMP FRAME (FLUSH BUFFER AND RESET, IGNORE HARDCOPY)
RETURN
ERASE THE SCREEN AND PRINT HARDCOPY ARE IGNORED FOR THE CALCOMP
COLOR SELECTION PRINTOUT.
PEN = MOD(ICOLOR-1,4) + 1
PEN = MIN0(MAX0(PEN,1),4)
LCLMAT(IDEVIC) = PEN
CALL CALPEN(PEN)
RETURN
CALCOMP AXIS SET.
SUBROUTINE DEVICE

593 C
594 106 CONTINUE
595 CALL CALPLT (FLOAT(CALRAS(3))/((X2R(1)-X1R(1))*1000.),
596 . FLOAT(CALRAS(4))/((Y2R(1)-Y1R(1))*1000.), 1001)
597 RETURN
598 C
599 C ENDFRM FOR CALCOMP
600 C
601 107 CONTINUE
602 CALL CALPLT(0., 0., 999)
603 GO TO 10
604 C
605 C PAPER PLOT SECTION
606 C
607 2 GO TO (201,202,203,204,205,206,207), MODEP
608 C
609 C PAPER PLOT INITIALIZATION.
610 C THE PAGE PLOT IS OUTPUT AS ONE LARGE VARIABLE FORMAT.
611 C SET UP THE FORMAT STATEMENT IN LPAGE.
612 C
613 201 CONTINUE
614 203 DO 211 J=1,65
615 LPAGE(2,J)=LINE
616 211 LPAGE(30,J)=SLASH
617 LPAGE(1,1)=START
618 LPAGE(30,65)=ENDIT
619 C
620 C SET UP THE PAGE AND CLEAN THE WORK AREA.
621 C
622 DO 221 J = 1, 65
623 DO 221 I = 4, 29
624 221 LPAGE(I,J) = HBLANK
625 DO 222 I = 5, 28
626 LPAGE(I,1) = HMINUS
627 222 LPAGE(I,63) = HMINUS
628 DO 223 J = 2, 62
629 LPAGE(4,J) = HAPOSA
630 223 LPAGE(29,J) = HAPOSB
631 LPAGE(29,1) = HASTRB
632 LPAGE(29,63) = HASTRB
633 LPAGE(4,63) = HASTRA
634 LPAGE(4,1) = HASTRA
635 RETURN
636 C
637 C PAPER PLOT- WRITE OUT THE PAGE AS IT IS.
638 C
639 202 WRITE(LUOUT, LPAGE)
640 RETURN
641 C
642 C HARDCOPY COMMAND
643 C
644 204 RETURN
645 C
646 C COLOR SELECTION PRINTOUT.
SUBROUTINE DEVICE

RETURN

C SET SCALING FOR PAGE PLOT
RETURN

ENDFRM FOR THE PAPER PLOTTER

WRITE (LUOUT,1212)
GO TO 10

CLEXDATA/MATRIX SECTION

GO TO (301,302,303,304,305,306,307), MODEP

INITIALIZE THE LEXIDATA/MATRIX - OPEN AND SET THE LUT ONLY

IF THE DEVICE IS/WAS CLOSED

IF(OPNLEX) THEN
CALL GSOPLTUOUT, 1, LIERR)
IF(LIERR.NE.0) GO TO 1002

CURSOR OFFSET - THIS IS INSTALLATION SPECIFIC

CALL DSCSL(10, 153, 75)
CALL DSCER
CALL SETLUT ; SET THE DEFAULT LOOK UP TABLE
CALL GSTYPE(3)

SET VIEW CORRESPONDENCE

CALL GDEFSV (1, 15, 0, 1, 4, 0)
CALL GACTV (1)
CALL GDFWIN (1, XIR(IDDEVIC), Y1R(IDDEVIC), X2R(IDDEVIC),
Y2R(IDDEVIC))
CALL GDFV (1, LEXRAS(1), LEXRAS(2), LEXRAS(3), LEXRAS(4))
VSNUM(1) = 2
VSNUM(2) = 0
CALL GDFV (1, 1, 1, VSNUM)
CALL GACTVU(1)
CALL GDASEG
CALL GCRSEG(IOO)
DO 3001 I = 1, 4
CALL GDPAT(100+I, USPAT(1,I))
CALL GCLSEG
CALL GSVUAS(IOO, VSNUM)
CALL GDLSEG(IOO)
CALL GSIVIS(1)
CALL GSVUAS (0, VSNUM)

DEFINE LOCATOR PORT EXTENTS AND SET TRACKING TYPE

CALL GDVINI(DEVID, 2, 1)
CALL GDFLPLT(DEVID, LEXRAS(1), LEXRAS(2), LEXRAS(3), LEXRAS(4))
CALL GSTTRK(DEVID, 2, 1)

SET ECHO TYPE
CALL GSTDVE(DEVID, 1, 1)
CALL GDVENB(DEVID)
CALL GRSLG (DEVID, 255)
OPNLEX = .FALSE.
ENDIF

SET THE PARAMETERS FOR THE LEXIDATA DISPLAY.
CALL GCLSEG
CALL DSCLR (-1)
CALL GCTSEG
CALL GSFTYP (LFLMAT (IDEVIC))
CALL GSLWID (LINWDM (IDEVIC))
CALL GSCHSZ (CHSIZE (2))
CALL GSCNDX (1)
RETURN

FRAME - FORCE OUT THE CONTENTS OF THE BUFFER
CALL GCLSEG
CALL GMPCUR
CALL GCTSEG
RETURN

ERASE THE SCREEN AND THE CURSOR
CALL DSCLR (-1)
CALL DSCSL (10, 153, 75)
RETURN

HARD COPY - CAMERA

IF(OPNMAT) THEN
CLOSE (MATUNT)
OPNTAB = .TRUE.
OPEN (UNIT=MATUNT, IOSTAT=IOS, ERR=314, FILE=MATFILE
, 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

ERROR ON CALL TO DEVICE
CLOSE (MATUNT)
IF (LUDIAG .NE. 0) WRITE (LUDIAG, 1014) MATUNT, IOS
STOP 'camera can not be accessed'

SET COLOR - THIS APPLIES TO WHAT IS IN THE LUT
SUBROUTINE DEVICE

758 C 759 305 LEXCOL = MOD(ICOLOR-1,15) + 1
760 LEXCOL = MAX0(LEXCOL,1)
761 LCLMAT(IDEVIC) = LEXCOL
762 CALL GSCNDX(LCLMAT(IDEVIC))
763 CALL GSLWID(LINWDM(IDEVIC))
764 LFLLEX = LFLMAT(IDEVIC)
765 IF(LFLLEX.LE.4) THEN
766 CALL GSFTYP(LFLLEX)
767 ELSE
768 CALL GSFTYP(96+LFLLEX)
769 ENDIF
770 RETURN
771 C 772 C SET THE WINDOW AND AREA OF NORMALIZED DEVICE COORDINATES
773 C 774 306 IF(OPNLEX) GO TO 301
775 CALL GCLSEG
776 CALL GDAVU(1)
777 CALL GDFWIN (1, XIR(IDEVIC), YIR(IDEVIC), X2R(IDEVIC),
778 . Y2R(IDEVIC))
779 CALL GDFVP (1, LEXRAS(1), LEXRAS(2), LEXRAS(3), LEXRAS(4))
780 CALL GDFVU (1, 1, 1, VSNUM)
781 CALL GACTVU (1)
782 C 783 C OPEN A NEW SEGMENT
784 C 785 CALL GCTSEG
786 RETURN
787 C 788 C 789 307 OPNLEX = .TRUE.
790 IF (.NOT.OPNMAT) THEN
791 CLOSE(MATUNT)
792 OPNMAT = .TRUE.
793 ENDIF
794 CALL GCLSEG
795 CALL GDASEG
796 CALL GFRAME
797 CALL DSCLR(-1)
798 CALL GRS LG(DEVID,255)
799 CALL GDVDSB(DEVID)
800 GO TO 10
801 C 802 C TEKTRONIX SECTION
803 C 804 4 GO TO (401,402,403,404,405,406,407), MODEP
805 C 806 C TEKTRONIX INITIALIZATION
807 C 808 401 CALL TEKINT (120)
809 CALL SWINDO (TEKRAS(1),TEKRAS(3),TEKRAS(2),TEKRAS(4))
810 CALL WINDO (XIR(IDEVIC), X2R(IDEVIC)-XIR(IDEVIC),
811 . YIR(IDEVIC), Y2R(IDEVIC)-YIR(IDEVIC))
812 LTSW = .TRUE.
CALL TEKHM
CALL TTKSND
CALL WAIT (IDLETM, 1, IS)
RETURN

FRAME (NO MODE SWITCHING)

CONTINUE
CALL TTKSND
CALL WAIT (IDLETM, 1, IS)
RETURN

ERASE THE SCREEN

CONTINUE
CALL TEKHM
CALL TEKERA
CALL TOUTST (2, ICTRLX)
CALL TTKSND
CALL WAIT (IDLETM, 1, IS)
RETURN

ISSUE HARD COPY COMMAND

CONTINUE
CALL TOUTST (2, ICODET)
CALL TEKHM
CALL TTKSND
CALL WAIT (ITIMET, 2, IS)
RETURN

COLOR - DOES NOT APPLY TO ADM OR TEKTRONIX 4054

CONTINUE
CALL TOUTST (2, ICODET)
CALL TEKHM
CALL TTKSND
CALL WAIT (ITIMET, 2, IS)
RETURN

SET SCREEN CORRESPONDENCE

CONTINUE
CALL SWINDO(TEKRAS(1), TEKRAS(3), TEKRAS(2), TEKRAS(4))
CALL WINDO (X1R(IDEVIC), X2R(IDEVIC)-X1R(IDEVIC),
Y1R(IDEVIC), Y2R(IDEVIC)-Y1R(IDEVIC))
RETURN

END THE FRAME ON THE ADM OR TEK 4054

CALL TEKHM
CALL TOUTST (2, ICTRLX)
CALL TTKSND
CALL WAIT (IDLETM, 1, IS)
CALL SVSTAT (SCRATCH)
GO TO 10

SLOT FOR AN EXTRA DEVICE
STOP 74

DELAY IN SECONDS FOR HARD COPY UNITS

ENTRY DELAY(ITIME)
ITIME = ITIME
RETURN

IOWAIT

ENTRY IOWAIT(IT)
CALL WAIT(IT, 1, IS)
RETURN

GET THE DEVICE ID FOR THE LEXIDATA TRACKING TYPE

ENTRY LEXDID(IDVL)
IDVL = DEVID
RETURN

OPEN ACCESS TO THE TABLET - IF POSSIBLE

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

THE TABLET HAS GENERATED AN ERROR

CLOSE(TABUNT)
IF(LUDIAG.GT.0) WRITE(LUDIAG, 1303) TABUNT, IOS
STOP 'can not access tablet'

PUNISH IF NOT INITIALIZED.

IF (LUDIAG.NE.0) WRITE (LUDIAG, 1004) MODEP, IDEVIC
IDEVIC = 0
RETURN

IF (LUDIAG.NE.0) WRITE(LUDIAG, 1013) LIERR
STOP 'Lexidata can not be accessed'

IF (LUDIAG.NE.0) WRITE(LUDIAG, 1006) IDEVIC
RETURN

FORMATS
SUBROUTINE DEVICE

923 C
924 1001 FORMAT (' FRAMES PLOTTED = ',I5,' ON DEVICE ',I3)
925 1003 FORMAT (' DEVICE = ',I2,' INITIALIZED')
926 1004 FORMAT (' DEVICE NOT INITIALIZED. ',2I5)
927 1005 FORMAT (' LUOUT = ',I3,' AND WILL BE SET TO ',I3,
928   ' AT THE NEXT CALL TO "DEVICE"')
929 1006 FORMAT (' DEVICE ',I3,' IS ALREADY OPEN')
930 1013 FORMAT (' CAN NOT OPEN LEXIDATA, ERROR =',I3)
931 1014 FORMAT (' DEVICE ',I3,' CAN NOT BE ACESSED, ERROR =',I3)
932 1101 FORMAT (' FRAME NUMBER ',I5,' INITIALIZED')
933 1201 FORMAT (' PLOTTER CLOSED WITH',2I5,' FRAMES')
934 1212 FORMAT ('1')
935 1214 FORMAT (' COLOR SELECT = ',I3)
936 1302 FORMAT(' CAMERA IS ALREADY AVAILABLE')
937 1303 FORMAT (' CAMERA ',I3,' CAN NOT BE ACESSED, ERROR =',I3)
938 1400 FORMAT (' FACTOR ',4F8.2)
939 1501 FORMAT (' ERASE COMMAND ISSUED AT FRAME ',I5)
940 1502 FORMAT (' HARDCOPY COMMAND ISSUED AT FRAME ',I5)
941 1503 FORMAT (' SET THE FILL TYPE FOR SURFACES ',5I3)
942 1504 FORMAT (' SET THE LINE WIDTH ',5I3)
943 1505 FORMAT (' SET THE CHARACTER SIZE (%) ',2F6.3)
944 END
SUBROUTINE DEVINP

LOGICAL JFIRST/.FALSE./, TFIRST/.FALSE./, KFIRST/.FALSE./
INTEGER BTMASK, DEVID, PBLK(6), TLOOK(4)/1, 2, 4, 8/
CHARACTER*1 INST(2)/Z1B, 'A'/
CHARACTER*20 DATAIN
COMMON/DEVTYP/IDVIC, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
COMMON/TTXDAT/TABIN, TABWIN, TABXYC
REAL TABIN(2), TABWIN(4), TABXYC(4)
COMMON/DEVPOS/X1R(5), Y1R(5), X2R(5), Y2R(5)

SELECT THE INPUT FROM THE DEVICE CODE

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

GET THE DEVICE ID FOR THE JOYSTICK

IF(JFIRST) THEN
    CALL LEIDID(DEVID)
    CALL GDFBGF(DEVID, 15, 0, IDUM)
    CALL DSCSL (8, 153, 75)
    JFIRST = .FALSE.
ENDIF

SET APPROPRIATE LIGHTS

CALL GSLTG(DEVID, BTMASK)

READ THE LOCATOR

XWC = 0.0
YWC = 0.0
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)
CALL GRBTG (DEVID, IDVV)
IF(IDVV.NE.0) GO TO 2

NORMALIZE TO <RE>WINDOWED COORDINATES

X = (XWC-XYCOOD(2)) / XYCOOD(1)
Y = (YWC-XYCOOD(4)) / XYCOOD(3)
Z = 0.0
RETURN
SUBROUTINE DEVINP

THE TABLET HAS BEEN SELECTED

IF(TFIRST) THEN
    TFIRST = .FALSE.
    CALL TABLET(OPENTB)
    IF(IDEVIC.EQ.0) THEN
        DX = 1280.
        DY = 1024.
    ELSE
        DX = (X2R(IDEVIC)-X1R(IDEVIC)) / XYCOORD(1)
        DY = (Y2R(IDEVIC)-Y1R(IDEVIC)) / XYCOORD(3)
    ENDIF
    TABXYC(1) = 8.2034E-5 * DX
    TABXYC(2) = 0.0
    TABXYC(3) = 1.0938E-4 * DY
    TABXYC(4) = 0.0
ENDIF

