MASTER COPY
DO NOT REMOVE

THE SLAC UNIFIED GRAPHICS SYSTEM
PL/1 VERSION

ROBERT C. BEACH
COMPUTATION GROUP
STANFORD LINEAR ACCELERATOR CENTER
STANFORD, CALIFORNIA 94305

CGTM No. 142
January 1973

Revised:
June 1973
November 1973
September 1974

CGTM 170
# Table of Contents

<table>
<thead>
<tr>
<th>SECTION</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION TO THE SLAC UNIFIED GRAPHICS SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>THE PURPOSE OF THE SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>A BRIEF DESCRIPTION OF THE PROCEDURES</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>A DETAILED DESCRIPTION OF THE PROCEDURES</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>GRAPHIC ELEMENT GENERATION</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1</td>
<td>PROCEDURE UGGEN</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2</td>
<td>PROCEDURE UGSET</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3</td>
<td>PROCEDURE UGCTRL</td>
<td>7</td>
</tr>
<tr>
<td>2.1.4</td>
<td>PROCEDURE UGSOUT</td>
<td>8</td>
</tr>
<tr>
<td>2.1.5</td>
<td>PROCEDURE UGSTART</td>
<td>9</td>
</tr>
<tr>
<td>2.1.6</td>
<td>PROCEDURE UGELS</td>
<td>10</td>
</tr>
<tr>
<td>2.1.7</td>
<td>PROCEDURE UGAIS</td>
<td>11</td>
</tr>
<tr>
<td>2.1.8</td>
<td>PROCEDURE UG3DHS</td>
<td>12</td>
</tr>
<tr>
<td>2.1.9</td>
<td>PROCEDURE UGMAX</td>
<td>15</td>
</tr>
<tr>
<td>2.1.10</td>
<td>PROCEDURE UGDATA</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>GRAPHIC DATA SET CONTROL</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1</td>
<td>PROCEDURE UGOPEN</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>PROCEDURE UGCLOSE</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3</td>
<td>PROCEDURE UGSLCT</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4</td>
<td>PROCEDURE UGSCALE</td>
<td>19</td>
</tr>
<tr>
<td>2.2.5</td>
<td>PROCEDURE UGINIT</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>DISPLAY DEVICE CONTROL</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1</td>
<td>PROCEDURE UGFUT</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2</td>
<td>PROCEDURE UGPICT</td>
<td>22</td>
</tr>
<tr>
<td>2.3.3</td>
<td>PROCEDURE UGCTRL</td>
<td>22</td>
</tr>
<tr>
<td>2.3.4</td>
<td>PROCEDURE UGGET</td>
<td>24</td>
</tr>
<tr>
<td>2.4</td>
<td>ATTENTION CONTROL</td>
<td>25</td>
</tr>
<tr>
<td>2.4.1</td>
<td>PROCEDURE UGREAD</td>
<td>27</td>
</tr>
<tr>
<td>2.4.2</td>
<td>PROCEDURE UGDATA</td>
<td>27</td>
</tr>
<tr>
<td>2.4.3</td>
<td>PROCEDURE UGREAD</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>KEYBOARD INPUT BUFFER CONTROL</td>
<td>28</td>
</tr>
<tr>
<td>2.5.1</td>
<td>PROCEDURE UGPUT</td>
<td>28</td>
</tr>
<tr>
<td>2.5.2</td>
<td>PROCEDURE UGGET</td>
<td>29</td>
</tr>
<tr>
<td>2.6</td>
<td>LIGHT PEN INPUT BUFFER CONTROL</td>
<td>30</td>
</tr>
<tr>
<td>2.6.1</td>
<td>PROCEDURE UGLPUT</td>
<td>30</td>
</tr>
<tr>
<td>2.6.2</td>
<td>PROCEDURE UGLGET</td>
<td>31</td>
</tr>
<tr>
<td>2.7</td>
<td>MISCELLANEOUS PROCEDURES</td>
<td>31</td>
</tr>
<tr>
<td>2.7.1</td>
<td>PROCEDURE UGDXY</td>
<td>32</td>
</tr>
<tr>
<td>2.7.2</td>
<td>PROCEDURE UGVIEWS</td>
<td>33</td>
</tr>
<tr>
<td>2.7.3</td>
<td>PROCEDURE UGORTH</td>
<td>34</td>
</tr>
<tr>
<td>2.7.4</td>
<td>PROCEDURE UGROI</td>
<td>35</td>
</tr>
<tr>
<td>2.7.5</td>
<td>PROCEDURE UGCIOL</td>
<td>36</td>
</tr>
<tr>
<td>2.7.6</td>
<td>PROCEDURE UGEHIV</td>
<td>37</td>
</tr>
</tbody>
</table>

3 USING THE SLAC UNIFIED GRAPHICS SYSTEM | 38 |

4 PROGRAMMING EXAMPLES | 39 |
| 4.1 PLOTTING A HIGHS SURFACE | 39 |
| 4.2 INTERACTIVE LISSAJOUS FIGURES | 42 |
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>NON-INTERACTIVE DISPLAY DEVICES</td>
<td>47</td>
</tr>
<tr>
<td>A.1</td>
<td>THE CALCOMP DRUM PLOTTERS (MODELS 564/565)</td>
<td>47</td>
</tr>
<tr>
<td>A.2</td>
<td>THE CALCOMP MICROFILE PLOTTER (MODEL 1675)</td>
<td>49</td>
</tr>
<tr>
<td>A.3</td>
<td>WILBUR DISPLAY FILES FOR THE Tektronix 4013</td>
<td>52</td>
</tr>
<tr>
<td>A.4</td>
<td>SAVING PICTURES IN A PARTITIONED DATA SET</td>
<td>55</td>
</tr>
<tr>
<td>B</td>
<td>INTERACTIVE DISPLAY DEVICES</td>
<td>58</td>
</tr>
<tr>
<td>B.1</td>
<td>THE IBM 2250 DISPLAY CONSOLE</td>
<td>58</td>
</tr>
<tr>
<td>B.2</td>
<td>THE GIF-IDIICH DISPLAY CONSOLE</td>
<td>61</td>
</tr>
<tr>
<td>B.3</td>
<td>THE Tektronix 4013 DISPLAY TERMINAL</td>
<td>67</td>
</tr>
<tr>
<td>C</td>
<td>THE CHARACTER STROKE GENERATING SUB-SYSTEM</td>
<td>71</td>
</tr>
<tr>
<td>C.1</td>
<td>MODULE UGX103A</td>
<td>71</td>
</tr>
<tr>
<td>C.2</td>
<td>PROCEDURES UGPOXY AND UGSTOP</td>
<td>75</td>
</tr>
<tr>
<td>C.3</td>
<td>MODULE UGX103Z</td>
<td>77</td>
</tr>
<tr>
<td>D</td>
<td>THE ERROR PROCESSING SUB-SYSTEM</td>
<td>78</td>
</tr>
<tr>
<td>D.1</td>
<td>PROCEDURE UGX1001</td>
<td>79</td>
</tr>
<tr>
<td>E</td>
<td>THE SCALING AND SCISSORING SUB-SYSTEM</td>
<td>80</td>
</tr>
<tr>
<td>E.1</td>
<td>MODULE UGX112X</td>
<td>81</td>
</tr>
<tr>
<td>F</td>
<td>THE OPTION SCANNING SUB-SYSTEM</td>
<td>83</td>
</tr>
<tr>
<td>F.1</td>
<td>PROCEDURE UGX1002</td>
<td>83</td>
</tr>
<tr>
<td>G</td>
<td>THE INTERNAL FORMAT OF A GRAPHIC ELEMENT</td>
<td>85</td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
<td>88</td>
</tr>
</tbody>
</table>

The textual part of this document was formatted using the FORMAT Text Processing Program. The figures were drawn on the 10 inch CALCOMP Drum Plotter using the SLAC Unified Graphics System and these drawings were then photographically reduced.
The SLAC Unified Graphics System is a collection of PL/1 procedures and FORTRAN subroutines which may be used to program for the graphic devices which can be driven from the central computing facility at SLAC. This document describes the PL/1 version of the system.

The graphic devices which are now supported by this system are:
1. The CALCOMP Drum Plotters (Models 564/565).
2. The CALCOMP Microfilm Plotter (Model 1675).
3. The Tektronix 4013 Display Terminal. This device may be used in an essentially non-interactive mode or in a fully interactive mode.
4. The IBM 2250 Display Console.
5. The GIF-IDIOM Display Console.

As new graphic devices become available at SLAC, it is intended that this system will be expanded to include them.

Since this document is rather long, the following suggestions are given for the person who is trying to read this document for the first time.

1. Read the remainder of Section 1. These sub-sections supply a basic introduction to this system.
2. In Section 2, read all of the introductory sections and the descriptions of the following procedures: UGINIT, UGEPNT, UGELIN, UGEXIT, UGOPEN, UGCLOS, UGEPUT, and UGPIC. If you are interested in interactive graphic devices, then you should also read about: UGCTRL, UGSTART, UGSTOP, UGWRITE, UGPUT, and UGGET.
3. Read the summary, in Sections A and B, about the specific graphic device in which you are interested.
4. Study the examples. To fully understand these examples you will have to look up some additional information.

SECTION 1.1: THE PURPOSE OF THE SYSTEM

SLAC has always had a number of graphic devices accessible from the central computing facility. In general, the PL/1 PROGRAMMER has had a set of procedures available for using each of these devices. One of the problems has been that the procedures available for one device have been totally different from those of another device. For instance, if a programmer had mastered the procedures for producing CALCOMP plots, it was necessary to learn an entirely different set of procedures to display the same picture on an IBM 2250 display console. Thus, changing from one graphic device to another required a programmer to learn a new set of procedures in addition to re-programming the problem.
Part of the problem has always been that two different graphic devices often have very little in common. For example, the CALCOMP drum plotters are non-interactive and a picture can contain an arbitrarily large amount of information while an IBM 2250 display console is interactive and has severe limitations on the complexity of the picture which can be displayed.

The SLAC Unified Graphics System is an attempt to eliminate many of these difficulties for a large class of graphic applications. The PL/1 programmer will use exactly the same set of procedures to create a picture for one graphic device as for another. Thus, if a programmer is using more than one graphic device, the number of different procedures that must be kept in mind is considerably reduced. In addition, the problem of changing a program from one graphic device to another is greatly simplified. For example, a program using the IBM 2250 can be converted to a program using the GIF-IDION with very little effort; in fact if the programmer is careful, no programming changes at all will be required, only the JCL need be changed. At the same time the PL/1 programmer will be able to take advantage of any of the special features of a specific display device. In taking advantage of these special features the programmer, of course, will have limited the possibilities of changing to another graphic device at some future time.

SECTION 1.2: A BRIEF DESCRIPTION OF THE PROCEDURES

The user of the SLAC Unified Graphics System describes a picture by packing information into an array of full word integers. These arrays, called "graphic elements", are usually constructed by calling procedures in this system to add point, line, and text data to the graphic element. A graphic element, constructed in this manner, is a device independent description of part of a picture. After a graphic element has been constructed, it may be transmitted to any of the display devices supported by this system. When a graphic element is transmitted to a display device, it is first transformed into orders for that device and then these orders are written to the device. After a graphic element has been transmitted to the device, the array containing the graphic element may be re-used; changing the array at that stage does not affect the picture. A complete picture may consist of many graphic elements.

The programmer specifies the positions of points, end points of lines, and centers of characters by giving floating point X and Y coordinates. The default drawing space (the "programmer coordinate system") for each graphic device is a square area with coordinates of (0.0,0.0) at its lower left hand corner and coordinates of (1.0,1.0) at its upper right hand corner. The picture is scissored at the outer boundaries of the display. A procedure is provided which allows the programmer to change these
scaling and scissoring limits.