CALL SYSIO(PBLK,72,8,DATAIN,20,0)
READ(DATAIN,22,ERR=21) IX,IY,IBT

CONVERT THE CURSOR BUTTON TO A STANDARD FORMAT

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

INPUT FROM THE KEYBOARD

CALL SYSIO (PBLK,41,5,'X,Y,Z,IDV = ',12,0)
READ(5,32,END=33,ERR=30) XCOR,YCOR,ZCOR,IDV
FORMAT(3F,I)
X = XCOR
Y = YCOR
Z = ZCOR
IDV = IAND(BTMASK,IDV)
RETURN
IDV = BTMASK
RETURN
SUBROUTINE ENUMBR

REALNO, XR1, YR1, XR2, YR2

DX = (XR2 - XR1)/10.0
THENO = REALNO
EXPON = 0.001
IF (REALNO .EQ. 0.0) GO TO 10
IF (ABS(THENO) .LT. 10.0) GO TO 11
EXPON = EXPON + 1.0
THENO = THENO*0.1000001
GO TO 12
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 THEANO, 5, 2)
11 CALL WDDRAW (XR1+5.0*DX, YR1, DX, 0.0, 0.7*DX, 0.0,
1 YR2-YR1, '*101."
12 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, NUMBR, WIDTH, DIGITS)

C X REAL - X COORDINATE (MATHEMATICAL SPACE) OF THE FIRST CHARACTER TO BE DRAWN
C Y REAL - Y COORDINATE (MATHEMATICAL SPACE) OF THE FIRST CHARACTER TO BE DRAWN
C DX REAL - INCREMENT TO BE ADDED TO THE X COORDINATE FOR EACH CHARACTER DRAWN
C DY REAL - INCREMENT TO BE ADDED TO THE Y COORDINATE FOR 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))

II = 1
I2 = 1
NUMBER = ABS(NUMBR)
IF (DIGITS.EQ.0) NUM = NUMBER + 0.5
IF (DIGITS.EQ.0) GO TO 5
WIDTH = WIDTH - 1
NUM = NUMBER*10**DIGITS +0.5
DO 10 I = 1,WIDTH
  J = WIDTH - I + 1
  NN(J) = NUM - 10*(NUM/10)
10  CONTINUE
BREAK OUT THE DIGITS.
NUM = NUM/10
CONTINUE
J = WIDTH - DIGITS
IF(J .NE. 0) GO TO 18
WORD(1) = TEXT(11)
II = II + 1
GO TO 25
18 SPACE = .TRUE.
DO 20 K = 1,J
SUBROUTINE FNUMBR

I = NN(K)

C LEADING ZERO = SPACE

IF (I.NE.0) SPACE = .FALSE.
IF(K .EQ. J .AND. SPACE) SPACE = .FALSE.
IF (I.EQ.0 .AND. SPACE) I = 10
II = II + 1
HTEMP = ISHFT(WORD(I2), 8)
WORD(I2) = IOR(HTEMP,TEXT(I+1))
IF(I1 .LT. 5) GO TO 20
II = II + 1
I1 = 1
CONTINUE

IF (DIGITS.EQ.0) GO TO 40

C PUT IN DECIMAL POINT

I1 = I1 + 1
HTEMP = ISHFT(WORD(I2), 8)
WORD(I2) = IOR(HTEMP,TEXT(I2))
IF(I1 .LT. 5) GO TO 26
II = II + 1
I1 = 1
DO 30 K = 1,DIGITS
II = II + 1
II = II + 1
HTEMP = ISHFT(WORD(I2), 8)
WORD(I2) = IOR(HTEMP,TEXT(I+1))
IF(I1 .LT. 5) GO TO 30
II = II + 1
I1 = 1
CONTINUE

WIDTH = WIDTH + 1

C PUT IN 1. (END OF TEXT)

I1 = I1 + 1
HTEMP = ISHFT(WORD(I2), 8)
WORD(I2) = IOR(HTEMP,TEXT(I+1))
IF(I1 .LT. 5) GO TO 42
II = II + 1
I1 = 1
GO TO 42
HTEMP = ISHFT(WORD(I2), 8)
WORD(I2) = IOR(HTEMP,TEXT(I2))

C LEFT JUSTIFY LAST WORD.

I1 = I1 + 1
IF(I1 .GE. 5) GO TO 46
WORD(I2) = ISHFT(WORD(I2), 8)
GO TO 44

C PUT IN THE MINUS SIGN
SUBROUTINE FNUMBR

1185 46 IF (NUMBR .GE. 0.0) GO TO 50
1186 IF(WWORD(1).NE.BLANK) GO TO 60
1187 NUMMX = 4 * I2
1188 DO 47 I = 2, NUMMX
1189 IP = I
1190 IF(WWORD(I).NE.BLANK) GO TO 48
1191 CONTINUE
1192 47 WWORD(IP-1) = HMINUS
1193 GO TO 50
1194 60 I2 = WIDTH/4
1195 IF(I2 .EQ. 0)GO TO 64
1196 DO 62 K = 1, I2
1197 62 WORD(K) = HASTR
1198 64 I1 = WIDTH - 4*I2
1199 IF(I1 .GT. 0) WORD(I2+1) = SFIL(I1)
1200 C
1201 50 CALL WDDRAW(XSTART,YSTART,DX,DY,SX,SXY,SY,WORD)
1202 RETURN
1203 END
SUBROUTINE GRAFIT

NPL INTEGER - THE NUMBER OF THE PLOT REFERENCED. UP TO FOUR
PLOTS MAY BE PLACED ON ONE GRAPH

X1 REAL - A FLOATING POINT NUMBER, THE LABEL OF THE X
AXIS MINIMUM. THE LITERAL STRING 'NONE'
PRODUCES NO LABEL

X2 REAL - A FLOATING POINT NUMBER, THE LABEL OF THE X AXIS
MAXIMUM. THE LITERAL STRING 'NONE' PRODUCES NO LABEL

X1R REAL - THE USER SPACE VALUE OF THE X AXIS MINIMUM

X2R REAL - THE USER SPACE VALUE OF THE X AXIS MAXIMUM

IX1 REAL - THE MINIMUM VALUE OF X IN THE DATA TO BE PLOTTED

IX2 REAL - THE MAXIMUM VALUE OF X IN THE DATA TO BE PLOTTED

Y1 REAL - A FLOATING POINT NUMBER, THE LABEL OF THE Y
AXIS MINIMUM. THE LITERAL STRING 'NONE'
PRODUCES NO LABEL

Y2 REAL - A FLOATING POINT NUMBER, THE LABEL OF THE Y
AXIS MAXIMUM. THE LITERAL STRING 'NONE'
PRODUCES NO LABEL

Y1R REAL - THE USER SPACE VALUE OF THE Y AXIS MINIMUM

Y2R REAL - THE USER SPACE VALUE OF THE Y AXIS MAXIMUM

IY1 REAL - THE MINIMUM VALUE OF Y IN THE DATA TO BE PLOTTED

IY2 REAL - THE MAXIMUM VALUE OF Y IN THE DATA TO BE PLOTTED

XTIT CHAR - A STRING OF LITERAL CHARACTERS TERMINATED BY
A '. THIS IS THE TITLE FOR THE X AXIS AND
SHOULD BE LESS THAN 30 CHARACTERS

NDVX INTEGER - THE NUMBER OF INTERVALS TO BE DRAWN ON THE
X AXIS

YTIT CHAR - A STRING OF LITERAL CHARACTERS TERMINATED BY
A '. THIS IS THE TITLE FOR THE Y AXIS AND
SHOULD BE LESS THAN 30 CHARACTERS

NDVY INTEGER - THE NUMBER OF INTERVALS TO BE DRAWN ON THE
Y AXIS

ENTRY POINTS (WITH ARGUMENTS):

PLOTCH (NPL,X,Y,N,CHARAC) - PLOTS ALPHANUMERIC CHARACTERS AT THE
COORDINATES PROVIDED

X REAL - A ONE DIMENSIONAL ARRAY CONTAINING THE
COORDINATES FOR X

Y REAL - A ONE DIMENSIONAL ARRAY CONTAINING THE
COORDINATES FOR Y

N INTEGER - THE NUMBER OF ENTRIES IN 'X' AND 'Y'. IF 'N' =
NO COORDINATES ARE PLOTTED

PCHR CHAR*1 A SINGLE CHARACTER TO BE USED FOR PLOTTING

PLOTLN (NPL,X,Y,N) - PLOTS STRAIGHT LINES THROUGH THE COORDINATES
PROVIDED. ARGUMENTS ARE THE SAME AS FOR
PLOTCH.

PARAMETER (NLINES=200,NPLTMX=4)

REAL IX1, IX2, IY1, IY2, NONE, HOLD(12,NPLTMX), WX(NLINES)

REAL CC(16), C(16), WY(NLINES)
SUBROUTINE GRAFIT

CHARACTER*1 XTIT(NDVX), YTIT(NDVY), PCHR, TITLE(132)
INTEGER IX, IY, IT, NTX, NTY, INDEX(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 NTIX(NPLTMX), NTICY(NPLTMX), XMINO(NPLTMX), XMAXO(NPLTMX)
DIMENSION YMNO(NPLTMX), YMNO(NPLTMX)
COMMON /DEVTPY/ IDEVIC, LSW, LTSW, XCOORD(4), LUOUT, LPAGE
COMMON/GRTYP/ 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 INDEX/3, 3, 3, 9, 9, 3, 3, 3, 3, 3, 3, 9, 9, 3, 3, 9/

C
C define the plotting regions.

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) = XR
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) = Y1Y
HOLD(12,NPL) = Y2Y
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
  C(I) = CC(I) * (HOLD(INDEX(I)+1,NPL)-HOLD(INDEX(I),NPL))
11
C determine number of characters in axis titles.
TITLE(1) = 'a'
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
         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)
      ENDIF
   WRITE (LUDIAG,12) '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
            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
      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)
      ENDIF
      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
      IF(LLABPL(NPL)) THEN
         NDVXLL = NTICX(NPL)
         NDVYLL = NTICY(NPL)
      ELSE
         NDVXLL = NDVX
      ENDIF
SUBROUTINE GRAFIT

NDVYLL = NDVY
ENDIF
IF (NDVXLL.GT.1) THEN
  DELX = (X2R - X1R)/FLOAT(NDVXLL)
  DO 3 I = 1, NDVXLL-1
  XX = X1R + FLOAT(I)*DELX
  CALL LINE (XX, Y1R, XX, Y1R+C(16))
  DO 7 I = 1, NDVXLL-1
  XX = X1R + FLOAT(I)*DELX
  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
  CALL LINE(XIR, YY, XIR+C(15), YY)
  DO 8 I = 1, NDVYLL-1
  YY = Y1R + FLOAT(I)*DELY
  CALL LINE(X2R, YY, X2R-C(15), YY)
END IF
RETURN

C ENTRY PLOTCH (NPL, X, Y, N, PCHR)

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)) * XYCOORD(3) * CHSIZE(5)
NLM = 1
IST = NLM
DO 5 I = 1, NLM
  WX(I-IST+1) = HOLD(3,NPL) + DX * (X(I)-HOLD(5,NPL))
  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)

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
IST = NLM
DO 5 I = 1, NLM
  WX(I-IST+1) = HOLD(3,NPL) + DX * (X(I)-HOLD(5,NPL))
  WY(I-IST+1) = HOLD(9,NPL) + DY * (Y(I)-HOLD(11,NPL))
  CALL LNPLOT (WX, WY, 1, 1, NLM-IST+1)
SUBROUTINE GRAFIT

1424 IF(NLM.GE.N) RETURN
1425 GO TO 14
1426 C
1427 ENTRY PLOTBAR (NPL, XL, YB, XR, YT, N, VALUE)
1428 C
1429 IF(N.LE.0) RETURN
1430 IF(NPL.GT.NPLMX OR NPL.LT.1) STOP 72
1431 IF(.NOT.PLOTON(NPL)) RETURN
1432 DX = (HOLD(4,NPL)-HOLD(3,NPL)) / (HOLD(6,NPL)-HOLD(5,NPL))
1433 DY = (HOLD(10,NPL)-HOLD(9,NPL)) / (HOLD(12,NPL)-HOLD(11,NPL))
1434 NLM = 1
1435 20 IST = NLM
1436 NLM = MIN(NLM+1,NLINES,N)
1437 CALL LINWID(1)
1438 DO 21 I = IST, NLM
1439 XI = HOLD(3,NPL) + DX * (XR(I)-HOLD(5,NPL))
1440 XO = HOLD(3,NPL) + DX * (XL(I)-HOLD(5,NPL))
1441 Y1 = HOLD(9,NPL) + DY * (YT(I)-HOLD(11,NPL))
1442 Y0 = HOLD(9,NPL) + DY * (YB(I)-HOLD(11,NPL))
1443 WX(1) = XO
1444 WX(2) = XO
1445 WX(3) = XI
1446 WX(4) = XI
1447 WY(1) = Y0
1448 WY(2) = Y1
1449 WY(3) = Y1
1450 WY(4) = Y0
1451 CALL PLYGON(WX, WY, 4)
1452 TX = XI - XO
1453 TY = HOLD(10,NPL) - HOLD(9,NPL)
1454 IDIGIT = 0
1455 IF (ABS(VALUE).LT.10.) IDIGIT = 1
1456 NDIG = 3
1457 IF(ABS(VALUE).GE.1000.) NDIG = 4
1458 YOFF = Y1 - .02 * TY
1459 IF (VALUE.LT.0.0) YOFF = Y0 - 0.15 * TY
1460 FAC = 1. / FLOAT(NDIG)
1461 XST = FAC *.95
1462 IF(VALUE.NE.0.0) CALL FNUMBR(XO+0.01*TX,YOFF, ,-.025->+.01
1463 .FAC*TX,0.0,XST*TX,0.0,0.12*TY,ABS(VALUE),NDIG,IDIGIT)
1464 21 CONTINUE
1465 CALL LINWID(3)
1466 IF(NLM.GE.N) RETURN
1467 GO TO 20
1468 C
1469 ENTRY GRISET (XL, YB, XR, YT)
1470 C
1471 CALL DEFINE (XL, YB, XR, YT)
1472 RETURN
1473 C
1474 ENTRY GRILAB (NPL, XLL, YBT, XRT, YTT)
1475 DX = (XRT-XLL) / 3.
1476 DY = (YTT-YBT) / 3.
1477 XINT1 = 10.**(INT(ALOG10(DX)))
1478 YINT1 = 10.**(INT(ALOG10(DY)))
SUBROUTINE GRAFIT

1479    XINT2 = DX / XINT1
1480    YINT2 = DY / YINT1
1481    IF(XINT2.LT.2.0) THEN
1482       DX = XINT1
1483    ELSE IF (XINT2.LT.5.0) THEN
1484       DX = 2.*XINT1
1485    ELSE IF (XINT2.LT.10.) THEN
1486       DX = 5.*XINT1
1487    ELSE
1488       DX = 10.*XINT1
1489 ENDIF
1490    IF(YINT2.LT.2.0) THEN
1491       DY = YINT1
1492    ELSE IF (YINT2.LT.5.0) THEN
1493       DY = 2.*YINT1
1494    ELSE IF (YINT2.LT.10.) THEN
1495       DY = 5.*YINT1
1496    ELSE
1497       DY = 10.*YINT1
1498 ENDIF
1499    XINT2 = XLL / DX
1500    YINT2 = YBT / DY
1501    IF(XINT2.LT.0.0) XINT2 = XINT2 - 0.99999
1502    IF(YINT2.LT.0.0) YINT2 = YINT2 - 0.99999
1503    XMINO(NPL) = DX * (INT(ABS(XINT2))*SIGN(1.,XINT2))
1504    YMINO(NPL) = DY * (INT(ABS(YINT2))*SIGN(1.,YINT2))
1505    XINT2 = XRT / DX
1506    YINT2 = YTT / DY
1507    IF(XINT2.GT.0.0) XINT2 = XINT2 + 0.99999
1508    IF(YINT2.GT.0.0) YINT2 = YINT2 + 0.99999
1509    XMAXO(NPL) = DX * (INT(ABS(XINT2))*SIGN(1.,XINT2))
1510    YMAXO(NPL) = DY * (INT(ABS(YINT2))*SIGN(1.,YINT2))
1511    NTICX(NPL) = (XMAXO(NPL)-XMINO(NPL)+0.5) / DX
1512    NTICY(NPL) = (YMAXO(NPL)-YMINO(NPL)+0.5) / DY
1513    LLABPL(NPL) = .TRUE.
1514    RETURN
1515    END
SUBROUTINE HHDRAW

SUBROUTINE HHDRAW (XCHAR, YCHAR, SX, SXY, SY, CHNUMB, SET, IERR)

XCHAR REAL - X STARTING COORDINATE OF THE CHARACTER TO BE DRAWN
YCHAR REAL - Y STARTING COORDINATE OF THE CHARACTER TO BE DRAWN
SX REAL - X MATH SPACE SIZE FOR CHARACTERS
SXY REAL - SLANT MODIFIER FOR CHARACTERS
SY REAL - Y MATH SPACE SIZE FOR CHARACTERS
CHNUMB INTEGER - INDEX X TO IDENTIFY CHARACTERS WITHIN THE SPECIFIED SET
SET INTEGER - NUMBER SPECIFYING A PARTICULAR CHARACTER SET

REAL XCHAR, YCHAR, SX, SXY, SY, X(128), Y(128)
INTEGER M, CHNUMB, SET, PEN(128), ERROR, IERR
CHARACTER*1 CHIRIV
COMMON /DEVTPY/ IDEVIC, LSW, LSW, XYCOOD(4), LUOUT, LPAGE
COMMON/GRFTYP/ ANGLE, IRVRSE, CHSIZE(9), ITKWIT, LUDIAG, LUHSET

IERR = 1
IF (SET.GT.1) GO TO 3
IF (SET.LT.1) RETURN

IRV = IREVCH(CHNUMB, CHIRIV)
CALL SYMBOL(XCHAR+CHSIZE(7), YCHAR+CHSIZE(8), CHIRIV)
RETURN

3 CALL HSETS (CHNUMB, X, Y, PEN, N, SET-1, ERROR)
IF (ERROR.NE.0) RETURN
IERR = 0

M = N
IF (M.EQ.0) RETURN

ROTATE COORDINATES IF ANGLE>0.0

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
SUBROUTINE HHDRAW

1571 4 DO 1 I = 1, M
1572     X(I) = X(I) * OSXP
1573 1  Y(I) = Y(I) * OSYP
1574 C   ADD THE ABSOLUTE POSITION
1575 C
1576 C
1577   DO 7 I = 1, M
1578     X(I) = X(I) + XCHAR
1579 7  Y(I) = Y(I) + YCHAR
1580 C   WRITE THE CHARACTER AS A SET OF CONNECTED STROKES
1581 C
1582 C
1583     IS = 1
1584     IL = IS
1585 10 IF(PEN(IL+1).EQ.0.OR.IL.GE.M) GO TO 2
1586     IL = IL + 1
1587     GO TO 10
1588 2  IF(IL.GT.IS) CALL LINES(X(IS),Y(IS),X(IS+1),Y(IS+1),IL-IS)
1589     IF(IL.GE.M) RETURN
1590     IL = IL + 1
1591     IS = IL
1592     GO TO 10
1593     RETURN
1594 END
SUBROUTINE HSETS (CHNUMB, XX, YY, PEN, N, SET, ERROR)

C*******************************************************************************
C  HSETS
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.

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)
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

OPEN FILE CONTAINING PACKED HERSHEY CHARACTER SETS IF NOT
PREVIOUSLY OPENED

IF (SET.EQ.NSET.AND.FOPEN) GO TO 41
CLOSE(LUHSET)

OPEN A NEW FILE FOR THE PACKED HERSHEY CHARACTER SUBSETS

OPEN(UNIT=LUHSET, IOSTAT=IOS, ERR=50, FILE='SYS:HI.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
FORMAT('HSET CHARACTER FILE OPENED TO UNIT =', I3)

READ STARTING RECORD POSITIONS OF THE SETS INTO 'TABLE'
READ THE NUMBER OF CHARACTERS IN EACH SET INTO 'NCHAR'

READ (LUHSET) TT

CHECK THAT THE SET AND CHARACTER NUMBERS ARE VALID

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

READ THE CHARACTERS' COORDINATES INTO THE 'IN' ARRAY
SUBROUTINE HSETS

C NSET = SET
1651 FIRST = TABLE(SET) ; FIRST RECORD OF SET
1652 LAST = TABLE(SET+1) - 1 ; LAST RECORD OF SET
1653 CHAR = 1
1654 COORD = 0
1655 DO 30 I = FIRST, LAST
1656 READ(LUHSET, REC=I) IOB
1657
1658 C STORE DATA IN ARRAY ELEMENTS FOR 1 CHARACTER AT A TIME
1659 C
1660 C J = 0
1661 32 J = J + 1
1662 COORD = COORD + 1
1663 IN(COORD, CHAR) = IOB(J)
1664 IF((IOB(J) .NE. MASK1).AND.(J .LT. 64)) GO TO 32
1665
1666 C END OF CHARACTER FOUND; READ COORDINATES FOR NEXT CHARACTER
1667 C
1668 C IF(IOB(J) .EQ. MASK1) THEN
1669 CHAR = CHAR + 1
1670 COORD = 0
1671 IF(CHAR.GT.NCHAR(SET)) GO TO 31
1672 ENDIF
1673 IF(J .LT. 64) GO TO 32
1674 30 CONTINUE ; READ NEXT RECORD
1675 31 CLOSE(LUHSET)
1676 C
1677 C UNPACK THE DATA FOR THE SPECIFIED CHARACTER & DETERMINE THE REAL
1678 C VALUES OF THE COORDINATES AND THE INTEGER VALUE OF THE PEN.
1679 C
1680 C
1681 41 N = 0
1682 40 IF(IN(N+1, CHNUMB) .EQ. MASK1) RETURN ; END OF CHARACTER
1683 N = N + 1
1684 C
1685 C FIND THE 'PEN' VALUE & ELIMINATE 'PEN' BIT FROM PACKED WORD
1686 C
1687 PEN(N) = ISHFT(IN(N, CHNUMB), -31)
1688 IOBC = IAND(IN(N, CHNUMB), MASK1)
1689 C
1690 C FIND THE X & Y COORDINATES
1691 C
1692 XX(N) = (ISHFT(IOBC, -16))/100.00
1693 YY(N) = (IAND(IN(N, CHNUMB), MASK2))/100.0
1694 GO TO 40 ; UNPACK NEXT COORDINATE AND PEN VALUE
1695 C
1696 C ERROR
1697 C
1698 50 ERROR = IOS
1699 IF(LUDIAG .NE. 0) WRITE(LUDIAG, 51) IOS
1700 51 FORMAT(' ERROR IN HSET - UNABLE TO OPEN CHARACTER FILE, ERROR =',
1701 .14)
1702 52 CLOSE(LUHSET)
1703 RETURN
1704 END
INTEGER FUNCTION IDCHAR (CHARX)

C*****************************************************************************
C IDCHAR
C*****************************************************************************

LOGICAL ZEROP/.TRUE./
INTEGER IDN(128), CHAR, KA, MASK/127/
CHARACTER*1 CHARX, CHARXX(4), JA(4), CHIRIV
CHARACTER*8 ASCII(16)
CHARACTER*1 ASCII(128)
EQUIVALENCE (CHAR,CHARXX)
EQUIVALENCE (JA,KA), (ASCII,ASCII)
DATA ASCII/'01234567','89ABCD\'EF', 'GHIJKLMNOP', 'OPQRSTUVWXYZ',

THIS ROUTINE INITIALIZES THE IDENTITY ARRAY ID TO THE IDENTITY NUMBER OF THE ACCEPTED ASCII TERMINAL CHARACTERS.

IF(ZEROP) THEN
  DO 2 I = 1, 128
  IDN(I) = 127
  DO 1 I = 1, 128
  JA(4) = ASCII(I)
  KA = IAND(KA, MASK)
  1 IF(IDN(KA+1).EQ.127) IDN(KA+1) = I
  ZEROP = .FALSE.
ENDIF

THIS INTEGER FUNCTION RETURNS THE IDENTITY NUMBER OF THE ONE BYTE CHARACTER ENTERED THROUGH THE ARGUMENT LIST AS A CHARACTER*1 VARIABLE.

CHARXX(4) = CHARX
CHAR = IAND(CHAR, MASK)
IDCHAR = IDN(CHAR+1)
RETURN

THIS ENTRY REVERSES THE CHARACTER PROCESS - USED BY HHDRAW WHEN CALLING SYMBOL. CHANGES THE NUMBER BACK INTO A CHARACTER

ENTRY IREVCH (ICHR, CHIRIV)
IREVCH = 0
IF(ICHR.GE.1.AND.ICHR.LE.128) CHIRIV = ASCII(ICHR)
RETURN
END
SUBROUTINE LABEL (CHARS, XL, YB, XR, YT, ANGLE)

COMMON /DEVTYP/ IDEVIC, LSW, LTSW, XYCOORD(4), LUOUT, LPAGE
COMMON/GRFTYP/XNGLE, IRVRSE, CHSIZE(9), ITKWIT, LUDIAG, LUHSET
CHARACTER*1 CHARS(*)
CALL WDCELLNT(CHARS, NC)
IF(NC.LE.0) THEN
  IF(LUDIAG.GT.0) WRITE(LUDIAG,3) (CHARS(I),I=1,132)
  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***********************************************************************

COMMON /DEVTYP/ IDEVIC, LSW, LTSW, XYCOORD(4), LUOUT, LPAGE

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.

REAL IX1, IX2, IY1, IY2, IDX, IDY

IX1 = XYCOORD(1)*X1 + XYCOORD(2)
IX2 = XYCOORD(1)*X2 + XYCOORD(2)
IY1 = XYCOORD(3)*Y1 + XYCOORD(4)
IY2 = XYCOORD(3)*Y2 + XYCOORD(4)
IDX = IX2 - IX1
IDY = IY2 - IY1
CALL PUTLNV (IX1, IY1, IDX, IDY, 1)
RETURN
END
SUBROUTINE LINES(X1,Y1,X2,Y2,NL)

THIS SUBROUTINE PLOTS LINES FROM (X1(J),Y1(J)) TO (X2(J),Y2(J))
WHILE CONVERTING TO RASTER NUMBERS THROUGH SCALE.

C***********************************************************************
C LINES
C***********************************************************************
PARAMETER (NLINES=200)
COMMON /DEVTYP/ IDEVIC,LSW,LSW4,XYCOORD(4),LOUT,LPAGE
COMMON/GRFTYP/ANGLE,IRVERSE,CHSIZE(9),ITKWIN,LUDIAG,LUHSET
REAL X1(NL),X2(NL),Y1(NL),Y2(NL)
REAL IX1(NLINES), IY1(NLINES), IDX(NLINES), IDY(NLINES)

NLM = NL
II = 1
I2 = MIN0(I2-1+NLINES,NL)
NLMIN = I2 - II + 1
DO 10 II = I1, I2
IX1(II-I1+1)=XYCOORD(1)*X1(II)+XYCOORD(2)
IDX(II-I1+1)=XYCOORD(1)*X2(II)+XYCOORD(2)-IX1(II-I1+1)
IY1(II-I1+1)=XYCOORD(3)*Y1(II)+XYCOORD(4)
IDY(II-I1+1)=XYCOORD(3)*Y2(II)+XYCOORD(4)-IY1(II-I1+1)
10 CALL PUTFNV(IX1,IX1,IDX,IDY,NLMIN)
IF (I2.GE.NL) RETURN
GO TO 5
RETURN
END
SUBROUTINE LNplot (XARRAY, YARRAY, I1, I2, I3)
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 = MINO (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)
C
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
INITIALIZE THE COUNTERS THE FIRST TIME THRU THIS ROUTINE
C
IF(FIRSTC) THEN
NV = 0
NEX = 0
NPX = 0
FIRSTC = .FALSE.
ENDIF
C
OPEN AND INPUT A STRUCTURES FILE
C
NV = NVM
NE = NEM
NP = NPM
CALL PLYPLT (MAPFIL,0)
CALL PLYGNS(WIN, XYZ(1,NV+1), NV, NE(1,NEX+1), NES(1,NEX+1),
. NE, NP(1,NPX+1), NPS(1,NPX+1), NP)
IF(NV.LE.0) RETURN
C
UPDATE THE EDGE AND POLYGON POINTERS
DO 132 I = NEX+1, NEX+1+NE
DO 132 J = 1, 2
132 NE(J,I) = NE(J,I) + NV
DO 133 I = NPX+1, NPX+1+NP
DO 133 J = 1, 8
133 IF(NP(J,I).GT.0) NP(J,I) = NP(J,I) + NV
133 CONTINUE
NV = NV + NV
NEX = NEX + NE
NPX = NPX + NP
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/DEVTP/IDEVIC,LSW,LTSW,XYCOOD(4),LUOUT,LPAGE
COMMON/GRFMOD/LFLMAT(5),LINWDM(5),LCLMAT(5)
DIMENSION X(N), Y(N), XL(2*NPMAX)

IF(IDEVIC.EQ.0) RETURN
NL = MINO(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
31 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 GMOVAX(XL(1), XL(2))
CALL GPOLAX(XL(3), NL-1)
RETURN
END
SUBROUTINE PLYP1I(CHOICE,ENDSTR,SSTART,SFIRST,SLAST,SVVALID,
IBLANK)


VARIABLES USED:

CFIND = POSITION OF 1ST COMMA FOUND (IF 'BLANK' OPTION SET)

CHOOSE = INTEGER CONTAINING CHARACTER IN THE ENTIRE STRING

LBLANK = 'BLANK' FLAG

(Sinput) = 0 - BLANKS NOT VALID SUBSTRING

(Sinput) = 1 - BLANKS FOLLOWED BY 1 COMMA - VALID SUBSTRING

(Sinput) = 2 - BLANKS FOLLOWED BY 2 COMMAS - VALID SUBSTRING

(Soutput) = -1,-2 - SPECIFIED BLANKS (1 OR 2) FOUND

SFIRST = POSITION OF FIRST CHARACTER OF SUBSTRING

SLAST = POSITION OF LAST CHARACTER OF SUBSTRING

SSTART = STARTING POSITION FOR SUBSTRING SEARCH

STRING = CHARACTER ARRAY CONTAINING THE CHARACTER STRING

SVVALID = 'VALID SUBSTRING' FLAG

= TRUE - VALID SUBSTRING FOUND

= FALSE - NO VALID SUBSTRING FOUND

LOGICAL SVVALID

INTEGER SFIRST,SLAST,SSTART,ENDSTR

INTEGER IBLANK,CFIND,CHOOSE(20),STR(20)

CHARACTER*1 STRING(80),SPACE/' '/,CR/ZD/,,COMMA/Z2C/

EQUIVALENCE (STR,STRING)

ASSUME VALID SUBSTRING

SVVALID=.TRUE.