The handling of point and line data in an efficient but device-independent manner is relatively straightforward, however the handling of text data poses some substantial problems. The reason is that some devices contain hardware character generators while others require that the programmer form the characters out of short line segments (called "strokes"). A programmed stroke generator can clearly make characters of any size and at any angle to the horizontal. Hardware character generators vary greatly from device to device and usually produce very limited character sizes or orientations. This system provides a stroke generator which may be used on any device. The programmer may specify one of the following possibilities when text data is added to a graphic element.

1. The text must be produced by the character generator. The programmer specifies character size and orientation. If a graphic element containing this type of text is transmitted to a device which does not have a character generator or has a character generator which cannot satisfy the given size or orientation requirements, then the text is ignored and will not appear in the picture.

2. A suggested character size and orientation is given. The size is specified in terms of the distance between the centers of adjacent characters in the programmer coordinate system. If the device character generator can come close to matching these requirements then it is used, otherwise the stroke generator is used.

3. The character size and orientation is given and the stroke generator is always used.

For non-interactive devices, there is little more that one can do except transmit pictures to them. However, for interactive devices, much more control is possible. Interactive devices usually have alphanumeric keyboards, and sometimes have light pens, function keyboards, and other control units. These control units are capable of generating attentions which may be signalled to the program from the device. Procedures are supplied in this system for enabling and disabling attentions. Another procedure allows the programmer to ask if an attention has occurred from a device. If a control unit is disabled, any attentions generated by it will not be reported to the program.

A program may use more than one display device. In this case only one display device is "active" at a time. Most of the procedures in this system communicate with the active device only. A procedure is available to make any display device the active one.
SECTION 2: A DETAILED DESCRIPTION OF THE PROCEDURES

This section gives a complete description of each of the procedures provided in the PL/1 version of the SLAC Unified Graphics System. The names of these procedures, and all other external names, start with the letters "UG" (standing for Unified Graphics). If the user of these procedures avoids external names beginning with "UG", there will be no naming conflicts between the user's names and names within this system.

The first argument in almost all procedures is a character string (named OPTIONS) which is used to specify information to the system which may be optional or device-dependent. The character string may contain any number of items separated by commas. Each item may be either (1) a simple string of characters or (2) a string of characters terminated by an equals sign, followed by a number, character string, or bit string. Thus the line:

'SHAL,ANGLE=5.3,DDNAME=FORS,LITES=01010'

is an example of the kinds of strings which are valid. If specific items of information are not supplied, default values will be assumed; if invalid information is supplied, it will be ignored. For example when an element is transmitted to a display device, the user may specify that the element may be detectable with a light pen. If this information is not specified, the element will not be detectable; if the display device is not an interactive device or does not have a light pen, then the item will be ignored and no error indication will be given. The descriptions of the procedures on the following pages include all of the items which can be utilized by any of the devices. Exactly which items in OPTIONS will be recognized by a graphic device is described in a later section devoted to that specific device.

This system contains a flexible scheme to report errors to the programmer. Any errors detected by these procedures are classified into one of four severity levels. The actions corresponding to these levels are:

1. Set error indicators.
2. Set error indicators and print a message.
3. Print a message and terminate the program without a core dump.
4. Print a message and terminate the program with a core dump.

The error indicators may be checked by the programmer at execution time to determine what the error was and possibly correct the problem. The error message contains the name of the procedure detecting the error and an index number. Additional information on this subject will be found in the section on the Error Processing Sub-System. The descriptions of each procedure include a list of all the error conditions, for all devices, which may be produced by the procedure. Both the index and level number (in parentheses) as well as a short description are given.
SECTION 2.1: GRAPHIC ELEMENT GENERATION

The procedures which are described in this section may be used to pack picture description data into a graphic element. A graphic element is nothing more than an array of full word integers. An example of a statement defining a graphic element is:

DECLARE ELEMENT(F06) FIXED BINARY(31,0);

As described below, the array ELEMENT may be initialized and have picture description data packed into it.

The first of the graphic element generation procedures is UGEINT. This procedure is used to initialize a graphic element before any picture description data is added to it. The next three procedures, UGEPFT, UGELIN, and UGETIT, perform only the most basic of functions. They add a single point, a single line segment, or a single line of text to a graphic element. The next two procedures, UGEPFT3 and UGELINS, are similar to UGEINT and UGELIN except that each call to these two procedures will add an array of points or line segments to the graphic element. Procedures UGAXIS and UGDSHS perform more complex operations; they add the description of a horizontal or vertical axis, or the description of a mesh surface with hidden lines eliminated to a graphic element. Finally procedures UGIEIN and UGIEITA add information to a graphic element which can only be processed by a limited set of display devices; most devices will ignore this special information.

The OPTIONS argument in these procedures usually specifies device dependent parameters about the graphic information. When the graphic element is finally transmitted to a display device as many of these parameters as possible will be utilized. The key words used for parameters are:

VDIN, DIMH, BRIT, VERT Intensity levels. Any of four different intensity levels may be given. The key words stand for "very dim", "dim", "bright", and "very bright".

HINE, STDY Hiss mode specification. Some interactive devices have a feature whereby part of the display may flash on and off. The key word STDY stands for "steady".

COL1, COL2, COL3, COL4 Color codes. Any of four colors may be specified.

SOLD, DASH, DDSH, DOTS Line structure specification. On some devices a line may be other than solid. The key words stand for "solid", "dashed", "dot-dashed", and "dots".

VSML, SMAL, Lay, VLRG Character size. These key words specify that the character generator on the device must be used. They stand for "very small", "small", "large", and "very large".

When default values must be supplied by the system, these default values will be: BRIT, STDY, COL1, SOLD, and SMAL.

In addition to using these procedures to construct graphic elements, the user may also put information directly into the graphic element. The correct format for this data is given in
the section on the Internal Format of a Graphic Element. The user should be warned, however, that incorrect data in a graphic element can cause unpredictable and disastrous results when it is transmitted to a display device.

The procedures described in this section are completely independent of the rest of the procedures in the system. Their only function is to aid in the generation of graphic elements. Since a graphic element is nothing more than an array filled with data, the user program can process it in many different ways. One obvious thing that the program can do is to transmit the element to a graphic device. Another possibility is that the user program could write the graphic element to a tape or disk data set. This capability provides a reasonably compact and device-independent means of saving pictorial data in a data set.

SECTION 2.1.1: PROCEDURE UGEINT

This procedure may be used to clear and initialize a graphic element. After this procedure has been called, other procedures may be called to add picture description data to the element. The first four words will be initialized so that they contain bookkeeping information.

The calling sequence is:
COLOR UGEINT(OPTIONS, ELEMENT);

The parameters in the calling sequence are:
COLOR OPTIONS A character string which may contain one of the following items:
COLOR CLEAR This is the normal use of this procedure. The element is cleared and made ready to accept more picture description data.
COLOR RESET The element is not cleared but is made ready to accept more data. The only time this operation is normally necessary is after a graphic element has been read from a data set and before any new data is added to it.
COLOR CONTINUE The element is cleared except that the last line data is retained. See the section on the Error Processing Sub-System for an example of a case when this operation is necessary.
COLOR ELEMENT The graphic element which is to be initialized.

The entry declaration for this procedure is:
COLOR DECLARE UGEINT EXTERNAL ENTRY(CHARACTER(*)
COLOR (*)) FIXED BINARY(31,0);

The errors detected by this procedure are:
COLOR 1(3): The length of the array ELEMENT is too small.
SECTION 2.1.2: PROCEDURE USEPRT

This procedure may be used to add a single point to a graphic element. If the previous data added to the element was points with the same options list, then two words of information are added to the graphic element; otherwise four words are added.

The calling sequence is:
CALL USEPRT (OPTIONS, X, Y, ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
VDIM, DMIN, BBIT, VERT Intensity levels.
WINK, STPY Wink mode specification.
COL1, COL2, COL3, COL4 Color codes.
X X coordinate of the point.
Y Y coordinate of the point.
ELEMENT The graphic element which will have the point added to it.

The entry declaration for this procedure is:
DECLARE USEPRT EXTERNAL ENTRY (CHARACTER (*),
FLOAT BINARY (21), FLOAT BINARY (21),
(* ) FIXED BINARY (31, 0));

The errors detected by this procedure are:
9(1): Not enough room was available in ELEMENT to contain the new point.

SECTION 2.1.3: PROCEDURE USELIN

This procedure may be used to add the end point of a single straight line segment to a graphic element. The vector may either be blanked or drawn. The user should blank to the first point and then either draw or blank to the following points to create the display. If the previous data added to the element was lines with the same options list, then two words of information are added to the graphic element; otherwise four words are added.

The calling sequence is:
CALL USELIN (OPTIONS, X, Y, BBIT, ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
SOLD, DASH, DBSH, DOTS Line structure specification.
VDIM, DMIN, BBIT, VERT Intensity levels.
WINK, STPY Wink mode specification.
COL1, COL2, COL3, COL4 Color codes.
X X coordinate of an end point of a line.
Y Y coordinate of an end point of a line.
BBIT  The blanking bit. The line segment formed in moving
to the given point is drawn if BBIT is '1'B, and
blanked if BBIT is '0'B.

ELEMENT  The graphic element which will have the end point of
the line added to it.

The entry declaration for this procedure is:
DECLARE UGELIN EXTERNAL ENTRY(CHARACTER(#) ,
FLOAT BINARY(21), FLOAT BINARY(21), BIT(1),
(#) FIXED BINARY(31,0));

The errors detected by this procedure are:
9(1): Not enough room was available in ELEMENT to contain the
new end point.

SECTION 2.1.4: PROCEDURE UGELIT

This procedure may be used to add text material to a graphic
element. The text will normally occupy a single line. The
actual characters will be produced either by the character
generator on the device, or by a programmed stroke generator. If
the previous data added to the element was text with the same
options list, then [N/4]+3 words of information are added to the
graphic element where N is the number of characters in the text
and [X] is the greatest integer in X; otherwise [N/4]+7 words are
added.

The calling sequence is:
CALL UGELIT(OPTIONS,Y,Y,BBIT,ELEMENT);

The parameters in the calling sequence are:

OPTIONS  A character string which may contain any of the
following items:
  VSHE,SHAH,LANG,VLNG  Character size. The use of
  one of these items indicates that the
text must be produced by the device
character generator. If the device does
not have a character generator, then this
text is ignored.
  XSPACING=(value)  Suggested character spacing.
  The device character generator will be
  used if the between character spacing
  will approximately match the given value.
  SPACING=(value)  Character spacing. The
  programmed stroke generator must be used.
  ANGLE=(value)  The angle that the characters
  make with the horizontal. The angle is
  measured in the counter-clockwise
direction.
  VDIM,DXHN,BBIT,VBRT  Intensity levels.
  WINK,STDF  Wink mode specification.
  COL1,COL2,COL3,COL4  Color codes.

If no character spacing of any kind is specified, then
a default of XSPACING=0.01367 is assumed. This

......
default will result in reasonably sized characters if
the default scaling parameters are in effect when the
element is transmitted to the display device.

X
Y
TEXT
ELEMENT

I coordinate of the center of the first character.
Y coordinate of the center of the first character.
The string containing the characters to be added to
the graphic element.
The graphic element which will have the text
information added to it.

The entry declaration for this procedure is:
DECLARE UGETIT EXTERNAL ENTRY (CHARACTER(9),
FLOAT BINARY(21), FLOAT BINARY(21), CHARACTER(9),
(*) FIXED BINARY(31, 0));

The errors detected by this procedure are:
1(2): The string TEXT contained more than 255 characters.
9(1): Not enough room was available in ELEMENT to contain the
new text information.

SECTION 2.1.5: PROCEDURE UGETPTS

This procedure may be used to add an array of points to a graphic
element. The same thing could be accomplished with repeated
calls to procedure UGENEXT, but one call to this procedure is more
efficient because the OPTIONS argument will be scanned only once.

The calling sequence is:
CALL UGETPTS (OPTIONS, IARRAY, YARRAY, NPTS, ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the
following items:
VDIM, DIME, BRT, VERT Intensity levels.
WINK, STDO Wink mode specification.
COLORS Color codes.

XARRAY An array containing the X coordinates of the points.
YARRAY An array containing the Y coordinates of the points.
NPTS The number of points in IARRAY and YARRAY.
ELEMENT The graphic element which will have the points added
to it.

The entry declaration for this procedure is:
DECLARE UGETPTS EXTERNAL ENTRY (CHARACTER(9),
(*) FLOAT BINARY(21), (*) FLOAT BINARY(21),
FIXED BINARY(31, 0), (*) FIXED BINARY(31, 0));

The errors detected by this procedure are:
9(1): Not enough room was available in ELEMENT to contain the
new points. The element is unchanged.
SECTION 2.1.6: PROCEDURE UGELNS

This procedure may be used to add an array of line segments to a graphic element. The same thing could be accomplished with repeated calls to procedure UGELIN, but one call to this procedure is more efficient because the OPTIONS argument will be scanned only once.

The calling sequence is:
CALL UGELNS (OPTIONS,YARRAY,YARRAY,NPTS,DBITS,ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items: SOLD,DASH,DOUBLE,DOTS Line structure specification.
VDIN,DIMH,BBIT,VBIT Intensity levels.
WINK,STDY Wink mode specification.
COL1,COL2,COl3,COl4 Color codes.
XARRAY An array containing the X coordinates of the end points of the line segments.
YARRAY An array containing the Y coordinates of the end points of the line segments.
NPTS The number of end points in XARRAY and YARRAY.
DBITS The blanking bits. The bits are used cyclically from DBITS. An argument of '1'B would result in a continuous curve; an argument of '10'B results in a line being drawn between the first and second points, the third and fourth, etc.
ELEMENT The graphic element which will have the line segments added to it.

The entry declaration for this procedure is:
DECLARE UGELNS EXTERNAL ENTRY (CHARACTER(*) ,
(9) FLOAT BINARY(21), (9) FLOAT BINARY(21),
FIXED BINARY(15,0), DBIT (9), (9) FIXED BINARY(31,0));

The errors detected by this procedure are:
9(1): Not enough room was available in ELEMENT to contain the new line segments. The element is unchanged.
SECTION 2.1.7: PROCEDURE UGAXIS

This procedure may be used to add the description of a horizontal or vertical axis with labels and tic marks to a graphic element. This procedure adds the actual data to the graphic element by calling procedures UGELIN and UGETIT. If either of these procedures signals an error, then the graphic element is restored to the state that it was in when UGAXIS was called and UGAXIS then signals an error.

The calling sequence is:
CALL UGAXIS (OPTIONS,LOPTIONS,COPTIONS,INDVAR,LOVAR,HIVAR,
LOLAB,HILAB,HLAB,ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
  VERT Indicates that a vertical axis is required, the default setting is a horizontal axis.
  HINHV The axis will be generated with an attempt to minimize the motion of the drawing stylus, the default is to minimize the amount of data put into the graphic element.
  HOLABL This item will suppress the labels on the axis; the axis itself and the tic marks will still be generated. The default is to have labels on the axis.
  LWID=<value> The field width of the labels, the default value is 8.
  LDEC=<value> The number of places to the right of the decimal point in the labels, the default value is 2.
  LXOFF=<value> The offset, in the X direction, of the first of the LWID characters in the label from the axis-tic mark intersection. The default value is -0.055 for a horizontal axis and -0.120 for a vertical axis.
  LYOFF=<value> The offset, in the Y direction, of the first of the LWID characters in the label from the axis-tic mark intersection. The default value is -0.020 for a horizontal axis and 0.000
for a vertical axis.

LOTH=<value> The length of the labeled tic marks on the negative side of the axis, the default value is 0.005.

HITH=<value> The length of the labeled tic marks on the positive side of the axis, the default value is 0.005.

NSTH=<value> The number of secondary tic marks between the labeled tic marks, the default value is 0. These tic marks are half the length of the labeled tic marks.

These default values are consistent with the default scaling and scissoring limits and the default character size.

LOPTIONS An options list that will be passed to UGETLIN.

COPTIONS An options list that will be passed to UGETTET.

INDVAR For a horizontal axis this is the Y coordinate of the axis in the programmer coordinate system, for a vertical axis this is the X coordinate.

LOVAR For a horizontal axis this is the X coordinate of the low end of the axis in the programmer coordinate system, for a vertical axis this is the Y coordinate.

HIVAR For a horizontal axis this is the X coordinate of the high end of the axis in the programmer coordinate system, for a vertical axis this is the Y coordinate.

LOLAB The label to be placed at the low end of the axis.

HILAB The label to be placed at the high end of the axis.

NLAB The number of labels and primary tic marks to be put on the axis.

ELEMENT The graphic element which will have the description of the axis added to it.

The reader should refer to procedure UGDIDY in the section on Miscellaneous Procedures for help in assigning values to LOLAB, HILAB, and NLAB which results in "round numbers" being used for labels.

The entry declaration for this procedure is:

DECLARE UGAXIS EXTERNAL ENTRY(CHARACTER(*),CHARACTER(*), Character(*),FLOAT BINARY(21),FLOAT BINARY(21),
FLOAT BINARY(21),FLOAT BINARY(21),FLOAT BINARY(21),
FIXED BINARY(15,0),(* FIXED BINARY(31,0));

The errors detected by this procedure are:

9(1): Not enough room was available in ELEMENT to contain the description of the axis. The element is unchanged.
SECTION 2.1.8: PROCEDURE UG3DHS

This procedure may be used to add the description of a perspective or orthogonal view of a three-dimensional mesh surface with hidden lines eliminated to a graphic element. A mesh surface is a surface which is defined by giving the x coordinates of points above a rectangular grid in the X-Y plane. The surface description is formed by joining adjacent points with straight lines. This procedure adds the actual data to the graphic element by calling procedure UGELIN. If UGELIN signals an error, then the graphic element is restored to the state that it was in when UG3DHS was called and UG3DHS then signals an error. The user should be warned that this procedure can generate a large amount of data which it will try to add to the graphic element.

This procedure does not solve the hidden line problem exactly. Instead, it produces an approximate solution. Under normal circumstances, it produces an acceptable picture with relatively fast execution speed. If the surface is very jagged, some line segments which should be eliminated may erroneously appear in the picture. This problem may usually be overcome by using a finer mesh. If the surface is very steep, some line segments which are visible may be erroneously eliminated. This second problem is accentuated by moving the eye position close to the surface in a perspective transformation. It can often be overcome by moving to a distant eye position or using an orthogonal transformation.

The calling sequence is:

CALL UG3DHS (OPTIONS,LOPTIONS,ARRAY,TRANS,UKAREA,ELEMENT);

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

LOWER Indicates that the under side of the surface is to be generated. Normally the upper side of the surface is generated. Two calls to this procedure are necessary to generate the full view of a surface.

NOCOMMON When both the upper and lower sides of the surface are produced, certain lines (for example, the front edge) will be duplicated. The programmer should use this option to suppress the common lines on either the upper or lower surface.

LOPTIONS An options list that will be passed to UGELIN.

ARRAY An array which contains the X, Y, and Z coordinates of the points on the surface. The format of ARRAY is:
The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YN) must be monotonically increasing. The X's and Y's do not have to be equally spaced, but better pictures usually result if this is the case.

TRANS
An array containing the transformation as produced by procedure UVIEW or UGORTH. There are a few restrictions on the views which may be used here. First, the eye point must not be directly above the surface but must be off to one side; and second, the transformation must be defined with a zero vector for HDIR.

UKARRA
An array which will be used as a work area. The amount of space that is needed in this array depends on many things including the view and the shape of the surface. A dimension of fifteen times the maximum dimension of ARRAY will be more than sufficient in most cases.

ELEMENT
The graphic element which will have the description of the mesh surface added to it.

The entry declaration for this procedure is:

DECLARE UG3DHS EXTERNAL ENTRY (CHARACTER(*), CHARACTER(*),
  (*,*) FLOAT BINARY(21), (31) FLOAT BINARY(21),
  (*) FLOAT BINARY(21), (**) FIXED BINARY(31,0));

The errors detected by this procedure are:

1(3): The array UKARRA is not large enough.
9(1): Not enough room was available in ELEMENT to contain the description of the mesh surface. The element is unchanged.

Algorithms of this nature have been described in the publications listed below. The algorithms used in UG3DHS is based on the information in [3,4].


SECTION 2.1.9: PROCEDURE UGENINV

On a very limited set of Display Devices, a graphic element may act as a subroutine. An element-subroutine will not be displayed except when it is invoked by another element. An element-subroutine may be invoked many times; in this way a sub-picture may appear on the screen many times while the orders to produce the sub-picture appear only once in the display file. Element-subroutines should be generated relative to an origin of (0.0,0.0). Element-subroutines may themselves invoke other element-subroutines and this nesting of invocations may be arbitrarily deep. This procedure may be used to invoke an element-subroutine from another element.

The calling sequence is:
CALL UGENINV(OPTIONS,X,Y,IDENT,ELEMENT);

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
  SOLD, DASH, DASH, DOTS Line structure specification.
  VDIN, DINT, BRIT, VBRT Intensity levels.
  HINK, STDYINK node specification.
  COL1, COL2, COL3, COL4 Color codes.

The above specifications apply to the invoked element-subroutine unless the element-subroutine overrides this information. In addition, some display devices can accept the following:

ANGLE=<value> The angle that the element-subroutine is to be rotated about when it is displayed. The default value is 0.0.

SCALE=<value> The scaling factor for the element-subroutine when it is displayed.
The default value is 1.0.

X X coordinate of the relative origin of the element-subroutine.

Y Y coordinate of the relative origin of the element-subroutine.

IDENT The identification of the element-subroutine to be invoked.

ELEMENT The graphic element which will have the invocation data added to it.

The entry declaration for this procedure is:
DECLARE UGENINV EXTERNAL ENTRY (CHARACTER(4), FLOAT BINARY(21),
  FLOAT BINARY(21), FIXED BINARY(15,0), (*) FIXED BINARY(31,0));

The errors detected by this procedure are:

9(1): Not enough room was available in ELEMENT to contain the invocation data.
SECTION 2.1.10: PROCEDURE UGEDTA

This procedure may be used to add data to an element which is transmitted to the display device in unmodified form. At present, this data may be sent only to a limited set of display devices. This procedure has a very specialized purpose and its successful utilisation cannot be accomplished with the information in this document.

The calling sequence is:

CALL UGEDTA (OPTIONS, STRING, ELEMENT);

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- SOLD, DASH, DBSH, DOTS Line structure specification.
- UDBN, UDBN, UDBT, UDBT Intensity levels.
- UINK, STDY Ink mode specification.
- COL1, COL2, COL3, COL4 Color codes.

STRING A character string containing the information to be added to the display element.

ELEMENT The graphic element which will have the given information added to it.

The entry declaration for this procedure is:

DECLARE UGEDTA EXTERNAL ENTRY (CHARACTER(*) , CHARACTER(*) ,
(*) FIXED BINARY (31,0));

The errors detected by this procedure are:
1(2): The string STRING contained more than 255 characters.
9(1): Not enough room was available in ELEMENT to contain the given information.