FORM AN EQUIVALENT CHARACTER STRING

DO 10 I=1,20

STR(I)=CHOOSE(I)

INVALID STARTING POSITION - PAST END OF STRING

IF(ENDSTR.LT.SSTART) GO TO 40

FIND POSITION OF FIRST ELEMENT OF SUBSTRING

CFIND=0

DO 20 I=SSTART,ENDSTR

SFIRST=I

20 CONTINUE

10 CONTINUE

A COMMA FOUND

IF (IBLANK.NE.0.AND.STRING(I).EQ.COMMA) THEN
    FOUND FIRST COMMA
    IF (CFIND.EQ.0) THEN
        CFIND=I
        IF (IBLANK.EQ.2) GO TO 20
    ENDIF
    FOUND BLANK SUBSTRING; NOTE POSITIONS OF DELIMITERS (COMMAS)
    SLAST=SFIRST
    SFIRST=CFIND
    IBLANK=-IBLANK
    GO TO 100
ENDIF

FOUND FIRST CHARACTER OF SUBSTRING - NOW FIND LAST

   STRING(I).NE.COMMA)) GO TO 60
CONTINUE

NO SUBSTRING FOUND - ONLY DELIMITER

WRITE(6,998)
FORMAT('REACHED THE END WITHOUT FINDING A NON-BLANK CHARACTER')
GO TO 40

FIND POSITION OF LAST CHARACTER OF SUBSTRING

IF (SFIRST.EQ.ENDSTR) GO TO 45
DO 50 J=SFIRST+1,ENDSTR
SLAST=J-1
CONTINUE

FOUND SUBSTRING DELIMITER - CAN RETURN NOW

IF((STRING(J).EQ.SPACE).OR.(STRING(J).EQ.COMMA)) GO TO 100
CONTINUE

NO SUBSTRING DELIMITER => LAST CHARACTER OF SUBSTRING IS THE
LAST CHARACTER OF THE STRING

SLAST=ENDSTR
GO TO 100

NO SUBSTRING FOUND

SVALID=.FALSE.
RETURN
SUBROUTINE PLYPL2(WINDOW,OPENN,WORLD)

READ WINDOW COORDINATES AND STORE THEM IN 'WINDOW' ARRAY. ALSO
CHECK VALIDITY OF COORDINATES. IF VALID, 'OPENN' IS TRUE;
OTHERWISE, 'OPENN' IS FALSE. WINDOW COORDINATES WILL BE STORED
AS FOLLOWS:

INDEX / WINDOW(INDEX)
----- ------------
 1   left
 2   bottom
 3   right
 4   top

LOGICAL OPENN
REAL WINDOW(4),WORLD(6)

INITIALIZE
OPENN= .FALSE. ;ASSUME INVALID COORDINATES

CHECK VALIDITY OF WINDOW COORDINATES

DO 30 I=1,2
IF(WINDOW(I).GE.WINDOW(I+2)) RETURN ;INVALID COORDINATES
30 CONTINUE

WINDOW COORDINATES VALID
OPENN= .TRUE.
RETURN •
END
SUBROUTINE PLYPL3(ARRAY,NARRAY,SP,NPNT,NVERT)

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,WGX,WGY,FANGLE,TANAL
COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYGOOD(4),LUOUT,LPAGE

C DEFINE THE WINDOW SPACE
DO 1 I = 1, 4
   SSP(I) = XYGOOD(I)
   CALL DEFINE(SP(1),SP(2),SP(3),SP(4))
1

C PLOT 1 VERTEX AND NUMBER AT A TIME
DO 20 I=1,NARRAY
   DO NOT PLOT A 'DELETED' VERTEX
   IF(NPNT(I).EQ.0) GO TO 20
   C ADD PERSPECTIVE
   D = DOZ/((DOZ+ARRAY(3,I))*TANAL)
   XX = (ARRAY(1,I)-WGX)*D + WXG
   YY = (ARRAY(2,I)-WGX)*D + WGY
   C DO NOT PLOT A VERTEX WHICH IS OUTSIDE OF WINDOW
   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 PLOT THE VERTEX USING THE CHARACTER MODE OF THE HARDWARE
   CALL SYMBOL(XX,YY,CHARAC)
   C SCALE MULTIPLIERS FOR NUMBERS
   XINC=0.025*(SP(3)-SP(1))
   NX=0.035*(SP(3)-SP(1))
   NY=0.035*(SP(4)-SP(2))
   NUMBR=I
   C DETERMINE NUMBER OF DIGITS 'NUM' (THE VERTEX LINE NUMBER)
   NUM=I
   DIGITS=0
   10 NUM=NUM/10.0
   DIGITS=DIGITS+1
   IF (NUM.GE.1) GO TO 10
   STOP
SUBROUTINE FLYPL3

2123 C CALCULATE X COORDINATE OF FIRST DIGIT OF THE NUMBER
2124 C
2125 IF (DIGITS.EQ.1) THEN
2126     XSTART=XX
2127 ELSE IF (MOD(DIGITS,2).EQ.0) THEN
2128     XSTART=XX-(DIGITS*XINC)/4
2129 ELSE
2130     XSTART=XX-(DIGITS*XINC)/2
2131 ENDIF
2132 ENDIF
2133
2134 C PLOT THE NUMBER
2135 C
2136 CALL NUMBR(XSTART,YY-NY,XINC,0.0,NX,0.0,
2137 * NY,NUMBR,DIGITS,0) ,PRINT VERTEX NUMBER
2138 20 CONTINUE
2139 C
2140 C RESTORE THE COORDINATE SYSTEM
2141 C
2142 C DO 2 I = 1, 4
2143 2 XYCOORD(I) = SSP(I)
2144 RETURN
2145 END
SUBROUTINE PLYPL4(ARRAY,NARRAY,VERTX,SP,SPEC)

C THIS SUBROUTINE PLOTS EDGES WHICH ARE WITHIN THE WINDOW OR
C EDGE SEGMENTS WHICH CROSS A PORTION OF THE WINDOW.

LOGICAL VALID
INTEGER ARRAY(2,NARRAY), SPEC(4,NARRAY)
REAL VERTX(3,NARRAY),SP(4),SSP(4),U1,V1,U2,V2
COMMON/PLSCMR/DOZ,WGX,WGY,FANGLE,TANAL
COMMON/DEV_TYP/IDevice,LSW,LTIR,YCOORD(4),LUOUT,LPAGE

C DEFINE THE WINDOW SPACE

DO 1 I = 1, 4
SSP(I) = XYCOORD(I)
CALL DEFINE(SP(1),SP(2),SP(3),SP(4))

C PLOT ONE EDGE AT A TIME

LWD = -1
LCL = -1
DO 150 J=1,NARRAY
C ADD PERSPECTIVE

D = DOZ/((DOZ+VERTX(3,ARRAY(1,J)))*TANAL)
U1=(VERTX(1,ARRAY(1,J))-WGX)*D+WGX
V1=(VERTX(2,ARRAY(1,J))-WGY)*D+WGY
D = DOZ/((DOZ+VERTX(3,ARRAY(2,J)))*TANAL)
U2=(VERTX(1,ARRAY(2,J))-WGX)*D+WGX
V2=(VERTX(2,ARRAY(2,J))-WGY)*D+WGY

C CLIP LEFT EDGE

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

V2 = (V2-V1)*U2/(U1-U2)+V2
U2 = SP(1)

C CLIP RIGHT EDGE

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

V1 = (V1-V2)*(SP(3)-U2)/(U1-U2)+V2
U1 = SP(3)

C CLIP BOTTOM EDGE

GO TO 70
SUBROUTINE PLYPL4

2202 C
2203 70 IF (V1.GE.SP(2).AND.V2.GE.SP(2)) GO TO 90
2204 IF (V1.LT.SP(2).AND.V2.LT.SP(2)) GO TO 150
2205 IF (V1.GT.SP(2)) GO TO 80
2206 U1 = (U1-U2)*V1/(V2-V1)+U1
2207 V1 = SP(2)
2208 GO TO 90
2209 80 U2 = (U2-U1)*V2/(V1-V2)+U2
2210 V2 = SP(2)
2211 C
2212 C CLIP TOP EDGE
2213 C
2214 90 IF (V1.LE.SP(4).AND.V2.LE.SP(4)) GO TO 110
2215 IF (V1.GT.SP(4).AND.V2.GT.SP(4)) GO TO 150
2216 IF (V1.GT.SP(4)) GO TO 100
2217 U2 = (U2-U1)*(SP(4)-V1)/(V2-V1)+U1
2218 V2 = SP(4)
2219 GO TO 110
2220 100 U1 = (U1-U2)*(SP(4)-V2)/(V1-V2)+U2
2221 V1 = SP(4)
2222 C
2223 C PLOT THE EDGE
2224 C
2225 110 IF(LWD.NE.SPEC(1,J)) THEN
2226    LWD = SPEC(1,J)
2227    CALL LINWID(LWD)
2228 ENDIF
2229 IF(LCL.NE.SPEC(2,J)) THEN
2230    LCL = SPEC(2,J)
2231    CALL COLOR(LCL)
2232 ENDIF
2233 CALL LINE(U1, V1, U2, V2)
2234 150 CONTINUE
2235 C
2236 C RESTORE THE ORIGINAL COORDINATE SYSTEM
2237 C
2238 DO 2 I = 1, 4
2239 2 XCOOD(I) = SSP(I)
2240 RETURN
2241 END
SUBROUTINE PLYPL5

SUBROUTINE PLYPL5(ARRAY,NARRAY,VERTX,SP,SPEC,NPVERT)

C THIS SUBROUTINE GRAPHWS POLYGONS WHICH ARE WITHIN THE WINDOW OR
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 INVALID, 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,LSW,XYCOOD(4),LUOUT,LPAGE

C DEFINE WINDOW SPACE
DO 1 I = 1, 4
SSP(I) = XYCOOD(I)
CALL DEFINE(SP(1),SP(2),SP(3),SP(4))

C PLOT POLYGONS ONE AT A TIME
LWD = -1
NDX = -1
LFL = -1
DO 10 I = 1, NARRAY
NUM=0
C
DO 15 J=1,NPVERT
IF(ARRAY(J,I).NE.0) NUM=NUM+1 ,NUMBER OF VERTICES IN POLYGON
 CONTINUE
C
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
 CONTINUE
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)
ENDIF

C

END
CALL LINES(X(1), Y(1), X(2), Y(2), NUM-1)
ENDIF
CONTINUE ;GET NEXT POLYGON
C
RESTORE THE COORDINATE SYSTEM
C
DO 2 I = 1, 4
XYCOORD(I) = SSP(I)
RETURN
END
SUBROUTINE PLYPLT(FILE,OPTION)
$INCLUDE BUILD.COM
PARAMETER (NPART=20)
LOGICAL  EXIST, VALID
INTEGER  NVERTO, NELMTO, NEDGE, NPOLY, FIRST, LAST, IOS
INTEGER  EDGS(NSPEC,NEDGE), POLS(NSPEC,NPOLY), SELECT
INTEGER  FMN(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 FILENM, OLDFIL, FILE
CHARACTER*20 NOREC, BLANK
DATA      VALID/.FALSE./, BLANK/ ' /, LUNIT/9/, OLDFIL/' /
DATA      IUNIT/5/, ZERO/0/
EQUIVALENCE (FILEMN,FN)
COMMON/PLSCMR/DOZ,WX,WCY,FANGLE,TANAL
COMMON/GRFTYP/ANGLE,IRVRES,CHSIZE(9),ITKWIT,LUDIAG,LUHSET
C******************************************************************
DATA      FILENM=FILE
IF(FILEMN.EQ,OLDFIL) GO TO 1100
DO 10 I = FIRST, LAST
J = I - FIRST + 1
10  FILENM(J:J) = FILENM(I:I)
FILE EXIST?
INQUIRE(FILE=FILENAME,IOSTAT=IOS,ERR=91,EXIST=EXIST)
IF(.NOT.EXIST) GO TO 93
OPEN(UNIT=LUNIT,IOSTAT=IOS,ERR=94,FILE=FILENAME,STATUS='OLD')
REWIND LUNIT
GET(I) READ DATA INTO VARIABLES AND ARRAYS
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-NEDGE
SUBROUTINE PLYPLT

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
READ(LUNIT,FMT=1530,ERR=98)

IF (NELMT0.LE.0) GO TO 1090

C INITIALIZE EDGE AND POLYGON COUNTERS
NE=0
NP=0
DO 1080 J=l,NELMTO
READ(LUNIT,FMT=1525,ERR=98) (JP(I),I=1,NPVERT), (ISPEC(I),I=1,
1 NSPEC)
COUNT NUMBER OF VERTICES IN ELEMENT
NCON=O
DO 1030 I=1,NP'7ERT
IF (JP(I).EQ.O) GO TO 1035
NCON=NCON+1
ELEMENT IS EDGE
IF (NCON.EQ.2) THEN
NE=NE+1
DO 1040 I=1,2
EDGE(I,NE)=JP(I)
DO 1050 I=1,NSPEC
ESPEC(I,NE)=ISPEC(I)
ELSE
NP=NP+1
DO 1060 I=1,NCON
POLY(I,NP)=JP(I)
IF (NCON.LT.NPVERT) THEN
DO 1065 I=NCON+1,NPVERT
POLY(I,NP)=0
ENDIF
DO 1070 I=1,NSPEC
PSPEC(I,NP)=ISPEC(I)
ENDIF
CONTINUE

CLOSE(LUNIT)
*************
***DISPLAY
*************
C**SPECIFY WINDOW**
IF (NVERTO.NE.0) THEN
CALL PLYPL2(WINDOW,VALID,WORLD)
ELSE
NOREC = 'NO RECORDS'
END
SUBROUTINE PLYPLT