SECTION 2.2: GRAPHIC DATA SET CONTROL

The procedures in this section perform basic operations on a graphic device. Procedure UGOPEN is used to open a graphic device, while UGCLOSE is used to close it. UGSLCT can be used to select a device and make it the active graphic device. UGSCAL can be used to make changes to the default scaling and scissoring values. Finally UGINFO can be used to retrieve information about a graphic device.

SECTION 2.2.1: PROCEDURE UGOPEN

This procedure must be used to open a graphic device and make it ready for use. A graphic device must be opened before any use can be made of it. Opening a graphic device makes it the active
The calling sequence is:

```
CALL UGOPEN (OPTIONS, IDENT);
```

The parameters in the calling sequence are:

**OPTIONS** A character string containing the graphic device type and any device dependent information which may be required. The device type may be given by one of the following specific items:
- **CAL/10D** The 10 inch CALCOMP Drum Plotter.
- **CAL/29D** The 29 inch CALCOMP Drum Plotter.
- **CAL/6UR** The CALCOMP Microfile Recorder with 16mm unsprocketed film.
- **CAL/FCH** The CALCOMP Microfile Recorder with Microfiche film.
- **NYL4013** NYLBUR Display Files for the TETRONIX 4013 are saved in a partitioned data set.
- **DISKPD** Graphic elements, in their device independent form, may be written to, or read from, a partitioned data set.
- **IB2250** The IBM 2250 Display Console.
- **GIF/FCH** The GIF-IDIXON Display Console.
- **TEK4013** The TETRONIX 4013 Display Terminal. This item refers to the terminals which are under the control of the HILLYN Terminal Manager.

In addition to the specific selection of an output device, there are some generic names available. By using these names, the actual device is selected by a JCL statement instead of by the program. If one of the following items is used, then the options list must also contain the item **DDNAME=value**. Procedure UGOPEN will find the internal record of the DD card whose name is given and determine the type of output device to be used.

**BDSCOPE** This item selects a refresh display scope: either an IBM 2250 or the GIF-IDIXON.

**IDSCOPE** This item selects an interactive display scope: either an IBM 2250, the GIF-IDIXON, or a TETRONIX 4013 running under HILLYN.

In addition to the items described above, there is another item whose purpose is to enable experimental systems for new devices to be checked-out without making any changes to the system.

**SYSTEM=value** Where the value of this item is the name of the second level open module for the device.

The device dependent items are described in the sections on the specific graphic devices.

**IDENT** A numeric value which is used to identify the graphic device being opened.
The entry declaration for this procedure is:
DECLARE UGOFHE EXTERNAL ENTRY (CHARACTER (6),
FIXED BINARY (15, 0));

The errors detected by this procedure are:
1(3): Duplicate or invalid identification.
2(3): Too many devices. Only four can be opened at once.
3(3): A valid device was not specified.
4(3): The DD card corresponding to the generic name cannot be
       found.
5(3): The DD card corresponding to the generic name does not
give a valid device.
11(3): The data set cannot be opened. Check your DD cards.
12(3): Not enough display buffer space is available.

SECTION 2.2.2: PROCEDURE UGCLOS

This procedure must be used to terminate the use of a graphic
device. No more use can be made of the graphic device until it
is re-opened. If the device being closed is the active device
and other devices remain open, then one of these open devices
will become the active device.

The calling sequence is:
CALL UGCLOS (IDENT);

The parameter in the calling sequence is:
IDENT The identification of the device to be closed.

The entry declaration for this procedure is:
DECLARE UGCLOS EXTERNAL ENTRY (FIXED BINARY (15, 0));

The errors detected by this procedure are:
1(3): Identification is invalid, no such device is open.

SECTION 2.2.3: PROCEDURE UGSLECT

This procedure may be used to select any open graphic device and
make it the active device. The programmer will not have to use
this procedure unless more than one graphic device is open
concurrently.

The calling sequence is:
CALL UGSLECT (IDENT);

The parameter in the calling sequence is:
IDENT The identification of the device which is to become
       the active device.

The entry declaration for this procedure is:
DECLARE UGSLECT EXTERNAL ENTRY (FIXED BINARY (15, 0));
The errors detected by this procedure are:
1(3): The identification is invalid; no such device is open.

SECTION 2.2.4: PROCEDURE UGSCAL

This procedure may be used to retrieve or change the scaling and/or scissoring parameters of the active graphic device. The default values for a device are given in the sections devoted to specific devices. More information and some examples on how these values may be used is found in the section on the Scaling and Scissoring Sub-System.

The calling sequence is:
CALL UGSCAL(OPTIONS,PROGLN,DEVCLN);

The parameters in the calling sequence are:

- OPTIONS A character string which may contain any of the following items:
  GBT, PUT Flags to indicate if the limits are to be retrieved or changed.
  PROGL A flag to indicate that the programmer limits are to be retrieved or changed.
  DEVCL A flag to indicate that the device limits are to be retrieved or changed.
- PROGLN An array for the programmer scaling limits.
- DEVCLN An array for the device scaling limits.

The entry declaration for this procedure is:
DECLARE UGSCAL EXTERNAL ENTRY(CHARACTER(*),
(2,2) FLOAT BINARY(21),(2,2) FIXED BINARY(31,0));

The errors detected by this procedure are:
1(3): No device is active.
2(3): The new values are not valid.

SECTION 2.2.5: PROCEDURE UGINFO

This procedure may be used to determine the active device type, its identification, and certain device dependent items of information about the use of the device. This latter information is described in the sections covering the specific devices and can be useful in optimizing the use of a device.

The calling sequence is:
CALL UGINFO(STRING, IDENT, ARRAY);

The parameters in the calling sequence are:

- STRING A character string which will be set to contain the specific device identifications described in procedure UGOPEN.
- IDENT This will be set to the identification of the device.
ARRAY This array will be set to the device dependent information.

The entry declaration for this procedure is:

```plaintext
DECLARE UCINFO EXTERNAL ENTRY(CHARACTER(8),
     FIXED BINARY(15,0),(5) FIXED BINARY(15,0));
```

The errors detected by this procedure are:

1(3): No device is active.

SECTION 2.3: DISPLAY DEVICE CONTROL

This section describes those procedures that are directly related to controlling the picture and the graphic device itself. Procedure UGEPUT will transmit a graphic element to a display device. Procedure UGPICT gives the programmer control over the picture. Procedure UCCTRL allows the programmer to control the display device itself. Finally, procedure UGEGUT may be used to retrieve pictures from a partitioned data set.

For some interactive devices, the programmer may give a numeric identification to an element when it is transmitted. This identification may then be used to manipulate the element at a later time. For instance an element may have its light pen detectability status changed, or the element may be deleted from a picture.

For most display devices it is more efficient to create a picture from a small number of large graphic elements than from a large number of small elements. This is especially true on interactive devices which allow element manipulation. When individual elements are deleted from a picture, it is more efficient to delete them in reverse order from the way they were added to the picture.

Some interactive displays can be turned on and off. This can be useful to prevent a partial picture from flashing on the screen while a display is being changed. Some displays also allow an individual element to be temporarily turned off. Such an element is said to be in the "cntr" state (as opposed to the "include" state).

SECTION 2.3.1: PROCEDURE UGEPUT

This procedure may be used to transmit a previously constructed graphic element to a display device. When an element is transmitted to a display device, it is first translated to device-dependent orders, and then added to the display file of the device. After a graphic element has been transmitted to a
display device, the programmer may re-use the array containing the graphic element.

The calling sequence is:
CALL USEPUT (OPTIONS, IDENT, ELEMENT);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
  - DBIC Make the element light pen detectable.
  - OUT Leave the element in the out state.
  - ECNP Send non-compact orders to the device.
  - ON, OFF Leave the display in the "on" or "off" state.
  - FTC Position the graphic element relative to the tracking pattern. Elements positioned relative to the tracking pattern are not scissored.
  - ESUB The element is to become an element-subroutine and will not be displayed except when it is invoked by another element. Default values of intensity levels, vink mode, color code, and line structure are not provided for the data within the element-subroutines. These values are set by the invoking element. Element-subroutines are not scissored.
  - HINESIX=<value> The device-dependent form of the graphic element is expanded until its length is at least as great as the given value.
IDENT A numeric value which may be used to identify the element. A zero value means that the element is unidentified. If a display allows elements to be identified, then transmitting two elements with the same identification will cause the second to replace the first.
ELEMENT The graphic element which is to be transmitted.

The entry declaration for this procedure is:
DECLARE USEPUT EXTERNAL ENTRY (CHARACTER(*) ,
  FIXED BINNARY(15,0),(*) FIXED BINNARY(31,0));

The errors detected by this procedure are:
  1(3): No device is active.
  11(2): The element is too large for the device dependent element buffer.
  12(2): There is not enough room in the display file for this element.
  13(2): The maximum element count has been exceeded.
  99(2): This procedure is not supported for the active device.
SECTION 2.3.2: PROCEDURE UGPICT

This procedure may be used to control the picture on a display device. On some display devices, individual elements may be manipulated or deleted. Multiple operations may be performed in one call to this procedure as long as the operations are not contradictory.

The calling sequence is:

CALL UGPICT (OPTIONS, IDENT);

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- CLEAR If IDENT is zero, the entire display is cleared, otherwise the specific element is deleted.
- DETC, NDET Change light pen detectable status of an element.
- ONIT, INCL Change omit–include status of an element.
- ON, OFF Turn the display on or off.
- ALIAS=<value> If the display is being cleared to start a new picture and the output device allows pictures to be named, then this item may supply an alias for the name.

IDENT If a specific element is being manipulated, then this should give its identification.

The entry declaration for this procedure is:

DECLARE UGPICT EXTERNAL ENTRY (CHARACTER(9),
FIXED BINARY(15,0));

The errors detected by this procedure are:

13: No device is active.
99(2): This procedure is not supported for the active device.

SECTION 2.3.3: PROCEDURE UGCTRL

This procedure may be used to perform a number of special operations on a display device. Multiple operations may be performed in one call to this procedure as long as the operations are not contradictory.

The calling sequence is:

CALL UGCTRL (OPTIONS, IODATA);

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- BEEP Sound the audible alarm on the display device.
- FRAN The value in IODATA.IARRAY(1) gives the
number of regeneration cycles that should be performed before issuing a FMAH attention (if the attention is enabled).

LITES=<value> The value should be a bit string which specifies which of the lights on the function keyboard are to be turned on. A one bit causes the corresponding light to be turned on and a zero bit causes it to be turned off.

TPPUT Put the tracking pattern on the display device. The X and Y coordinates of the initial position of the tracking pattern in the programmer coordinate system must be given in IODATA.XARRAY(1) and IODATA.XARRAY(2).

TPGET Get the current position of the tracking pattern. The X and Y coordinates of the current position of the tracking pattern in the programmer coordinate system will be put into IODATA.XARRAY(1) and IODATA.XARRAY(2); the coordinates in the device coordinate system are put into IODATA.YARRAY(1) and IODATA.YARRAY(2). If the tracking pattern is not on the screen, then IODATA.XARRAY(1) and IODATA.XARRAY(2) are set to minus one.

TPREM Remove the tracking pattern from the display device.

TPIAC Put the display device into the inactive mode with respect to light pen input buffers. This mode is the default status for light pen input buffers.

TPDRE Put the display device into the draw mode with respect to light pen input buffers and set the segment length from IODATA.XARRAY(1). In this mode the display console operator can draw freehand curves on the screen with the light pen. The segment length value is given in programmer units and specifies the approximate length of the line segments which form the curve.