2422 IF(LUDIAG.GT.0) WRITE(LUDIAG,108) NOREC
2423 RETURN
2424 C
2425 IF VALID WINDOW, SET 'SPACE' COORDINATES TO 'WINDOW', OTHERWISE
2426 SET THE 'SPACE' COORDINATES TO 'WORLD'
2427 C
2428 C
2429 IF(V) THEN ; VALID WINDOW
2430 DO 40 I=1,4
2431 SPACE(I)=WINDOW(I)
2432 ELSE
2433 DO 41 I=1,2
2434 SPACE(I)=WORLD(I)
2435 DO 42 I=3,4
2436 SPACE(I)=WORLD(I+1)
2437 ENDIF
2438 C
2439 **FIND CENTER OF WINDOW**
2440 WCX=(SPACE(3)+SPACE(1))/2
2441 WCY=(SPACE(4)+SPACE(2))/2
2442 C
2443 **INITIALIZE FIELD OF VIEW**
2444 DOZ=10000.
2445 FANGLE=90.
2446 TANAL=1.0
2447 C
2448 INITIALIZE AND SET UP FOR A SINGLE GRAPH - 'NOREG', IF SET, WILL
2449 INDICATE THAT THERE ARE NO RECORDS TO BE DISPLAYED
2450 C
2451 NOREC=BLANK
2452 C
2453 IF (OPTION.EQ.0) THEN ; NO PLOTS - READ IN DATA FILE ONLY
2454 RETURN
2455 ELSE IF (OPTION.EQ.1) THEN
2456 IF (NEDGEO.EQ.0) NOREC='NO EDGES'
2457 ELSE IF (OPTION.EQ.2) THEN
2458 IF (NVERTO.EQ.0) NOREC='NO VERTICES'
2459 ELSE IF (OPTION.EQ.3) THEN
2460 IF (NPOLYO.EQ.0) NOREC='NO POLYGONS'
2461 ELSE IF (OPTION.EQ.4) THEN
2462 IF (NEDGEO.EQ.0 .AND. NPOLYO.EQ.0) NOREC='NO EDGES OR POLYGONS'
2463 ELSE
2464 NOREC='INVALID OPTION'
2465 IF(LUDIAG.GT.0) WRITE(LUDIAG,108) NOREC
2466 RETURN
2467 ENDIF
2468 C
2469 **PLOT**
2470 C
2471 IF (NOREC.EQ.BLANK) THEN
2472 IF (OPTION.EQ.1) THEN
2473 CALL PLYPL4(EDGE,NEDGEO,VERTEX,SPACE,ESPEC)
ELSE IF (OPTION.EQ.2) THEN
   CALL PLYPL3(VERTEX,NVERT,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 A ROUTINE TO RETURN THE COORDINATE ARRAY VALUES FROM A DATA FILE
C***************************************************************************
ENTRY PLYGNS(WIN,VER,NVM,EDG,EDGS,NEM,EDG,EDGS,NEM,NPOL,POLS,NPM)
NVM = MIN(NVERT,NVM,NVERT0)
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)
88 CONTINUE
EDGS(J,I) = ESPEC(J,I)
DO 84 J = 1, NSPEC
84 CONTINUE
NPM = MIN(NPOLY,NPM, NP)
DO 85 I = 1, NPM
DO 86 J = 1, NPVERT
86 POL(J,I) = POLY(J,I)
85 CONTINUE
POLS(J,I) = PSPEC(J,I)
87 POLS(J,I) = PSPEC(J,I)
88 CONTINUE
RETURN
C ERRORS
IF(LUDIAG.GT.0) WRITE(LUDIAG,100) FILE
RETURN
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
SUBROUTINE PLYPLT

RETURN
2533 93 IF(LUDIAG.GT.0) WRITE(LUDIAG,103)
2534 RETURN
2535 94 CLOSE(UNIT=LUNIT, IOSTAT=IOS, ERR=95)
2536 95 IF(LUDIAG.GT.0) WRITE(LUDAIG,104) IOS
2537 RETURN
2538 98 IF(LUDIAG.GT.0) WRITE(LUDIAG,110)
2539 RETURN
2540 C
2541 C FORMATS
2542 C
2543 100 FORMAT( 'INVALID FILE DESCRIPTOR',2X,A17)
2544 101 FORMAT( 'NO RECORDS IN FILE')
2545 102 FORMAT( 'INQUIRE ERROR = ',14)
2546 103 FORMAT( 'FILE DOES NOT EXIST')
2547 104 FORMAT( 'FILE ERROR',14)
2548 107 FORMAT( 'WINDOW DEFAULTS TO "WORLD"')
2549 108 FORMAT(1X,A20,/,/) FORMAT( 'ERROR - DATA FORMAT PROBLEMS -- ENTER NEW COMMAND')
2550 110 FORMAT( 'ERROR - DATA FORMAT PROBLEMS -- ENTER NEW COMMAND')
2551 1500 FORMAT(A5,4E12.5)
2552 1510 FORMAT(6E12.5)
2553 1520 FORMAT(16I5)
2554 1525 FORMAT(8I5,5X,8I5)
2555 1530 FORMAT(I5,3E12.5)
2556 END
SUBROUTINE PROPOL (NTRIA, XT1, YT1, FT1, XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)

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 ALP1(NPT), ALP2(NPT), ALP3(NPT), XI(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 XM1,YM1,XM2,YM2,YM3,XM1,YM1,XM2,YM2/4*0.,4*100./

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

PROPOL (NTRIA, XT1, YT1, FT1, XT2, YT2, FT2, XT3, YT3, FT3, FL, NX, NY)

NTRIA INTEGER NUMBER OF TRIANGLES IN LIST
XT1 REAL ARRAY(NTRIA) X POSITIONS OF FIRST VERTICES
YT1 REAL ARRAY(NTRIA) Y POSITIONS OF FIRST VERTICES
FT1 REAL ARRAY(NTRIA) FUNCTION VALUES AT FIRST VERTICES
XT2 REAL ARRAY(NTRIA) X POSITIONS OF SECOND VERTICES
YT2 REAL ARRAY(NTRIA) Y POSITIONS OF SECOND VERTICES
FT2 REAL ARRAY(NTRIA) FUNCTION VALUES AT SECOND VERTICES
XT3 REAL ARRAY(NTRIA) X POSITIONS OF THIRD VERTICES
YT3 REAL ARRAY(NTRIA) Y POSITIONS OF THIRD VERTICES
FT3 REAL ARRAY(NTRIA) FUNCTION VALUES AT THIRD VERTICES
FL REAL VALUE OF F AT WHICH CONTOUR IS DRAWN
NX INTEGER MAXIMUM VALUE OF X POSITIONS
NY INTEGER MAXIMUM VALUE OF Y POSITIONS

PROPOL ASSUMES A CONTOUR CROSSING LIES BETWEEN (XT1,YT1) AND (XT2,YT2). THE VERTICES MUST BE NUMBERED COUNTER-CLOCKWISE AROUND THE TRIANGLE. THE CODE IN PROPOL IS SPECIALIZED TO THE RECTANGULAR X-Y CASE BY THE TRANSFORMATION OF DO LOOP 145. PLOAR PLOTS CAN BE FORMED BY CHANGING THIS TRANSFORMATION.

ENTRY POINTS: CNTSET, CNTFRM (SEE DOCUMENTATION OR LISTING BELOW)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

IF (NTRIA .GT. NPT) NTRIA = NPT
XD = (XM2-XM1) / FLOAT(NX-1)
XB = ((XM2-XM1)-(XM3-XM4)) / FLOAT((NY-1)*(NX-1))
XC = (XM4 - XM1) / FLOAT(NY-1)
YD = (YM4 - YM1) / FLOAT(NX-1)
YB = ((YM4-YM1)-(YM3-YM2)) / FLOAT((NY-1)*(NX-1))
YC = (YM2 - YM1) / FLOAT(NX-1)

-200-
SUBROUTINE PROPOL

2613  C   DO 100 I = 1, NTRIA
2614  DIFF3(I) = FT2(I) - FT1(I)
2615  DIFF1(I) = FT3(I) - FT1(I)
2616  DIFF2(I) = FT3(I) - FT2(I)
2617  SGN3(I) = SIGN (1.0, DIFF3(I))
2618  SGN1(I) = SIGN (1.0, DIFF1(I))
2619  SGN2(I) = SIGN (1.0, DIFF2(I))
2620  DO 110 I = 1, NTRIA
2621  DIFF3(I) = ABS(DIFF3(I)) + DELTA
2622  DIFF1(I) = ABS(DIFF1(I)) + DELTA
2623  DIFF2(I) = ABS(DIFF2(I)) + DELTA
2624  DFL3(I) = SGN3(I)*FL - FT1(I)
2625  DFL1(I) = SGN1(I)*FL - FT1(I)
2626  DFL2(I) = SGN2(I)*FL - FT2(I)
2627  DO 120 I = 1, NTRIA
2628  ALF3(I) = DFL3(I)/DIFF3(I)
2629  ALF1(I) = DFL1(I)/DIFF1(I)
2630  ALF2(I) = DFL2(I)/DIFF2(I)
2631  X3(I) = XT1(I) + ALF3(I)*(XT2(I) - XT1(I))
2632  Y3(I) = YT1(I) + ALF3(I)*(YT2(I) - YT1(I))
2633  X1(I) = XT1(I) + ALF1(I)*(XT3(I) - XT1(I))
2634  Y1(I) = YT1(I) + ALF1(I)*(YT3(I) - YT1(I))
2635  X2(I) = XT2(I) + ALF2(I)*(XT3(I) - XT2(I))
2636  Y2(I) = YT2(I) + ALF2(I)*(YT3(I) - YT2(I))
2637  DO 130 I = 1, NTRIA
2638  IF TEST(I) IS LESS THAN ZERO, THEN MISS, OTHERWISE HIT.
2639  DO 130 I = 1, NTRIA
2640  X2(I) = AND (RMASK(I), X2(I))
2641  Y2(I) = AND (RMASK(I), Y2(I))
2642  X1(I) = AND (RMASK(I), X1(I))
2643  Y1(I) = AND (RMASK(I), Y1(I))
2644  X1(I) = X1(I) + X2(I)
2645  Y1(I) = Y1(I) + Y2(I)
2646  DO 140 I = 1, NTRIA
2647  X1(I) = X1(I) + X2(I)
2648  Y1(I) = Y1(I) + Y2(I)
2649  X1(I) = X1(I) + X2(I)
2650  Y1(I) = Y1(I) + Y2(I)
2651  DO 150 I = 1, NTRIA
2652  X3(I) = AMIN1(XMAX, AMAX1(XMIN, X3(I)))
2653  Y3(I) = AMIN1(YMAX, AMAX1(YMIN, Y3(I)))
2654  CALL LINES (X3, Y3, X1, Y1, NTRIA)
2655  NTRIA = 0
ENTRY CNTSET (XM1, YM1, XM2, YM2, XM3, YM3, XM4, YM4)

DESCRIPTION: CNTSET MAY BE CALLED BY THE USER TO MOVE THE CONTOUR PLOT GENERATED AROUND ON THE PLOTTING REGION OR TO STRETCH OR COMPRESS THE PLOT. THE DATA STATEMENT GIVES THE DEFAULT FOR A LARGE SQUARE PLOT. THE PLOT WILL EXTEND FROM XM1 TO XM4 IN THE HORIZONTAL AND FROM YM1 TO YM4 IN THE VERTICAL. ALL FOUR OF THESE VALUES SHOULD BE IN THE RANGE 1 TO 1023.

ENTRY CNTFRM

DESCRIPTION: CNTFRM IS A USER-CALLED ROUTINE TO PLOT THE RECTANGULAR BOUNDARY OF THE CONTOURED REGION. CNTFRM HAS NO ARGUMENTS SINCE THE REGION IS SPECIFIED BY DEFAULT OR VIA A PREVIOUS CALL TO CNTSET.
SUBROUTINE PUTCH (X, Y, CHARAC)

INTEGER LPAGE(30, 65)
COMMON/DEVTYP/IDEVIC,LSW,LTSW,XYCOORD(4),LUOUT,LPAGE
COMMON/GRFTYP/ANGLE,IRVRSE,CHSIZE(9),ITKWIN,LUDIAG,LHSET
CHARACTER*1 CHARAC, CH(4)
EQUIVALENCE (CH,ICH)
INTEGER K, K1, M, IK, IM, MASK(5)
INTEGER LSHFT, AND, OR
LOGICAL LSW, LTSW
DATA MASK/ZFFFFFOOO, ZFFOOFFFF, Z00FFFFF, ZFF0000F00,
     ZFF000000/, CHSZ/0.0/
ICH = 0
CH(1) = CHARAC
CHS = CHSIZE(2)
GO TO (1,2,3,5), IDEVIC

CALCOMP SECTION.

CALL CALSYM (X, Y, CHS, CH, 0, 0.0, 1)
LSW = .TRUE.
RETURN

PAGE PLOT SECTION.

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 = IANDdCH, 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

LEXIDATA/MATRIX SECTION

CALL GSCHSZ (CHS)
XL = MAX(X,0.)
YL = MAX(Y,0.)
CALL QMOVA (XL, YL)
CALL GTXTWC (CH, 1)
RETURN

TEKTRONIX SECTION

CONTINUE
CALL MOVEA(X,Y)
SUBROUTINE PUTC

2775 CALL A1OUT(1,ICH)
2776 LTSW=.TRUE.
2777 RETURN
2778 END
SUBROUTINE PUTLNV (X1, Y1, DX, DY, NL)

PARAMETER (N_LINES = 200)

INTEGER LENGTH, COUNT
REAL XI(NL), Y1(NL), DX(NL), DY(NL)
INTEGER MASK1 / 15/, 0P11 / 12/ , IVERT / 023/
INTEGER LPAGE(30, 65), HSYMBL, HDOT, HAPOS, HMINUS
REAL JX1(NLINES), JY1(NLINES), JX2(NLINES), JY2(NLINES)
REAL JX1, JY1, JX2, JY2

INTEGER MASK1 / 15/, 0P11 / 12/ , IVERT / 023/
INTEGER LPAGE(30, 65), HSYMBL, HDOT, HAPOS, HMINUS
REAL JX1(NLINES), JY1(NLINES), JX2(NLINES), JY2(NLINES)
REAL JX1, JY1, JX2, JY2

common/DEVTYP/ I_DEVIC, LSW, LSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
common/GRFTYP/ ANGLE, XDEVIG, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
IF(DY(I).GT.2*DX(I)) HSYM=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 = HSYM
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(HSYM, M1)
LPAGE(K+4,IY) = IOR(IM, IK)

CONTINUE
RETURN

LEXIDATA/MATRIX SECTION

CALL GMOVA(JXi(IL),JY1(IL))
XY(1,IS) = JX2(IL)
XY(2,IS) = JY2(IL)

IF(JXi(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

CALL GPLNA(XY,IS)
IF(IL.GT.NLP) RETURN
GO TO 32
RETURN

TEKTRONIX SECTION

DO 40 I=1,NLP
IF(LTWS) GO TO 42
IF(JXS.EQ.JX1(I).AND.JYS.EQ.JY1(I)) GO TO 45

CALL MOVEA(JX1(I),JY1(I))
LTWS=.FALSE.

CALL DRAWA(JX2(I),JY2(I))
JXS=JX2(I)
JYS=JY2(I)
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, 8, 1, 3, 5, 7, 9, 11, 13,
    0, 15, 0, 0, 15, 0, 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)

THIS SUBROUTINE AND ITS ASSOCIATED ENTRIES CONSTRUCT AND
DELIVER PLOTS OF A SURFACE Z(I,J) WITH HIDDEN LINES REMOVED
FOR I = 1, ..., NX AND J = 1, ..., NY. THE TYPE, ORIENTATION,
AND DETAILS OF THE PLOTS ARE QUITE FLEXIBLE AS SEEN BY CAREFUL
STUDY OF THE TEST PROGRAM AND THE RESULTING OUTPUT
THE PARTICULAR VERSION IS AN INTERFACE TO THE GENERAL PURPOSE
PLOTTING PACKAGE WHICH CAN DRAW COLOR PLOTS ON THE CALCOMP,
LEXIDATA, PRINTER OR TEKTRONICS (4000'S SERIES).

SURFACE Initializes THE SURFACE PLOTTING PACKAGE AND MUST
BE CALLED EACH TIME A NEW PLOT IS DESIRED TO RESET THE
HIDDEN LINE ARRAYS.

NOTE THAT MAX(NX,NY) MUST BE LESS THAN NPT

THE LOGICAL PLOTTING REGION IS A 3D RECTANGULAR PARALLELEPIPED WITH 8 CORNER VERTICES.

XX REAL  - ARRAY DIMENSIONED 8 CONTAINING THE 8 CORNER
          VERTEX X LOCATIONS. THE VERTICES ARE NUMBERED
          AS SHOWN ON THE SURFACE GEOMETRY SHEET (AVAILABL
          FROM THE SPL LIBRARIAN OR THE AUTHOR).

YY REAL  - ARRAY DIMENSIONED 8 CONTAINING THE 8 CORNER
          VERTEX Y LOCATIONS. THE VERTICES ARE NUMBERED
          AS SHOWN ON THE SURFACE GEOMETRY SHEET WHICH IS
          AVAILABLE FROM THE SPL LIBRARIAN OR THE AUTHOR.

ZZMIN REAL  - THE VALUE OF Z(I,J) TO BE PLOTTED AT THE
              BOTTOM SURFACE OF THE PARALLELEPIPED.

ZZMAX REAL  - THE VALUE OF Z(I,J) TO BE PLOTTED AT THE
              UPPER SURFACE OF THE PARALLELEPIPED.

NX INTEGER  - DIMENSION AND RANGE OF I IN Z(I,J)

NY INTEGER  - DIMENSION AND RANGE OF J IN Z(I,J)

SFRAME (MODE1) - PLOTS THE FRAME OF THE RECTANGULAR PARALLELEPIPED PLOTTING REGION.

MODE1 INTEGER - MODE1 = 1 PLOTS + AT THE VERTICES
                 = 2 ALSO PLOTS LINE SEGMENTS

CONNECTING THE VERTICES.

SSKIRT (ZZ, NX, NY, MODE2) - PLOTS ANY OF THE SKIRTS WHICH MAY
BE DESIRED. ONLY ONE SKIRT IS PLOTTED PER CALL AND THE
HIDDEN LINE ALGORITHM IS INVOKED. THEREFORE, THE SIDE
SKIRTS (+2, -2, +4, -4) AND THE DATA SURFACE BACK SKIRTS
(+3, -3) SHOULD ONLY BE PLOTTED AFTER THE DATA SURFACE
HAS BEEN CONSTRUCTED USING SURFACE. THIS ENTRY DOES NOT
PLOT THE Z SURFACE (HERE THE DATA ARE CALLED ZZ TO AVOID
DECLARATION CONFLICTS). HOWEVER Z(I,J) ARE NEEDED TO
DEFINE THE SKIRT POSITIONS.

ZZ REAL  - ARRAY CONTAINING THE DATA TO BE PLOTTED AS A
SURFACE

NX INTEGER  - DIMENSION AND RANGE OF I IN Z(I,J)
SUBROUTINE SRFSET

INTEGER - DIMENSION AND RANGE OF J IN Z(I,J)

MODE2 INTEGER - MODE2 = 1(-1) LOWER (UPPER) FRONT SKIRT
MODE2 = 2(-2) LOWER (UPPER) RIGHT SKIRT
MODE2 = 3(-3) LOWER (UPPER) BACK SKIRT
MODE2 = 4(-4) LOWER (UPPER) LEFT SKIRT

SURFAC (Z, NX, NY, MODE3) - CONSTRUCTS AND PLOTS THE DATA SURFACE AND SETS THE HIDDEN LINE ARRAYS WHICH ARE USED IN SURFSK.

Z REAL - ARRAY DIMENSIONED (NX,NY) CONTAINING THE DATA TO BE PLOTTED AS A SURFACE

NX INTEGER - DIMENSION AND RANGE OF I IN Z(I,J)

NY INTEGER - DIMENSION AND RANGE OF J IN Z(I,J)

MODE3 INTEGER - MODE3 = 1 PLOTS THE UPPER SURFACE
MODE3 = -1 PLOTS THE LOWER SURFACE

PARAMETER (NPT=101)

LOGICAL SW(1024)

REAL XX(8), YY(8)

REAL X(8), Y(8), H(1280), G(1280)

REAL AJ1(NPT), AJN(NPT), AII(NPT), AIN(NPT), ZVAL(NPT)

REAL RDZVB(NPT), RDZTV(NPT)

LOGICAL SWITCH

REAL Z(NX,NY), XI(NPT), X2(NPT), Y1(NPT), Y2(NPT)

REAL ZZ(NX, NY)

COMMON /SURCMN/ X, Y, ZMIN, ZMAX, RDZ, RNXM1, RNYM1, NNX, NNY

INITIALIZE HIDDEN LINE ARRAYS.

NNX = NX

NNY = NY

IF(MAXO(NX,NY).GT.NPT) STOP 76

YMIN = 1.E+25

YMAX = -1.E+25

DO 3 I = 1, 8

YMIN = AMIN1(YMIN, YY(I))

YMAX = AMAX1(YMAX, YY(I))

DO 1 I = 1, 1280

G(I) = YMAX

H(I) = YMIN

FOR SPECIAL EFFECTS ZBOT AND ZTOP MAY DIFFER FROM ZMIN AND ZMAX.

ZBOT = ZZMIN

ZTOP = ZZMAX

ZMIN = ZZMIN

ZMAX = ZZMAX

RDZ = 1.0/(ZTOP - ZBOT)

DO 2 I = 1, 8

X(I) = XX(I)

Y(I) = YY(I)

RNXM1 = 1.0/FLOAT(NX - 1)

RNYM1 = 1.0/FLOAT(NY - 1)

RETURN
ENTRY SFRAME (MODE1)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

DESCRIPTION: SURFRTM PLOTS THE FRAME OF THE RECTANGULAR PARALLELO-
PIPED PLOTTING REGION.

ARGUMENTS:

MODE1 INTEGER

MODE1 = 1 PLOTS ONLY AT THE VERTICES.

MODE1 = 2 ALSO PLOTS LINE SEGMENTS

CONNECTING THE VERTICES.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

CALL CHPLOT (X, Y, '+', 1, 1, 8)

NOW CHECK THE VARIOUS LINES.

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))
    CALL SURF5 (X(5), Y(5), X(7), Y(7))
IF (Y(8).GT.Y(6) .OR. X(8).GT.X(6))
    CALL SURF5 (X(8), Y(8), X(6), Y(6))
IF (Y(3).LT.Y(1) .OR. X(3).LT.X(1))
    CALL SURF5 (X(1), Y(1), X(3), Y(3))
IF (Y(4).LT.Y(2) .OR. X(4).GT.X(2))
    CALL SURF5 (X(2), Y(2), X(4), Y(4))
IF (Y(7).GE.Y(5) .AND. Y(8).GE.Y(6))
    CALL SURF5 (X(8), Y(8), X(7), Y(7))
IF (X(8).GE.X(6) .AND. X(4).GE.X(2))
    CALL SURF5 (X(8), Y(8), X(4), Y(4))
IF (Y(4).LE.Y(2) .AND. Y(3).LE.Y(1))
    CALL SURF5 (X(4), Y(4), X(3), Y(3))
IF (X(3).LE.X(1) .AND. X(7).LE.X(5))
    CALL SURF5 (X(7), Y(7), X(3), Y(3))
12 CONTINUE
RETURN

ENTRY SSKIRT (ZZ, NX, NY, MODE2)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

DESCRIPTION: SURFSK PLOTS ANY OF THE SKIRTS WHICH MAY BE DESIRED.

ONLY ONE SKIRT IS PLOTTED PER CALL AND THE HIDDEN LINE ALGORITHM

IS INVOKED. THEREFORE, THE SIDE SKIRTS (+2, -2, +4, -4) AND THE
SUBROUTINE SRFSET

3062 C BACK SKIRTS (+3, -3) SHOULD ONLY BE PLOTTED AFTER THE DATA SURFACE
3063 C HAS BEEN CONSTRUCTED USING SURFACE. THIS ENTRY DOES NOT PLOT THE Z
3064 C SURFACE (HERE THE DATA ARE CALLED ZZ TO AVOID DECLARATION CON-
3065 C FLICTS). HOWEVER Z(I,J) ARE NEEDED TO DEFINE THE SKIRT POSITIONS.
3066 C
3067 C ARGUMENTS:
3068 C ZZ REAL ARRAY (NX,NY) THE DATA TO BE PLOTTED AS A SURFACE. I
3069 C NX INTEGER DIMENSION AND RANGE OF I IN Z(I,J). I
3070 C NY INTEGER DIMENSION AND RANGE OF J IN Z(I,J). I
3071 C MODE2 INTEGER
3072 C    MODE2 = 1(-1) LOWER (UPPER) FRONT SKIRT
3073 C    MODE2 = 2(-2) LOWER (UPPER) RIGHT SKIRT
3074 C    MODE2 = 3(-3) LOWER (UPPER) BACK SKIRT
3075 C    MODE2 = 4(-4) LOWER (UPPER) LEFT SKIRT
3076 C
3077 C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3078 C NSKIRT = IABS(MODE2)
3079 C GO TO (21, 22, 23, 22), NSKIRT
3080 C
3081 C THE FRONT SKIRT IS ADDED.
3082 C
3083 C 21 CONTINUE
3084 C XL = X(1)
3085 C XR = X(2)
3086 C YL = Y(1)
3087 C YR = Y(2)
3088 C IF (MODE2 .GT. 0) GO TO 211
3089 C XL = X(5)
3090 C XR = X(6)
3091 C YL = Y(5)
3092 C YR = Y(6)
3093 C 211 DO 212 I = 1, NX
3094 C F11 = FLOAT(I-1)/FLOAT(NX-1)
3095 C FIN = FLOAT(NX-I)/FLOAT(NX-1)
3096 CXA = XR*F11 + XL*FIN
3097 CYA = YR*F11 + YL*FIN
3098 C CALL SURF4 (XB,YB, I,1, ZZ(I,1))
3099 C CALL SURF5 (XA, YA, XB, YB)
3100 C RETURN
3101 C
3102 C THE RIGHT OR LEFT SIDE SKIRT IS ADDED. IF IT IS OBSCURED THE
3103 C HIDDEN LINE ALGORITHMS ARE USED SO CALL ONLY AFTER SURFACE IS USED
3104 C 22 NSK = (4 - NSKIRT)/2
3105 C I = 1 + (NX-1)*NSK
3106 C XST = X(NSK+1)
3107 C YST = Y(NSK+1)
3108 C XND = X(NSK+3)
3109 C YND = Y(NSK+3)
3110 C IF (MODE2 .GT. 0) GO TO 221
3111 C XST = X(NSK+5)
3112 C YST = Y(NSK+5)
3113 C XND = X(NSK+7)
3114 C YND = Y(NSK+7)
3115 C 221 DO 222 J = 1, NY
3116 C PJ1 = FLOAT(J-1)/FLOAT(NY-1)
SUBROUTINE SRFSET

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))

THE SKIRT MAY BE OBSCURED. CHECK ON TOP OR BOTTOM.

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)

CONTINUE
RETURN

THE BACK SKIRT IS ADDED (ASSUMED CALLED AFTER SURFACE).

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)

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)

CONTINUE
RETURN

ENTRY SURFAC (Z, NX, NY, MODE3)

DESCRIPTION: SURFACE CONSTRUCTS AND PLOTS THE DATA SURFACE AND SET
THE HIDDEN LINE ARRAYS WHICH ARE USED IN SURFSK.

ARGUMENTS:
Z REAL ARRAY (NX,NY) THE DATA TO BE PLOTTED AS A SURFACE. I
NX INTEGER DIMENSION AND RANGE OF I IN Z(I,J). I
NY INTEGER DIMENSION AND RANGE OF J IN Z(I,J). I
MODE3 INTEGER MODE3 = 1 PLOTS THE UPPER SURFACE I
SUBROUTINE SRFSET

YA = AMAX1(Y1(I), H(IHX))
YB = AMAX1(Y2(I), H(IHX))
IF (YB.GT.YA) CALL SURF5 (X1(I), YA, X2(I), YB)
GO TO 35

YA = AMIN1(Y1(I), G(IHX))
YB = AMIN1(Y2(I), G(IHX))

IF (YB.LT.YA) CALL SURF5 (X1(I), YA, X2(I), YB)
GO TO 35

THIS IS A SLANTED LINE SEGMENT.

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) .OR. (MODE3.GT.0 .AND. H(K).LE.YK+0.5)) GO TO 36
SWITCH = .FALSE.
SW(K) = .FALSE.
NSWT = NSWT - 1
CONTINUE

IF (SWITCH) CALL SURF5 (X1(I), Y1(I), X2(I), Y2(I))
IF (SWITCH .OR. NSWT.EQ.0) GO TO 35

PART OF THE LINE IS OBSCURED SO WE NEED TO PLOT SEGMENTS.

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)
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)
CALL SURF5 (XA, YA, FLOAT(K), YK)
CONTINUE
CONTINUE
CONTINUE
CONTINUE

PLOT THE I TO I+1 LINE SEGMENTS IF NOT HIDDEN.