TPERS Put the display device into the erase mode with respect to light pen input buffers. In this mode, the display console operator can use the light pen to erase parts of the curves which were created in the draw mode.

TPCTL This is a special item which allows a tracking pattern to be put on the screen, positioned by the console operator, and its coordinates returned to the program. The coordinates are returned in the manner that they are for the TPGET item.

PSON Put the display device into a mode where it will check for element-subroutines.

PSON Put the display device into a mode where
it will ignore element-subroutines. This mode is the default status for element-
subroutines.

**ESTEC**

If the LPEH attention has been enabled with the OFF option, then an element-
subroutine trace back of the light penned item may be obtained if it is requested 
before the display is restarted. The element-subroutine trace back will appear 
in IODATA.IARRAY and will have the following form: The first word is the 
number of levels in the invocation trace back (if the light penned item is not 
within an element-subroutine, then this value is zero). The next two words 
contain the identification of the highest level element during the invoking and the 
byte offset of the invocation within the element. The next two words are for the 
next level of invocation, etc. Up to four levels of invocation may be 
reported; additional levels are lost.

**SHOPEN**

Open the shutter on the synchronized movie camera.

**SHCLOS**

Close the shutter on the synchronized movie camera.

**IODATA**

An input and output structure that is used when certain items are given in OPTIONS. The declaration 
for IODATA should be:

```
DECLARE 1 IODATA,
    2 IARRAY(10) FIXED BINARY(15,0),
    2 IARRAY(2) FLOAT BINARY(27);
```

The entry declaration for this procedure is:

```
DECLARE UGCTRL EXTERNAL ENTRY(CHARACTER(*))
    1,2 (10) FIXED BINARY(15,0),2 (2) FLOAT BINARY(21);
```

The errors detected by this procedure are:

1(3): No device is active.
99(2): This procedure is not supported for the active device.

**SECTION 2.3.4: PROCEDURE UGEGET**

This procedure may be used to retrieve an element from a 
partitioned data set. The element should have been written into 
the partitioned data set by using UCPUPUT with a device type of 
DISKFDS. This procedure is usable only when the active device 
type is DISKFDS with the INPUT option.

The calling sequence is:

```
CALL UGEGET (OPTIONS, ELEMENT);
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the
following items:
NAME=<value> When this item is given, the element that will be read is the first element in the partitioned data set member with the given name. If this item is not given, the element that will be read is the next element in the partitioned data set member that was previously accessed.

ELEMENT The array which will have the graphic element stored into it. The graphic element is ready to be transmitted to any graphic device; however if additional display information is to be added to the graphic element, then procedure UGKINT should be called with the RESET option before the new information is added to the element.

The entry declaration for this procedure is:
DECLARE UGKINT EXTERNAL ENTRY (CHARACTER(*) ,
(*) FIXED BINAY(31,0));

The errors detected by this procedure are:
1(3): No device is active.
11(1): End of Picture....No more elements are available in the partitioned data set member.
12(3): No partitioned data set member with the given name is available.
13(3): The graphic element that you are trying to read is larger than the array ELEMENT.
99(2): This procedure is not supported for the active device.

SECTION 2.4: ATTENTION CONTROL

This section describes procedures which allow the programmer to control attentions from an interactive display device. It is by means of these procedures that the programmer synchronizes the execution of the program with the actions of the console operator. Procedure UGHATN may be used to enable attentions from a control unit (such as the light pen) on a display device. Procedure UGDATN may be used to disable attentions from a control unit. When an attention is generated by an enabled control unit it is saved on an attention queue; an attention from a disabled unit is not saved. Finally, procedure UGHATN may be used to check to see if an attention has occurred and optionally wait until one has occurred.

Attentions are enabled or disabled by specifying an attention code to the appropriate procedure. When an attention is reported, this code as well as other useful information is made available to the program. This information is inserted into an attention data structure which must have the following declaration:
DECLARE 1 ATTEND,
   2 ATCODE CHARACTER(4),
   2 IARRAY (5) FIXED BINARY(15,0),
   2 IARRAY (2) FLOAT BINARY(21);

The following list gives (1) the valid attention codes (returned in ATTEND.ATCODE), (2) the contents of ATTEND.IARRAY(1), and (3) a description of the attention and the contents (if any) of the rest of ATTEND.IARRAY and ATTEND.IARRAY.

NONE 0 This is not an attention code but is a possible return value from UCTRATH when no attention is available to report.

KBBD 1 An attention generated by a special key on the keyboard which is reserved for this use.

LPEN 2 This occurs when the light pen is pointed at a detectable graphic element and the light pen switch is closed. ATTEND.IARRAY(2) gives the identification of the element, ATTEND.IARRAY(3) gives the byte offset of the item in the element that was pointed to, ATTEND.IARRAY(4) and ATTEND.IARRAY(5) give the X and Y coordinates of the item pointed to in the device coordinate system while ATTEND.IARRAY(1) and ATTEND.IARRAY(2) give the coordinates in the programmer coordinate system in effect when the attention is reported.

FRAM 3 This attention is generated after a specified number of display regeneration cycles.

PFEM 4 Program Function Keyboard Make. ATTEND.IARRAY(2) gives the button number which caused the attention and ATTEND.IARRAY(3) will contain a one if the button had its light on or a zero if its light was off.

PFEB 5 Program Function Keyboard Break. ATTEND.IARRAY(2) and ATTEND.IARRAY(3) are the same as for PFEM.

PFEM 6 The same as PFEM except that the attention is reported only if the button's light is on.

PFEB 7 The same as PFEB except that the attention is reported only if the button's light is on.

LPSH 8 Light Pen Switch Make.

LPSB 9 Light Pen Switch Break.

When an attention is enabled on some devices, the programmer may specify the disposition of the display after the attention has occurred. Normally the display will keep on running, however procedure UCTRATH will accept OFF in its option list to indicate that the display device is to be turned off when the attention occurs.
SECTION 2.4.1: PROCEDURE UGRATH

This procedure may be used to enable attentions from an interactive display device. When an enabled attention is generated by a device, a record of it is put on the attention queue for that device.

The calling sequence is:
CALL UGRATH (OPTIONS);

The parameter in the calling sequence is:
OPTIONS A character string which may contain OFF and any of the attention codes.

The entry declaration for this procedure is:
DECLARE UGRATH EXTERNAL ENTRY (CHARACTER (9)));

The errors detected by this procedure are:
1(3): No device is active.
99(2): This procedure is not supported for the active device.

SECTION 2.4.2: PROCEDURE UGDATH

This procedure may be used to disable attentions from an interactive display device.

The calling sequence is:
CALL UGDATH (OPTIONS);

The parameter in the calling sequence is:
OPTIONS A character string which may contain any of the attention codes.

The entry declaration for this procedure is:
DECLARE UGDATH EXTERNAL ENTRY (CHARACTER (9));

The errors detected by this procedure are:
1(3): No device is active.
99(2): This procedure is not supported for the active device.

SECTION 2.4.3: PROCEDURE UGRATH

This procedure may be used to retrieve an attention record from the attention queue of an interactive display device. When this procedure is called and the queue is empty, there are a number of possible options for the programmer to specify. These options are (1) UGRATH may return immediately, (2) it may wait for an indefinite length of time for an attention, or (3) it may wait for a specified length of time and return when the time interval runs out or when an attention occurs.
The calling sequence is:
CALL UGRATE (TIME, ATTN);

The parameters in the calling sequence are:
TIME A zero or positive value indicates the wait time (in
seconds) for this procedure. On return, TIME contains
the unexpired time. If TIME has a negative value the
procedure will wait indefinitely.
ATTN An attention data structure.

The entry declaration for this procedure is:
DECLARE UGRATE EXTERNAL ENTRY (FLOAT BINARY (21)),
1,2 CHARACTEN (4), 2 (5) FIXED BINARY (15, 0),
2 (2) FLOAT BINARY (21));

The errors detected by this procedure are:
1 (3): No device is active.
99 (3): This procedure is not supported for the active device.

SECTION 2.5: KEYBOARD INPUT BUFFER CONTROL

A "Keyboard Input Buffer" is a special kind of text element which
may be displayed on an interactive display and modified by the
console operator. When a keyboard input buffer is displayed, a
cursor will appear in it. The console operator may type on the
keyboard to insert characters into the buffer at the position
marked by the cursor. Procedure UCKPUT will put a keyboard input
buffer on an interactive device and procedure UCGET will read
the input buffer from the device. Keyboard input buffers are not
scissored; it is the programmer's responsibility to see that they
fit on the screen.

If the interactive device allows an element to be manipulated,
then a keyboard input buffer may also be identified and
manipulated. Its identification should be different from that of
any other keyboard input buffer or graphic element.

On some devices, multiple keyboard input buffers are permitted.
In this case there will be a key on the keyboard to move the
cursor from one keyboard input buffer to another so that the
console operator can enter or change information in any keyboard
input buffer.

SECTION 2.5.1: PROCEDURE UCKPUT

This procedure may be used to display a keyboard input buffer.

The calling sequence is:
CALL UCKPUT (OPTIONS, X, Y, IDENT, STRING);
The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- VSNL,VSAL,LANG,VLEI Character size.
- VDIN,DIMN,DLIT,VDIT Intensity levels.
- COL1,COL2,COL3,COL4 Color codes.
- LCASE On some devices, the console operator may type either upper or lower case characters. By default, all lower case characters are translated to upper case when the keyboard input buffer is read. However the programmer may specify LCASE if this translation is not wanted.

I X coordinate of the center of the first character.
Y Y coordinate of the center of the first character.
IDENT The identification of the keyboard input buffer.
STRING A character string which contains the initial values for the keyboard input buffer. This may be a string of blanks.

The entry declaration for this procedure is:

```plaintext
DECLARE UGNGET EXTERNAL ENTRY (CHARACTER(*),
FLOAT BINARY(21),FLOAT BINARY(21),
FIXED BINARY(15,0),CHARACTER(*)) ;
```

The errors detected by this procedure are:

1(3) : No device is active.
11(2) : STRING is longer that 256 characters.
12(2) : There is not enough room in the display file for the buffer.
13(2) : The maximum element count has been exceeded.
99(2) : This procedure is not supported for the active device.

SECTION 2.5.2: PROCEDURE UGNGET

This procedure may be used to read a keyboard input buffer from a display device.

The calling sequence is:

```plaintext
CALL UGNGET (IDENT,STRING,ICUR) ;
```

The parameters in the calling sequence are:

IDENT The identification of the keyboard input buffer to be read.
STRING A varying length character string which will have the keyboard input buffer placed in it.
ICUR If the keyboard input buffer being read contains the cursor, then this is the index of the character containing the cursor; otherwise it is zero.

The entry declaration for this procedure is:

```plaintext
DECLARE UGNGET EXTERNAL ENTRY (FIXED BINARY (15,0),
CHARACTER(*) VARYING,FIXED BINARY (15,0)) ;
```
The errors detected by this procedure are:
1(3): No device is active.
11(2): There is no keyboard input buffer with the given identification.
12(2): There is not enough room in STRING for the keyboard input buffer.
99(2): This procedure is not supported for the active device.

SECTION 2.6: LIGHT PEN INPUT BUFFER CONTROL

A "Light Pen Input Buffer" is a special kind of element which may be displayed on some interactive devices and modified by the console operator. When a light pen input buffer is in the display file, the console operator may use the light pen to draw free-hand curves on the screen and erase parts of the curves that have been drawn. Procedure UGLPUT will add a light pen input buffer to the display file and procedure UGLGET will read the input buffer from the device. The identification of a light pen input buffer must be different from that of any other graphic element.

The only graphic device at SLAC which can accept light pen input buffers is the GIP-IDITION Display Console.

SECTION 2.6.1: PROCEDURE UGLPUT

This procedure may be used to add a light pen drawing buffer to a display file.

The calling sequence is:
CALL UGLPUT(OPTIONS,IDENT,LENGTH);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
SOL,DASH,DASH,DOTS Line structure specification.
VDIN,DIMN,BINT,VERT Intensity levels.
COL1,COL2,COL3,COL4 Color codes.
IDENT The identification of the light pen input buffer.
LENGTH The length (in bytes) of the light pen input buffer.

The entry declaration for this procedure is:
DECLARE UGLPUT EXTERNAL ENTRY CHARACTER(0),
FIXED BINARY(15,0),FIXED BINARY(15,0));

The errors detected by this procedure are:
1(3): No device is active.
11(2): The element is too large for the device dependent
element buffer.

12(2): There is not enough room in the display file for this element.
13(2): The maximum element count has been exceeded.
99(2): This procedure is not supported for the active device.

SECTION 2.6.1: PROCEDURE UGLGET

This procedure may be used to read a light pen input buffer from a display device. The data from the screen is made available in a form that is the same as that required by procedure UGELMS.

The calling sequence is:
CALL UGLGET (IDENT, XARRAY, YARRAY, NPTS, BBITS);

The parameters in the calling sequence are:
IDENT  The identification of the light pen input buffer to be read.
XARRAY An array where the X coordinates of the end points of the line segments will be stored.
YARRAY An array where the Y coordinates of the end points of the line segments will be stored.
NPTS  The number of end points that were stored in XARRAY and YARRAY.
BBITS A bit string where the blanking bits will be stored.

The entry declaration for this procedure is:
DECLARE UGLGET EXTERNAL ENTRY (FIXED BINARY (15, 0),
(* FLOAT BINARY (21), (* FLOAT BINARY (21), FIXED BINARY (15, 0),
BIT (*));

The errors detected by this procedure are:
1(3): No device is active.
11(2): There is no light pen input buffer with the given identification.
12(2): There is not enough room in XARRAY, YARRAY, and BBITS for the light pen data.
99(2): This procedure is not supported for the active device.

SECTION 2.7: MISCELLANEOUS PROCEDURES

This section contains the descriptions of procedures which do not properly fit into any of the earlier sections.
SECTION 2.7.1: PROCEDURE UGDIDY

This procedure is an aid in using the procedure UGAXIS which was described in the section on Graphic Element Generation. Consider the following problem: suppose the extent of the data in one direction can only be determined at execution time and suppose that the program has determined that the data extends from 2.637 to 7.913. Usually the programmer does not want these values to label the axis but would prefer "round numbers" to label the axis. In this case, for instance, it is preferable to have the axis run from 2.00 to 8.00 with 7 labeled tic marks, or perhaps from 2.50 to 8.00 with 12 labeled tic marks. This procedure accepts as its input the extent of the data and limits on the number of labeled tic marks and produces values for the parameters LOLAB, NILAB, and NLAB in procedure UGAXIS which assures that all labeled tic marks are labeled with "round numbers".

The calling sequence is:

CALL UGDIDY (LODATA, HIDATA, NILAB, NAXLAB, LOLAB, NILAB, NLAB);

The parameters in the calling sequence are:

LODATA  The low extent of the data.
HIDATA  The high extent of the data.
NILAB  The minimum acceptable number of labeled tic marks.
NAXLAB  The maximum acceptable number of labeled tic marks.
LOLAB  A computed value which will be LODATA reduced to a "round number".
NILAB  A computed value which will be HIDATA increased to a "round number".
NLAB  A computed value which will make all of the labels "round numbers".

The entry declaration for this procedure is:

DECLARE UGDIDY EXTERNAL ENTRY (FLOAT BINARY (21),
   FLOAT BINARY (21), FIXED BINARY (15,0), FIXED BINARY (15,0),
   FLOAT BINARY (21), FLOAT BINARY (21), FIXED BINARY (15,0));

No error messages are produced by this procedure.

Algorithms of this nature have been described in the publications listed below. The algorithm used in UGDIDY is a modification of the one described in [2].


SECTION 2.7.2: PROCEDURE UGVIEW

This procedure may be used to define a projective transformation from three space into two space. The transformation may be used with procedure UGPROJ to project a point in three dimensional space into the programmer coordinate system. The meaning of the input to this procedure is illustrated in Figure 2.7.1. First a reference point, REFP, and a view direction, VDIR, are given. The reference point may be thought of as a point on the forehead of a person and the view direction as the direction in which the person is looking. Along the view direction, at a distance of SCRD, is a projection screen. This screen is a square with a side length of SCRZ and its horizontal axis is defined by the vector HDIR. Finally an eye point, E, is defined by moving a distance HYED from REFP in the direction of HDIR. The transformation takes a point P in three dimensional space and projects it by means of a straight line from E through P onto the screen to define the point Q. The point Q is the projective transform of P and its coordinates are (t,u).

The calling sequence is:
CALL UGVIEW (OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,TRANS);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
  HYED=<value> A parameter which is useful in making stereo pairs; a positive value will produce a right eye transformation and a negative value produces a left eye transformation. The default value is 0.0.
  YLO=<value> The Y coordinate at the left hand side of the screen. The default value is 0.0.
  YHI=<value> The Y coordinate at the right hand side of the screen. The default value is 1.0.
  HLO=<value> The Y coordinate at the bottom of the screen. The default value is 0.0.
  HHI=<value> The Y coordinate at the top of the screen. The default value is 1.0.

REFP The projective reference point.
VDIR The view direction.
HDIR The horizontal direction of the projection screen. If this vector is given as (0.0,0.0,0.0), then a vector parallel to the X-Y plane will be supplied.
SCRD The distance from REFP to the SCREEN.
SCRZ The size of the screen.
TRANS An array which will be set to the projective transformation. This array will contain a 3 by 4 projection matrix as well as the given projective parameters.
The entry declaration for this procedure is:

```plaintext
DECLARE UGVIEU EXTERNAL ENTRY (CHARACTER(9)),
(3) FLOAT BINARY(21), (3) FLOAT BINARY(21),
(3) FLOAT BINARY(21), FLOAT BINARY(21), FLOAT BINARY(21),
(31) FLOAT BINARY(21));
```

No error messages are produced by this procedure.

**Figure 2.7.1:** The definition of a Projective Transformation.

---

**SECTION 2.7.3: PROCEDURE UGORTH**

This procedure may be used to define an orthogonal transformation from three space into two space. The transformation may be used with procedure UGPROJ to project a point in three dimensional space into the programmer coordinate system. The meaning of the input to this procedure is essentially the same as for UGVIEU. The basic difference is that the point P is projected onto the screen parallel to the vector VDIR. The user will notice that the position of REFIP is not critical, it may be moved along VDIR without changing the transformation. Also the value of SCRZ is completely redundant. Nevertheless, the programmer should supply
reasonable values for these parameters because this information is saved in TRANS and is utilized by certain other procedures.

The calling sequence is:
CALL UGORTH (OPTIONS, REFP, VDIR, HDIR, SCRD, SCREW, TRANS);

The parameters in the calling sequence are:
OPTIONS A character string which may contain any of the following items:
   XLO=<value> The X coordinate at the left hand side of the screen. The default value is 0.0.
   XHI=<value> The X coordinate at the right hand side of the screen. The default value is 1.0.
   YLO=<value> The Y coordinate at the bottom of the screen. The default value is 0.0.
   YHI=<value> The Y coordinate at the top of the screen. The default value is 1.0.

REFP The orthogonal reference point.
VDIR The view direction.
HDIR The horizontal direction of the projection screen. If this vector is given as (0,0,0,0,0,0), then a vector parallel to the X-Y plane will be supplied.
SCRD The distance from REFP to the screen.
SCREW The size of the screen.
TRANS An array which will be set to the orthogonal transformation. This array will contain a 3 by 4 projection matrix as well as the given orthogonal parameters.

The entry declaration for this procedure is:
DECLARE UGORTH EXTERNAL ENTRY (CHARACTER (9),
   (3) FLOAT BINARY (21), (3) FLOAT BINARY (21),
   (3) FLOAT BINARY (21), FLOAT BINARY (21), FLOAT BINARY (21),
   (31) FLOAT BINARY (21));

No error messages are produced by this procedure.

SECTION 2.7.4: PROCEDURE UGPROJ

This procedure uses a transformation defined by procedure UGVVIEW or UGORTH to project a three dimensional point into two dimensions.

The calling sequence is:
CALL UGPROJ (TRANS, PT3D, PT2D);

The parameters in the calling sequence are:
TRANS An array containing the transformation.
PT3D The coordinates of the given three dimensional point.
PT2D The projected point.
The entry declaration for this procedure is:

DECLARE UGCTOL EXTERNAL ENTRY((3) FLOAT BINARY(21),
   (3) FLOAT BINARY(21), (2) FLOAT BINARY(21));

No error messages are produced by this procedure.

SECTION 2.7.5: PROCEDURE UGCTOL

This procedure will accept a character string as input, invoke the character stroke generator, and supply the resulting line segments and blanking bits as output. The line segments are made available in a form that is the same as that required by procedure UGELNS. Before passing these line segments on to UGELNS, the programmer may transform them in any manner. One simple thing a programmer could do is create italic lettering.

The calling sequence is:

CALL UGCTOL(OPTIONS,TEXT,XARRAY,YARRAY,NPTS,BBITS);

The parameters in the calling sequence are:

OPTION A character string which may contain any of the following items:
   X=<value> The X coordinate of the center of the first character. The default value is 0.0.
   Y=<value> The Y coordinate of the center of the first character. The default value is 0.0.
   SPACING=<value> The character spacing value which will be used to develop the strokes. The default value is 0.01367.
   ANGLE=<value> The angle that the characters will make with the horizontal. The default value is 0.0.

TEXT The string containing the characters to be converted to line segments.

XARRAY An array where the X coordinates of the end points of the line segments will be stored.

YARRAY An array where the Y coordinates of the end points of the line segments will be stored.

NPTS The number of end points that were stored in XARRAY and YARRAY.

BBITS A bit string where the blanking bits will be stored.

The entry declaration for this procedure is:

DECLARE UGCTOL EXTERNAL ENTRY(CHARACTER(*) ,CHARACTER(*),
   (*) FLOAT BINARY(21), (*) FLOAT BINARY(21), FIXED BINARY(15,0),
   BIT(*));

The errors detected by this procedure are:

1(2): The string TEXT contained more than 255 characters.
2(2): There is not enough room in XARRAY, YARRAY, and BBITS for the line segment data.
SECTION 2.7.6: PROCEDURE UGEFDIV

This procedure may be used to divide a previously generated element into two smaller elements. The primary purpose of this procedure is to divide elements that have been retrieved by procedure UGEGET from a partitioned data set. Such an element may be too large for display on a given display device. This procedure operates by first trying to divide the element at a boundary between different types of display data. Thus if an element contains line data followed by point data, the first divided element will contain only points and the second will contain only lines. If the original element contains only one type of data, then an attempt will be made to divide that data into two parts. Only point, line, and character data can be divided in this latter manner.

The calling sequence is:

CALL UGEFDIV(ELEMT1, ELEMT2);

The parameters in the calling sequence are:

ELEMT1 An array which contains the original element on input, and the first part of the divided element on output.

ELEMT2 An array which will contain the second part of the divided element on output. On input, this array must have been cleared by procedure UGEGET. In certain cases this array will have to be almost as large as the graphic element in ELEMT1 although it usually need only be about half as large.

The entry declaration for this procedure is:

DECLARE UGEFDIV EXTERNAL ENTRY((*) FIXED BINARY(31,0),
(* ) FIXED BINARY(31,0));

The errors detected by this procedure are:

1(3): The given element could not be divided.
There are two program libraries that the programmer must be aware of. One contains modules which must be made available to the LINK-EDITOR/LOADER, and the other contains modules which must be made available at execution time.

The first of these program libraries is WIL.CG.RCB.UGPLIB. In this library are all of the procedures that have been described in addition to some second level procedures that the programmer does not normally need. If the LINK-EDITOR is used, this data set should be concatenated with the SYSLIB data sets in the LINK step. If the LOADER is used, then it should be concatenated with the SYSLIB data sets in the GO step. The modules in this data set are essentially device independent.

The second program library is WIL.CG.RCB.UGPLIB. This library must be made available either to the job in a JOBLIB statement, or to the GO step in a STEPLIB statement. The modules in this library are loaded dynamically and contain most of the device dependent code.

There is a third library which is made available for the convenience of the PL/1 programmer. This data set is WIL.CG.RCB.UG2SRC. It is a source library and contains entry declaration statements for all of the procedures in this system. It is suggested that this library be concatenated with the SYSLIB data sets in the PL/1 step. If this is done, and the MACRO option is specified, then the PL/1 statement:

```
INCLUDE UGOPEN,UGCLOS;
```

would cause the entry declarations for procedures UGOPEN and UGCLOS to be inserted into the PL/1 program. The presence of these statements will assure that the correct types of arguments are passed to the procedures and will let the compiler look for some common programming errors.

The programmer may use the LINK-EDITOR to create overlay structures. Any of the modules in this package may be overlaid with one exception. This exception is that the module named UGx000 must not be overlaid while any device is open.
This section contains some examples of complete programs which use the SLAC Unified Graphics System. These examples include the complete JCL that is required to run the programs.

SECTION 4.1: PLOTTING A MESH SURFACE

This program produces a perspective view of a mesh surface on the CALCOMP Drum Plotter. The program, with its comments, is relatively self-explanatory except for two items. First, the extended character stroke generator with upper and lower case capability is used; this is described in the section on the Character Stroke Generating Sub-System. Second, an error processing procedure is used; this is described in the section on the Error Processing Sub-System. Without this error processing procedure, the array ELEMENT would have to be approximately seven times larger than its present size.

```plaintext
//SUURLSP JOB 'usu$qg',CLASS=B
 //EIC PLICLG,
 //PCPLIF='ATR,IBB,STXT,EST,MACRO'.
 //PL1LIB='YUL.CG.RCB.UGFTPNC'.
 //LKBFL='YUL.CG.RCB.UGP LIB',
 //GOSLIB='YUL.CG.RCB.UGLIB'
 //PL1_SYSIN DD *
/%%% SAMPLE PROGRAM: PLOTTING A MESH SURFACE %%%/

SURFACE: PROCEDURE OPTIONS (MAIN):

% INCLUDE UGHINT, UGELIN, UGETXT, UG3DHS;
% INCLUDE UGOPEN, UGCLOS;
% INCLUDE UGEPOT, UGPICT;
% INCLUDE UGVIEU;
% INCLUDE UGPTOX;

DECLARE REFP(3) PLOT BINARY(21) INITIAL (50, 75, 40);
DECLARE VDIR(3) PLOT BINARY(21) INITIAL (-50, -75, -30);
DECLARE HDIR(3) PLOT BINARY(21) INITIAL (0, 0, 0);
DECLARE TRANS(31) FLOAT BINARY(21);
DECLARE ARRAY(32, 52) FLOAT BINARY(21);
DECLARE WMEAN(780) FLOAT BINARY(21);
DECLARE ELEMENT(1000) FIXED BINARY(31, 0) EXTERNAL;
DECLARE C20 CHARACTER(20);
DECLARE (I, J) PLOT BINARY(21);
DECLARE (I, J) FIXED BINARY(31, 0);
```
/* INITIALIZE THE PROGRAM: OPEN THE GRAPHIC DEVICE AND
ASSURE THAT A FRESH PLOTTING AREA IS AVAILABLE, FILL
IN THE ARRAY WHICH DEFINES THE SURFACE, AND GENERATE
THE PROJECTIVE TRANSFORMATION. */
CALL UGOPEN("CALL/10D",99);
CALL UGPICX("CLEAR",0);
DO I=2 TO 52; X=I-27;
  ARRAY(1,X)=X;
  DO J=2 TO 32; Y=J-17;
  ARRAY(J,1)=Y;
  R=SQT(X**2+Y**2);
  ARRAY(J,1) = ((750/(R**2+75))+5)*COS(0.4*H);
END;
END;
CALL UGVIEW("",REFP,VDIR,HDIR,100,75,TRANS);
/* CREATE THE PICTURE: THE GRAPHIC ELEMENT IS CLEARED,
A BORDER IS ADDED TO THE ELEMENT, A TITLE IS ADDED,
THE SURFACE ITSELF IS ADDED TO THE ELEMENT, AND
FINALLY THE GRAPHIC ELEMENT IS TRANSMITTED TO THE
GRAPHIC DEVICE. */
CALL UGHEMT("CLEAR",ELEMENT);
CALL UGHEMT("",0,0,0,ELEMENT);
CALL UGHEMT("",1,0,1,ELEMENT);
CALL UGHEMT("",1,1,1,ELEMENT);
CALL UGHEMT("",0,0,1,ELEMENT);
CALL UGHEMT("",0,0,1,ELEMENT);
CALL UGEPXY("MESH SURFACE EXAMPLE",C20);
CALL UGETXT("SPACING=0.04",0.2,0.8,C20,ELEMENT);
CALL UG3DNS("UPPER","",ARRAY,TRANS,UKAREA,
  ELEMENT);
CALL UG3DNS("LOWER",NOCONNECT,"",ARRAY,TRANS,UKAREA,
  ELEMENT);
CALL UGEPFU("",0,ELEMENT);
/* TERMINATE THE PROGRAM: THE GRAPHIC DEVICE IS CLOSED
AND THE PROGRAM RETURNS TO THE SYSTEM. */
CALL UGCLOS(99);
RETURN;
END SURFACE;
* PROCESS("ATB,REF,STMT,HELP,MACRO");
/**** ERROR PROCESSING PROCEDURE ****/
UGERR: PROCEDURE (LEVEL,NAMES,INDEX);
DECLARE (LEVEL,INDEX) FIXED BINARY(15,0);
DECLARE NAME CHARACTER(8);
% INCLUDE UGEMT, UGEPFU;
DECLARE ELEMENT(1000) FIXED BINARY(31,0) EXTERNAL;
/* IF THE ERROR IS AN INDICATION THAT NO MORE SPACE IS
AVAILABLE IN THE GRAPHIC ELEMENT THEN THE CURRENT
CONTENTS OF THE ELEMENT ARE TRANSMITTED TO THE
GRAPHIC DEVICE, THE ELEMENT IS RE-INITIALIZED, AND THE ERROR INDICATOR IS RE-SET. */
IF INDEX=9 THEN DO:
    CALL UGEPUT(",0,ELEMENT):
    CALL UGEINT("CONTINUE",ELEMENT):
    LEVEL=0;
END:
RETURN;

END UGEERR;
/*LKED.SYSIN DD *
//INCLUDE SYSLIB (UGI103A)
//GO.PLOTAPP DD DSNAME=SGPLOT,DISP= (NEW, PASS),
// VOLUME=SER=PLOTAP,UNIT=17, LABEL=(1, HL),
// DCB= (RECFIL=0, BLKSIZE=3000, DBM=0)

Mesh Surface Example

Figure 4.1.1: The Mesh Surface.
SECTION 4.2: INTERACTIVE LISSAJOUS FIGURES

This program is an interactive program which runs on the IBM 2250 Display Console. The first picture which appears on the screen contains some descriptive material, a keyboard input buffer, and a light pen message to terminate the program. The console operator may use the light pen to terminate the program or he may modify the curve parameters in the keyboard input buffer and generate a keyboard attention. After a keyboard attention is generated, a second picture will appear on the screen. The second picture contains the graph of the figure and two light pen messages. The console operator may use the light pen to delete the coordinate axes in the picture or he may return to the first picture.

Notice that this program always causes the display to be turned off when an attention occurs and never turns the display back on until a complete picture is available. This results in two desirable features. First, the picture will not flicker on the screen while elements are being changed. Second, the appearance of a picture signals the console operator that the program is waiting for him.

This program illustrates an important principle which should be followed in writing interactive programs; it is especially important that interactive programs do not terminate abnormally because the console operator has made a minor error. In this program, for instance, it is very difficult for the console operator to enter information into the keyboard input buffer which is invalid or which will cause an undetected conversion error.

```c
//uuuLSAJF JOB 'uuuSgg',CLASS=0
// EXEC P1CG,
//   PL1FBR='ATR,RELP,STRT,NEST,HACRO',
//   PL1LIB='WIL.CG.RCB.UGPSRC',
//   LEBELIB='WIL.CG.RCB.UGPLIB',
//   GOSLIB='WIL.CG.RCB.UGILIB'
//PL1.SYSIN DD *
/** Sample Program: Interactive LiLissajous Figures **/

LISSAJ: PROCEDURE OPTIONS(HAIB);

% INCLUDE UGEINT, UGELIN, UGETXT, UGAXIS;
% INCLUDE UGOPEN, UGCLOS;
% INCLUDE UGERPUT, UGPICT;
% INCLUDE UGERATN, UGDATA, UGREAT;
% INCLUDE UGERPUT, UGGET;