DO 33 I = 2, NX
IXB = X2(I) + 0.5
IXA = X2(I-1)
DIX = 1.0/FLOAT(IXB - IXA)
SUBROUTINE SRFSET

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) .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
PART OF THE LINE IS OBSCURED SO WE NEED TO PLOT SEGMENTS.
3300 SW(IXA-1) = .FALSE.
3301 SW(IXB+1) = .FALSE.
3302 DO 44 K = IXA, IXB
3303 YK = Y2(I-1)*(IXB-K)*DIX + Y2(I)*(K-IXA)*DIX
3304 IF (SW(K) .AND. .NOT.SW(K-1)) XA = K
3305 IF (SW(K) .AND. .NOT.SW(K+1)) YA = YK
3306 IF (SW(K) .AND. .NOT.SW(K+1)) CALL SURF5 (XA,YA,FLOAT(K),YK)
3307 44 CONTINUE
3308 33 CONTINUE
3309 C
3310 31 CONTINUE
3311 RETURN
3312 END
SUBROUTINE SURF4 (XVAL, YVAL, I, J, ZIN)
*
** THIS IS AN AUXILIARY ROUTINE TO SURFACE. IT PERFORMS A TRILINEAR INTERPOLATION IN THE 3D RECTANGLE OUTLINED BY (X(I), Y(I)) ON THE PLOTTING SURFACE FOR I = 1, ..., 8. THE TEXT WILL BE SUBSTITUTED IN LINE WHEN THE ASC COMPILER BUG HAS BEEN FIXED. **
*
REAL X(8), Y(8)
COMMON /SURCMN/ X, Y, ZMIN, ZMAX, RDZ, RNXM1, RNYM1, NNX, NNY
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(7)) + FJ1*(FI1*X(8) + FIN*X(7)) + FJ1*(FI1*X(8) + FIN*X(7)))
YVAL = RDZ*(ZVAL-ZMIN)*(FJN*(FI1*Y(6) + FIN*Y(7)) + FJ1*(FI1*Y(8) + FIN*Y(7)) + FJ1*(FI1*Y(8) + FIN*Y(7)))
RETURN
*
** THIS SUBROUTINE IS A GENERAL-PURPOSE GRAPHICS INTERFACE FOR USE WITH THE SURFACE SUBROUTINE FOR 3D SURFACE PLOTS WITH HIDDEN LINES REMOVED. **
*
ENTRY SURF5 (X1, Y1, X2, Y2)
IF ( ((X1-X2)**2 + (Y1-Y2)**2) .GT. 2.25) 1
CALL LINE (X1, Y1, X2, Y2)
RETURN
END
SUBROUTINE SYMBOL (XCHAR, YCHAR, CHARAC)

COMMON /DEVTYP/ IDEVIC, LSW, LTSW, XYCOOD(4), LUOUT, LPAGE
COMMON/GRFTYP/ANGLE, IRVRSE, CHSIZE(9), ITKWIT, LUDIAG, LUHSET
CHARACTER*1 CHARAC

THIS SUBROUTINE USES THE HARDWARE CHARACTERS TO PLOT DATA
POINTS AT THE LOCATION SPECIFIED BY (XCHAR, YCHAR) IN THE PLOTTING
REGION DETERMINED BY XYCOOD.

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 PUTC(H (X, Y, CHARAC)
RETURN
END
SUBROUTINE VIEWTR(XDC,WINDOW,VERTEX,NV,EDGE,ESPEC,NE, 
. POLY,PSPEC,NP)

C THIS SUBROUTINE PLOTS EDGES AND POLYGONS WHICH ARE WITHIN THE WINDOW 
C EDGE SEGMENTS WHICH CROSS A PORTION OF THE WINDOW.

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)

DEFINE THE WINDOW SPACE

CALL DEFINE(WINDOW(1),WINDOW(2),WINDOW(3),WINDOW(4))

DOZ=10000. 
FANGLE=90. 
TANAL = 1.0 
WCX = 0.5 *(WINDOW(1)+WINDOW(3)) 
WCY = 0.5 *(WINDOW(2)+WINDOW(4))

PLOT ONE EDGE AT A TIME

DOZ=150 J=1,NE

MOVE VERTICES INTO A LOCAL WORK ARRAY

DO 151 I = 1, 2 
DO 151 K = 1, 3

XYZ(K,I) = VERTEX(K,EDGE(I,J))

APPLY THE TRANSFORM

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) 
XYZ(3,I) = XDC(3,1)*U1 + XDC(3,2)*U2 + XDC(3,3)*U3 + XDC(3,4)

ADD PERSPECTIVE

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

CLIP LEFT EDGE

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
V2 = (V2-V1)*U2/(U1-U2)+V2
U2 = WINDOW(1)
CLIP RIGHT EDGE
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
V2 = (V2-V1)*(WINDOW(3)-U1)/(U2-U1)+V1
U2 = WINDOW(3)
GO TO 70
V1 = (V1-V2)*(WINDOW(3)-U2)/(U1-U2)+V2
U1 = WINDOW(3)
CLIP BOTTOM EDGE
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
U1 = (U1-U2)*V1/(V2-V1)+U1
V1 = WINDOW(2)
GO TO 90
U2 = (U2-U1)*V2/(V1-V2)+U2
V2 = WINDOW(2)
CLIP TOP EDGE
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
U2 = (U2-U1)*(WINDOW(4)-V1)/(V2-V1)+U1
V2 = WINDOW(4)
GO TO 110
U1 = (U1-U2)*(WINDOW(4)-V2)/(V1-V2)+U2
V1 = WINDOW(4)
PLOT THE EDGE
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)
CONTINUE
PLOT POLYGONS ON TOP OF EDGES
SUBROUTINE VIEWTR

C
DO 10 J = 1, NP
C
NUM=0
Determine the number of vertices (non-zero entries) in the polygon
C
DO 15 I=1,8
IF(POLY(I,J).NE.0) NUM=NUM+1 ;Number of vertices in polygon
CONTINUE
MOVE the vertices into a local work array
DO 61 I = 1, NUM
DO 61 K = 1, 3
XYZ(K,I) = VERT£X(K,POLY(I,J))
C
APPLY the transform
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)
XYZ(3,I) = XDC(3,1)*U1 + XDC(3,2)*U2 + XDC(3,3)*U3 + XDC(3,4)
ADD perspective
DO 60 IV=1,NUM
D = DOZ/((DOZ+XYZ(3,IV))*TANAL)
X(IV) = (XYZ(1,IV) - WCX)*D + WCX
Y(IV) = (XYZ(2,IV) - WCY)*D + WCY
PLOT the polygon
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 FILTPY(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
CONTINUE ;Get next polygon
RETURN
SUBROUTINE VIEWTR

END

C
SUBROUTINE VOLFRM (MODE1)

C THIS ENTRY PUTS A FRAME AROUND THE REGION IF DESIRED.
C FIRST PLOT + AT ALL OF THE VERTICES.

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 NOW CHECK THE VARIOUS LINES.

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

END
SUBROUTINE VOLUMO (Q, R, NX, NY, NZ)

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)
   Y(I) = R(I)
1 RETURN

ENTRY VOLUM6 (XA, YA, XG, YG, ZG)

DII = (FNX - XG)*RNX
DJ1 = (FNY - YG)*RNY
DK1 = (FNZ - ZG)*RNZ

DIN = 1.0 - DII
DJN = 1.0 - DJ1

DKN = 1.0 - DK1

XA = (((X(1)*DI1+X(2)*DIN)*DJ1 + (X(3)*DI1+X(4)*DIN)*DJN)*DK1
    + ((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
    + ((Y(5)*DI1+Y(6)*DIN)*DJ1 + (Y(7)*DI1+Y(8)*DIN)*DJN)*DKN

RETURN

END
SUBROUTINE VOLUM7 (CLEVEL, F00, F10, F11, F01, NSEG, X, Y)

REAL X(4), Y(4)

AT LEAST ONE CROSSING WILL BE FOUND. TEST THE LOWER SEGMENT FIRST.

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

RECORD THE LOWER CROSSING.

ISEG = ISEG + 1

X(ISEG) = DX

Y(ISEG) = 0.0

TEST THE RIGHT SEGMENT.

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

RECORD THE RIGHT SIDE CROSSING.

ISEG = ISEG + 1

X(ISEG) = 1.0

Y(ISEG) = DY

TEST THE TOP SEGMENT.

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

RECORD THE TOP CROSSING.

ISEG = ISEG + 1

X(ISEG) = DX

Y(ISEG) = 1.0

TEST THE LEFT SIDE SEGMENT.

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

RECORD THE LEFT SIDE CROSSING.

ISEG = ISEG + 1

X(ISEG) = 0.0

Y(ISEG) = DY

SAVE ANY LINE SEGMENTS FOUND

3601
3602 C
3603 C
3604 C
3605 C
3606 C
3607 C
3608 C
3609 C
3610 C
3611 C
3612 C
3613 C
3614 C
3615 C
3616 C
3617 C
3618 C
3619 C
3620 C
3621 C
3622 C
3623 C
3624 C
3625 C
3626 C
3627 C
3628 C
3629 C
3630 C
3631 C
3632 C
3633 C
3634 C
3635 C
3636 C
3637 C
3638 C
3639 C
3640 C
3641 C
3642 C
3643 C
3644 C
3645 C
3646 C
3647 C
3648 C
3649 C
3650 C
3651 C
3652 C
3653 C
3654 C
3655 C
SUBROUTINE VOLUM7

5 NSEG = ISEG
RETURN
END
SUBROUTINE VOLUM8 (XA, YA, XB, YB, T, NT)

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

IF (SQRT((XB-XA)**2 + (YB-YA)**2) .LT. 1.9) RETURN

CHECK FOR HIDDEN PORTIONS OF THE LINE.

RSEG=FLOAT(NTT)/1024.
LAT = RSEG*XA
LBT = 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(IAT, JAT)) GO TO 1
CALL LINE (XA,YA, XB,YB)
RETURN

AT LEAST PART OF THE LINE IS HIDDEN. FIND REASONABLE LIMITS.

KMAX = (MAXO(XI,KJ)/XINC)*KINC + 1
IF (KMAX .LT. 2) RETURN

NOW CHECK EACH SEGMENT ALONG THE LINE.

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)
IF (SW1) GO TO 3

UPDATE THE FIRST POINT IF SW1 IS .FALSE.

IF (.NOT.T(IK, IY)) GO TO 3
XST = X
YST = Y
SW1 = .TRUE.

SET THE END POINT.

3 IF (T(IK, 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)
3714     SW1 = .FALSE.
3715     2 CONTINUE.
3716     RETURN
3717     END
3718     C
SUBROUTINE VOLUME (MODE, F, NX, NY, NZ, CLEVE, NCL, T, NT)

THIS SET OF ROUTINES IS DESIGNED TO DRAW ON A DEVICE, SUCH AS A PRINTER, A PLOTTER, THE SCA4020 OR THE TEKTRONIX GRAPHICS TERMINAL, A TWO-DIMENSION PROJECTION OF A THREE DIMENSIONAL ARRAY (FIGURE). THE HIDDEN LINES ARE ELIMINATED TO SIMULATE THE EFFECT OF "3D".

THE COLOR OF THE FIGURES (FOR THE SCA4020 ONLY) MUST BE SET BY THE USER IN THE CALLING PROGRAM USING THE APPROPRIATE CALLS TO THE GRAPHICS PACKAGE.

THE AXIS ARE LABELED AS FOLLOWS: X IS LEFT TO RIGHT
Y IS INTO THE PAGE
Z IS BOTTOM TO TOP

TO INITIALIZE THE PLOTTING PACKAGE, "VOLSET" MUST BE CALLED.
IF A FRAME SURROUNDING THE DRAWING IS WANTED, CALL "VOLFRM".
THE ROUTINE LOOPS OVER THE Y AXIS FROM FRONT TO BACK, PLOTTING EACH SUCCESSIVE (X,Z) PLANE. AT THE END OF THE (X,Z) CYCLE, THE ARRAY "T" IS SET TO INDICATE WHAT WILL BE HIDDEN BY THE CURRENT PLANE.

CALLING SEQUENCE OR OPERATIONAL PROCEDURE:
CALL VOLSET(XP, YP, T, NT)
CALL VOLUME (MODE, F, NX, NY, NZ, CLEVE, NCL)
CALL VOLFRM(MODE1)

ARGUMENTS (TYPE AND SIGNIFICANCE) AND/OR INITIAL CONDITIONS:
XP CONTAINS THE LOCATIONS OF THE X POSITION OF THE VERTICES (8)
YP CONTAINS THE CORRESPONDING Y VALUES (8)
T IS THE PLOTTING ARRAY (BOOLEAN) TO ELIMINATE THE HIDDEN LINES WHICH IS NT X NT IN SIZE (THIS ARRAY TOGETHER WITH NX,NY,NZ DETERMINE THE FINESSE OF THE PLOT).
MODE DETERMINES WHICH SURFACE IS PLOTTED (USED WITH CLEVE)
ABS(MODE)=1 WILL PLOT THE CONTOUR LEVEL "CLEVE" IN EACH PLANE AND THEN FIXES THE HIDDEN LINE MATRIX "T".
ABS(MODE)=2 SIMPLY FIXES THE HIDDEN LINE MATRIX "T" WITHOUT PLOTTING. THIS IS USEFUL IN AVOIDING MULTIPLE PLOTS OF THE SAME FIXED FIGURE.
MODE<0 IS USED WHEN THE VALUE OF THE FUNCTION ON THE "OUTSIDE" IS LESS THAN "CLEVE".
MODE>0 IS USED WHEN THE VALUE OF THE FUNCTION ON THE "OUTSIDE" IS GREATER THAN "CLEVE".
F IS THE THREE-DIMENSIONAL FIGURE TO BE PLOTTED.
NX, NY, NZ ARE THE DIMENSIONS OF THIS ARRAY <F(NX, NY, NZ)>
CLEVE IS THE CONTOUR SURFACE LEVEL FOR PLOTTING.
NCL IS THE NUMBER OF PAIRS OF (MODE,CLEVE) TO BE PLOTTED.
MODE1 IS THE MODE OF FRAMING 0=> "+" AT THE EIGHT VERTICES;
1=> A BOX FORMED BY LINES.
SUBROUTINE VOLUME

COMMON/VOLCOM/XR,YR,A,TP,AP,DX,DY,DZ,X,Y,Z,SCI,FSEG,NTT
LOGICAL T(NT,NT)
DIMENSION XR(8),YR(8),DX(4),DY(4),DZ(4),X(4),Y(4),Z(4),SCI(4)
INTEGER MODE(NCL)
REAL F(NX,NY,NZ),CLEVE(NCL)
REAL XP(8),YP(8)

CALL VOLUME (XR,YR,NX,NY,NZ)

THE OUTER LOOP IS OVER THE I,K SURFACES MOVING BACK.

DO 1 J = 1, NY
   IF (J.EQ.1) GO TO 2
THEN LOOP OVER THE VARIOUS CONTOUR LEVELS.

DO 20 LL = 1, NCL
   CLEVEL = CLEVE(LL)
   MODEL = MODE(LL)
   IF (IABS(MODE) .EQ. 2) GO TO 20
   FIRST COMPUTE THE FRONT-TO-BACK LINE SEGMENTS.

   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.LE.CLEVEL .OR. FMIN.GT.CLEVEL) GO TO 4
   CALL VOLUM7 (CLEVEL, F(I,J-1,K), F(I,J,K), F(I,J-1,K),
                 F(I,J,K), F(I,J-1,K), NSEG, DX, DY, DZ)
   IF (NSEG.LE.1) GO TO 4
   DO 5 L = 1, NSEG
      X(L) = FLOAT(I) + DX(L)
      Y(L) = FLOAT(J-1) + DY(L)
      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)
   CONTINUE
20 CONTINUE
   CONTINUE
3 CONTINUE
   DO 6 I = 1, NX
   DO 7 K = 2, NZ
   FMAX = AMAX1 (F(I-1,J-1,K), F(I-1,J,K), F(I,J,K), F(I-1,J,K))
   FMIN = AMIN1 (F(I-1,J-1,K), F(I-1,J,K), F(I,J,K), F(I-1,J,K))
   IF (FMAX.LE.CLEVEL .OR. FMIN.GT.CLEVEL) GO TO 7
   CALL VOLUM7 (CLEVEL, F(I-1,J-1,K), F(I,J,K), F(I-1,J,K),
                 F(I,J,K), NSEG, DX, DY)
   IF (NSEG.LE.1) GO TO 7
   DO 8 L = 1, NSEG
      X(L) = FLOAT(I-1) + DX(L)
SUBROUTINE VOLUME

3829     Y(L) = FLOAT(J-1) + DY(L)
3830     Z(L) = FLOAT(K)
3831     CALL VOLUM6 (XA, YA, X(1), Y(1), Z(1))
3832     CALL VOLUM6 (XB, YB, X(2), Y(2), Z(2))
3833     CALL VOLUM6 (XA, YA, XB, YB, T, NT)
3834     IF (NSEG .NE. 4) GO TO 7
3835     CALL VOLUM6 (XA, YA, X(3), Y(3), Z(3))
3836     CALL VOLUM6 (XB, YB, X(4), Y(4), Z(4))
3837     CALL VOLUM6 (XA, YA, XB, YB, T, NT)
3838     CONTINUE
3839     CONTINUE
3840     CONTINUE
3841     C
3842     C     NOW PLOT THE X-Z SURFACE CONTOURS BEFORE CORRECTING THE HIDDEN LINE ARRAY.
3843     C
3844     C
3845     DO 21 LL = 1, NCL
3846     CLEVEL = CLEVEL(LL)
3847     MODEL = MODE(LL)
3848     IF (ABS(MODELL) .EQ. 2) GO TO 21
3849     DO 9 K = 2, NZ
3850     DO 10 I = 2, NX
3851     FMAX = AMAX1 (F(I-1, J, K-1), F(I, J, K-1), F(I-1, J, K))
3852     FMIN = AMIN1 (F(I-1, J, K-1), F(I, J, K-1), F(I-1, J, K))
3853     IF (FMAX .LT. CLEVEL .OR. FMIN .GT. CLEVEL) GO TO 10
3854     CALL VOLUM7 (CLEVEL, F(I-1, J, K-1), (I, J, K-1),
3855           F(I, J, K), F(I-1, J, K), NSEG, DX, DZ)
3856     IF (NSEG .LE. 1) GO TO 10
3857     DO 11 L = 1, NSEG
3858     X(L) = FLOAT(I-1) + DX(L)
3859     Y(L) = FLOAT(J)
3860     Z(L) = FLOAT(K-1) + DZ(L)
3861     CALL VOLUM6 (XA, YA, X(I), Y(I), Z(I))
3862     CALL VOLUM6 (XB, YB, X(2), Y(2), Z(2))
3863     CALL VOLUM6 (XA, YA, XB, YB, T, NT)
3864     IF (NSEG .NE. 4) GO TO 10
3865     CALL VOLUM6 (XA, YA, X(3), Y(3), Z(3))
3866     CALL VOLUM6 (XB, YB, X(4), Y(4), Z(4))
3867     CALL VOLUM6 (XA, YA, XB, YB, T, NT)
3868     CONTINUE
3869     CONTINUE
3870     CONTINUE
3871     C
3872     C     FILL THE HIDDEN LINE ARRAY AFTER PLOTTING IN THE NEW PLANE.
3873     C
3874     DO 22 LL = 1, NCL
3875     CLEVEL = CLEVEL(LL)
3876     MODEL = MODE(LL)
3877     DO 12 K = 2, NZ
3878     DO 13 I = 2, NX
3879     IF (MODEL .GT. 0) .AND. AMIN1(F(I-1, J, K-1), F(I, J, K-1),
3880           F(I-1, J, K), F(I, J, K)).GT.CLEVEL) GO TO 13
3881     IF (MODEL .LT. 0) .AND. AMAX1(F(I-1, J, K-1), F(I, J, K-1),
3882           F(I-1, J, K), F(I, J, K)).LT.CLEVEL) GO TO 13
3883     C
SUBROUTINE VOLUME

THERE IS HIDDEN STUFF SOMEWHERE IN THE CELL. FIRST FIND THE LIMITS OF THE CELL.

CALL VOLUME6 (X00, Y00, FLOAT(I-1), FLOAT(J), FLOAT(K-1))
CALL VOLUME6 (X10, Y10, FLOAT(I), FLOAT(J), FLOAT(K-1))
CALL VOLUME6 (X11, Y11, FLOAT(I), FLOAT(J), FLOAT(K))
CALL VOLUME6 (X01, Y01, FLOAT(I-1), FLOAT(J), FLOAT(K))
DIX = AMAX1 (ABS(X10-X00), ABS(X11-X01))
DIX = AMAX1 (ABS(Y01-Y00), ABS(Y11-Y10))
NSEGX = (DIX + FSEG)/FSEG
NSEGZ = (DIX + FSEG)/FSEG

DI = 1.0/NSEGX
DK = 1.0/NSEGZ
DO 14 II = 1, NSEGX
XGRID = (FLOAT(I) - 0.5)*DI
DO 15 KK = 1, NSEGZ
ZGRID = (FLOAT(K) - 0.5)*DK
CALL VOLUME6 (XA, YA, XGRID+FLOAT(I-1), FLOAT(J), ZGRID+FLOAT(K-1))
IAX = XA/FSEG
IYA = YA/FSEG
IF (.NOT.T(IAX, IYA)) GO TO 15

FIND THE VALUE OF F AT THE POINT UNDER SCRUTINY.

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(IAX, IYA) = .FALSE.
IF (MODEL.LT.0. AND. FVAL.GT.CLEVEL) T(IAX, IYA) = .FALSE.

ENTRY VOLSET (XP, YP, T, NT)

THIS FIRST ENTRY INITIALIZES THE PLOT GEOMETRY AND FILLS THE HIDDEN LINE ARRAY T WITH THE INITIAL .TRUE. FOR TRANSPARENT. THE ARRAY T IS NT X NT (OR 16K WORDS MAX) AND IS PASSED IN FROM THE OUTSIDE. THIS CORRESPONDS TO 1024/NT RASTER UNITS ON THE SC4020.

SCI(1) = 1.0
SCI(2) = 0.0
SCI(3) = 1.0
SCI(4) = 0.0
DO 100 I = 1, NT
DO 100 J = 1, NT
T(I,J) = .TRUE.
FSEG = 1024.0/FLOAT(NT)
DO 103 I = 1, 8
XR(I) = XP(I)
3939   103   YR(I) = YP(I)
3940       NTT = NT
3941       RETURN
3942       END
SUBROUTINE WDCOUNT

COUNT THE NUMBER OF CHARACTERS (LEGITIMATE) IN A STRING


DATA E/'V'/, P/',', B/'B'/', H/'H'/, U/'U'/, D/'D'/, L/'L'/
DATA C/'C'/', R/'R'/, O/'O'/, M/'M'/

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
      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
101 I = I + 2
132 IF(I.GT.132) GO TO 400 ; QUIT ANYHOW
100 ENDIF
200 I = I + 1
300 IP = IP + 1
SUBROUTINE WDCOUNT

3998      GO TO 100
3999      C
4000      C    TERMINATE
4001      C
4002      300    NC = IP
4003      RETURN
4004      C
4005      C    TERMINATE
4006      C
4007      400    NC = 0
4008      RETURN
4009      END
SUBROUTINE WDDRAW(XSTART, YSTART, DX, DY, SX, SXY, SY, W)

XSTART REAL - X COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
CHARACTER TO BE DRAWN

YSTART REAL - Y COORDINATE (MATHEMATICAL SPACE) OF THE FIRST
CHARACTER TO BE DRAWN

DX REAL - INCREMENT TO BE ADDED TO THE X COORDINATE FOR
EACH CHARACTER DRAWN

DY REAL - INCREMENT TO BE ADDED TO THE Y COORDINATE FOR
EACH CHARACTER DRAWN

SX REAL - X MATH SPACE SIZE FOR CHARACTERS

SXY REAL - SLANT MODIFIER FOR CHARACTERS

SY REAL - Y MATH SPACE SIZE FOR CHARACTERS

W - CHARACTER STRING TO BE DRAWN.

CONTROL CHARACTERS

L - TEXT IS DRAWN TO THE LEFT OF XSTART,YSTART

M - TEXT IS CENTERED AT XSTART,YSTART

R - TEXT IS DRAWN TO THE RIGHT OF XSTART,YSTART

H - CHANGE CHARACTER SETS ... H01, H02, ...
    ... H20, H21 FOR CHARACTER SETS
    1, 2, ... 20, AND 21.

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.

C - CHANGE COLOR, C00, C01, ... TO CHANGE
    TO COLOR 0, 1, ...

. - END OF CHARACTER STRING.

**********************************************************************

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/'1'/, P/'0'/, 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, IRVERSE, CHSIZE(9), ITKWIT, LUDIAG, LUHSET

EQUIVALENCE (CC, CHNUMB)

INTEGER SELT(132), SE, CH(132), ICL(132)

C
SUBROUTINE WDDRAW

4065 C SET THE PLOTTING SPACE PARAMETERS
4066 C
4067 CHSIZE(2) = SY * XYCOORD(3) * CHSIZE(6)
4068 CHSIZE(7) = CHSIZE(2)*(CHSIZE(3)+.5*(1.8-CHSIZE(6)))/XYCOORD(1)
4069 CHSIZE(8) = CHSIZE(2)*(CHSIZE(4)+.5*(1.8-CHSIZE(6)))/XYCOORD(3)
4070 C
4071 XL = 0.0
4072 YL = 0.0
4073 IP = 0
4074 ICOL = 0
4075 XLAST = XL
4076 WIDTH = 0.0
4077 SC = 1.0
4078 IC = 1
4079 I = 1
4080 CENTER = 'R'
4081 C
4082 C SCAN THE INPUT STRING FOR JUSTIFICATION AND END OF TEXT
4083 C
4084 100 IF(W(I).EQ.E) THEN
4085 IF(W(I+1).EQ.H) THEN
4086 SE = CTOI(W(I+2),IC)*10 + CTOI(W(I+3),IC)
4087 IF (SE.LT.1.0R.SE.GT.24) SE = SDFLT
4088 I = I + 2
4089 ELSE
4090 IF(W(I+1).EQ.U) THEN
4091 XL = XL + SX * 0.20 * SC
4092 YL = YL + SY * 0.8 * SC
4093 SC = SC * 0.6
4094 ELSE
4095 IF(W(I+1).EQ.D) THEN
4096 XL = XL + SX * 0.06 * SC
4097 YL = YL - SY * 0.20 * SC
4098 SC = SC * 0.6
4099 ELSE
4100 IF(W(I+1).EQ.L.OR.W(I+1).EQ.R.OR.W(I+1).EQ.M) THEN
4101 CENTER = W(I+1)
4102 ELSE
4103 IF(W(I+1).EQ.B) THEN
4104 XL = XL - WLAST
4105 ELSE
4106 IF(W(I+1).EQ.O) THEN
4107 YL = 0.0
4108 SC = 1.0
4109 ELSE
4110 IF(W(I+1).EQ.C) THEN
4111 ICOL = CTOI(W(I+2),IC)*10 + CTOI(W(I+3),IC)
4112 I = I + 2
4113 ELSE
4114 IF(W(I+1).EQ.E) THEN
4115 GO TO 200
4116 ELSE
4117 IF(W(I+1).EQ.P) THEN
4118 GO TO 300
4119 ELSE
SUBROUTINE WDDRAW

I = I - 1
GO TO 101
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
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(LUDLAG.GT.0) WRITE(LUDLAG,201) CHNUMB,SE,W(I)
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))
CWMAX = MAX(CWMAX, XLL(J))
202
ENDIF
ENDIF
ENDIF
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
SUBROUTINE WDDRAW

4175 SETL(IP) = SE
4176 WLAST = (0.500*SC+0.5)*(CWMAX-CWMIN) + 0.20*SC*DX
4177 XL = XL + WLAST
4178 YL = YL + DY
4179 203 I = I + 1
4180 GO TO 100
4181 C
4182 C DRAW IT - FIRST SET THE CORRECT CENTERING
4183 C
4184 300 XOFF = X(1)
4185 WIDTH = XL - XOFF
4186 IF (CENTER.EQ.R) THEN
4187 DXX = 0.0
4188 ELSE IF (CENTER.EQ.M) THEN
4189 DXX = WIDTH / 2.
4190 ELSE IF (CENTER.EQ.L) THEN
4191 DXX = WIDTH
4192 END IF
4193 DO 301 I = 1, IP
4194 301 X(I) = X(I) - XOFF - DXX
4195 C
4196 C DO ANY SHIFTING AND NECESSARY ROTATIONS
4197 C
4198 COSX = COS(ANGLE)
4199 SINX = SIN(ANGLE)
4200 SX = ABS(XYCOORD(1))
4201 SY = ABS(XYCOORD(3))
4202 SX = 1. / SX
4203 SY = 1. / SY
4204 DO 304 I = 1, IP
4205 XI = X(I) * COSX * SX - Y(I) * SINX * SY
4206 YI = X(I) * SINX * SX + Y(I) * COSX * SY
4207 X(I) = XI * OSX + XSTART
4208 Y(I) = YI * OSY + YSTART
4209 C
4210 C NOW DO THE DRAW
4211 C
4212 IF(IP.LE.0) GO TO 303 , NOTHING TO DO
4213 ICOL = 0
4214 DO 305 J = 1, IP
4215 IF(ICL(J).NE.0.AND.ICL(J).NE.ICOL) THEN
4216 CALL COLOR(ICL(J))
4217 ICOL = ICL(J)
4218 ENDIF
4219 305 CALL HHDRAW (X(J), Y(J), SX*SCL(J), SY*SCL(J), CH(J), SETL(J), IER)
4220 303 CHSIZE(2) = CHSIZE(1)
4221 ANGLE = 0.0
4222 RETURN
4223 C
4224 C CHANGE THE DEFAULT CHARACTER SET - BYPASS THE ESCAPE SEQUENCE
4225 C
4226 C ENTRY CHRSET (ISET)
4227 IF(ISET.GT.0) SE = ISET
4228 IF(LUDIAG.GT.0) WRITE(LUDIAG,302) ISET
RETURN
302 FORMAT(' CHANGE THE DEFAULT CHARACTER SET TO #',I3)
RETURN
END
**Title and Subtitle**

A Device Independent Graphics Kernel

**Author(s)**

Walter W. Jones and Alicia B. Fadell

**Perfoming Organization**

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
Gaithersburg, Maryland 20899

**Sponsoring Organization**

NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

**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.

**Key Words**

device independence; display devices; graphics

**Availability**

- Unlimited
- For Official Distribution. Do Not Release to NTIS
- Order From National Technical Information Service (NTIS), Springfield, VA. 22161

**No. of Printed Pages**

241

**Price**

USCOMM-DC 6043-P80