DECLARE 1 ATTND,
   2 ATCODE CHARACTER(4),
   2 IARRAY(5) FIXED BINADY(15,0),
   2 IARRAY(2) FLOAT BINADY(21);
DECLARE PAMHS CHARACTER(20);
DECLARE VRAM CHARACTER(20) VARYING;
DECLARE XGRAM CHARACTER(22);
```
DECLARE ELEMENT(1000) FIXED BINARY(31,0);  
DECLARE BIT BIT(1);  
DECLARE (A,X,Y) FLOAT BINARY(21);  
DECLARE (B,X,BIT,ST,1) FIXED BINARY(15,0);  
/* INITIALIZE THE PROGRAM: OPEN THE GRAPHIC DEVICE, STOP  
THE DISPLAY, ENABLE THE LIGHT PEN, GENERATE THE TITLE,  
AND INITIALIZE THE LISSAJOUS FIGURE PARAMETERS. */  
CALL UGOPEN ('IBM2250',99);  
CALL UGPICT ('OFF',0);  
CALL UGCHCT ('LPEN',OFF);  
CALL UGCHLT ('CLEAR',ELEMENT);  
CALL UGCHTT ('LABS',0.244,0.97,  
'LISSAJOUS FIGURE GENERATOR',ELEMENT);  
CALL UGCLIN ('",0.0,0.05,'"0',B,ELEMENT);  
CALL UGCLIN ('",1.0,0.05,'"1',B,ELEMENT);  
CALL UGECUT ('",0,ELEMENT);  
STD1: B=2; H=3; A=0.25;  
/* PUT PARAMETER REQUEST ON THE SCREEN: THE DISPLAY  
CONSISTS OF A DESCRIPTION OF THE LISSAJOUS FIGURES, A  
KEYBOARD INPUT BUFFER FOR ENTERING THE PARAMETERS, AND  
A LIGHT PEN MESSAGE FOR TERMINATING THE PROGRAM. */  
STD2: CALL UGCHLT ('CLEAR',ELEMENT);  
CALL UGCHTT ("SHAL",0.199,0.7,  
'THE EQUATION OF A LISSAJOUS FIGURE IS: ',ELEMENT);  
CALL UGCHTT ("SHAL",0.269,0.66,  
'I=COS(2*PI(N*T/H+A)) ',ELEMENT);  
CALL UGCHTT ("SHAL",0.269,0.66,  
'I=COS(2*PI(T)) ',ELEMENT);  
CALL UGCHTT ("SHAL",0.199,0.5,  
'YOU MAY NOW CHANGE THE PARAMETERS H, W, AND A',  
ELEMENT);  
CALL UGCHTT ("SHAL",0.199,0.48,  
'AND GENERATE A KEYBOARD ATTENTION; OR YOU MIGHT',  
ELEMENT);  
CALL UGCHTT ("SHAL",0.199,0.46,  
'LIGHT PEN THE TERMINATE MESSAGE TO QUIT. ',  
ELEMENT);  
CALL UGEPUT ("",10,ELEMENT);  
PUT STRING(PARMS) EDIT(H,W,A) (2 F(5),P(10,4));  
CALL UGEPUT ("SHAL",0.371,0.3,11,PARMS);  
CALL UGCHLT ('CLEAR',ELEMENT);  
CALL UGCHTT ("SHAL",0.7,0.025,'TERMINATE',ELEMENT);  
CALL UGEPUT ('ON,DETC',1,ELEMENT);  
/* PROCESS OPERATOR ACTION: THE KEYBOARD IS ENABLED AND  
THE PROGRAM WAITS FOR AN ATTENTION. WHEN AN ATTENTION  
IS RECEIVED, THE KEYBOARD IS DISABLED. A LIGHT PEN  
ATTENTION CAUSES THE PROGRAM TO TERMINATE. A KEYBOARD  
ATTENTION CAUSES THE INPUT BUFFER TO BE READ, THE  
PICTURE TO BE CLEARED, AND THE INPUT BUFFER TO BE  
CONVERTED TO NUMERICS AND CHECKED. THE CONVERSION ON  
CONDITION IS USED TO CHECK FOR CONVERSION ERRORS. */  
CALL UGCHTN ('KBRD,OFF');  
CALL UGCHTN (-1,ATTND);  
CALL UGCHTN ('KBRD');
IF ATTND.ATCODE='LPEN' THEN GO TO STN5;
CALL UGETxt (11, VPARH, I);
CALL UGPICT ('CLEAR', 1);
CALL UGPICT ('CLEAR' , 11);
CALL UGPICT ('CLEAR' , 10);
VPARH=VPARN !" X'
CONVERSlON GO TO STN1;
GET STRING (VPARN) LIST (M, H, A);
REVERv CONVERSION;
 IF (H<=0) OR (N<=0) THEN GO TO STN1;
/
CREATE THE LISSAJOU5 FIGURE. THE FIGURE IS GENERATED
BY LETTING THE PARAMETER T RUN FROM ZERO TO (N DIVIDED
BY THE GREATEST COMMON DIVISOR OF N AND H). THEN A
HORIZONTAL AND VERTICAL AXIES AND TWO LIGHT PEN
MESSAGES ARE ADDED TO THE PICTURE. */
MT=H; NT=H;
STN3: IF MT>NT THEN DO;
     I=MT; NT=MT; MT=I;
END;
I=NT/NT; NT=NT-NT*I;
IF NT->0 THEN GO TO STN3;
MAX=N/MT;
CALL UGPICT ('CLEAR', ELEMENT); BEIT=' 0'B;
DO I=1 TO 401;
   T=FLOAT ((I-1) *MAX)/400;
   X=COS (6.2831853* ((H*T)/H)+A));
   Y=COS (6.2831853*T);
   CALL UGPICT ('SHAL' ,0.4*X*0.5,0.4*Y*0.5, BEIT, ELEMENT); 
   BEIT='U'B;
END;
CALL UGETxt ('', 10, ELEMENT);
CALL UGPICT ('CLEAR', ELEMENT); 
CALL UGPICT ('HORIZ', NST=4, '', 'SHAL', 0.5, 0.1, 0, 0.9, 
           -1.0, 1.0, 0, 5, ELEMENT); 
CALL UGPICT ('VERT', NST=4, '', 'SHAL' ,0.5, 0.1, 0, 9, 
           -1.0, 1.0, 0, 5, ELEMENT); 
CALL UGETxt ('', 11, ELEMENT);
CALL UGPICT ('CLEAR', ELEMENT);
CALL UGETxt ('SHAL' ,0.2, 0.025, 'NEW FIGURE', ELEMENT); 
CALL UGETxt ('DETC', 0, ELEMENT); 
CALL UGETxt ('CLEAR', ELEMENT); 
CALL UGETxt ('SHAL', 0.7, 0.025, 'DELETE AXES', ELEMENT); 
CALL UGETxt ('ON', DETC', 0, ELEMENT); 
/
PROCESS OPERATOR ACTION: THE PROGRAM WAITS FOR AN
ATENTION. A LIGHT PEN DETECT ON THE 'DELETE AXES'
MESSAGE CAUSES THE AXES ONLY TO BE DELETED, OTHERWISE
THE ENTIRE PICTURE IS DELETED AND THE FIRST DISPLAY IS
RESTORED. */
STN4: CALL UGPICT (-1, ATTND); 
CALL UGPICT ('CLEAR', 2); 
CALL UGPICT ('CLEAR', 11); 
IF ATTND.TABRAT (2)=2 THEN DO;
   CALL UGPICT ('ON', 0);
   GO TO STN4;
END;
CALL UGPICT('CLEAR',1);
CALL UGPICT('CLEAR',10);
GO TO STK2;

/* TERMINATE THE PROGRAM: THE GRAPHIC DEVICE IS CLOSED 
AND THE PROGRAM RETURNS TO THE SYSTEM. */
STK5: CALL UGCLOSE(99);
RETURN;

END LISSAJ;

//GO.SCOPE50 DD UNIT=0D1

LISSAJOUS FIGURE GENERATOR

THE EQUATION OF A LISSAJOUS FIGURE IS:
X=CB5T(2*PI*(M*T/N+A))
Y=CB5T(2*PI*T)

YOU MAY NOW CHANGE THE PARAMETERS M, N, AND A 
AND GENERATE A KEYBOARD ATTENTION; OR YOU MAY
LIGHT PEN THE TERMINATE MESSAGE TO QUIT.

2 3 0.2500

Figure 4.2.1: The first picture showing suggested parameters.
Figure 4.2.2: The second picture when suggested parameters are used.
This section gives a detailed description of each of the non-
interactive devices that are supported by this system. For each
device there is a description of the hardware including its
features and limitations, and a review of the capabilities of
each procedure as it applies to the device. In particular the
valid and default items in the OPTIONS list for each procedure
are given.

SECTION A.1: THE CALCOMP DRUM PLOTTERS (MODELS 564/565)

The CALCOMP Drum Plotters are off-line electro-mechanical
plotting devices. A program which produces pictures for a
CALCOMP Drum Plotter will write plotting commands on a 7-track
magnetic tape. This tape is then mounted on a CALCOMP plotter
which reads the commands on the tape and draws the pictures.

Two different CALCOMP units are available. The first draws on
paper that is 10 inches wide (the Y direction) and essentially
unlimited in length (the X direction). The pen is moved
incrementally in the X and Y directions in steps of 0.01 inches.
The second plotter differs in that the paper is 29 inches wide
and its step size is 0.005 inches. These devices do not have a
built-in character generator.

Special attention should be given to the construction of a
graphic element if it is to be transmitted to a CALCOMP Drum
Plotter. The reason is that this device is quite slow; a
complicated picture can require hours of plotting time.
Therefore the programmer should try to minimize pen motion.
Operations such as moving from the lower left section of a
picture to the upper right, drawing a few lines, and then moving
back to the lower left should clearly be avoided.

A DD card is necessary at execution time to define the output
data set (normally a tape). A typical DD card is:

```c
//PLOTTAPE DD DSNNAME=66PLOT,DISP=(NEW,PASS),
  // VOLUME=SER=PLOTAP,UNIT=T7,LABEL=(1,NL),
  // DCB=(RECFM=U,BLKSIZE=3000,DEH=0)
```

The ddname, PLOTTAPE in the example, must correspond to the value
given in the DSNNAME item in the options list of UGOPEN. The
BLKSIZE parameter in the DCB must be at least 256; a value around
3000 is suggested.

The procedures which may be used with this device are:
CALL UGOPEN(OPTIONS, IDENT);

The valid items in OPTIONS are:
- CAL/10D  Specifies the 10 inch plotter.
- CAL/29D  Specifies the 29 inch plotter.
- DDNAME=<value> The ddbname of the output tape. The default value is PLOTTAPE.
- TITLE=<value> A character string of at most 16 characters which will be used to identify the output. Characters 1-8 and 10-16 are used to prepare an initial plot to identify the output. The default is a title composed of the job name and output bin number. The programmer should not normally change the default value.

CALL UGCLOS(IDENT);
CALL UGSLCT(IDENT);
CALL UGSCAL(OPTIONS, PRGCLS, DEVCLH);

The default values for the programmer and device limits for the
10 inch plotter are:
\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\quad \begin{bmatrix}
0 & 1000 \\
0 & 1000 \\
\end{bmatrix}
\]

Those for the 29 inch plotter are:
\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\quad \begin{bmatrix}
0 & 5800 \\
0 & 5800 \\
\end{bmatrix}
\]

CALL UGINFDATA(STRING, IDENT, ARRAY);

The array ARRAY will contain the following:
- ARRAY(4) The number of words used in the largest graphic element that has been transmitted to the device.
- ARRAY(2) .. ARRAY(5) Zero.

CALL UGEFNT(OPTIONS, IDENT, ELEMENT);

For this device, the contents of both OPTIONS and IDENT are ignored.

CALL UGPICT(OPTIONS, IDENT);

The only valid item in OPTIONS is CLEAR. The contents of IDENT is ignored and assumed to be zero. This clearing operation causes the pen to move over to a fresh plotting area where the programmer may begin another picture. There will be about ten inches of blank paper between pictures.
In addition to the errors described with each procedure, there are some other errors which may be detected. These errors are identified by a procedure name of UGCALD02 and their description is:

\[ N(\theta) \]: (Where \( N \) is greater than 255.) An extremely unusual error condition has been detected.

SECTION A.2: THE CALCOMP MICROFILM PLOTTER (MODEL 1675)

The CALCOMP Microfilm Plotter at SLAC is a device with a 5 inch, flat-faced, CRT. This CRT is photographed by a camera which can contain either 16mm unsprocketed film or 105mm Microfiche film. Pictures may be transmitted to the device off-line by writing the pictures on a magnetic tape and mounting the tape on the device, or on-line by transmitting the pictures directly over a channel.

This unit can display points, lines and characters. The manufacturer claims that 30 intensity levels are possible, but these cannot normally be distinguished by the eye and only four intensity levels are supported by this system. The character generator can produce horizontal characters of a single size. The following table gives the between character spacing and suggested between lines spacing assuming the default programmer coordinate system.

<table>
<thead>
<tr>
<th>Character Size</th>
<th>Character Spacing</th>
<th>Line Spacing</th>
<th>Characters per Line</th>
<th>Lines per Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAL</td>
<td>0.00916</td>
<td>0.01500</td>
<td>109</td>
<td>66</td>
</tr>
</tbody>
</table>

When the PULSCR option is used in procedure UGOPEN, then 132 characters will fit on a single line. The LABGSCR option results in additional characters per line and lines per display being available.

A DD card is necessary at execution time to define the output data set. A typical DD card for off-line operation is:
On-line operation is not yet possible at SLAC. The ddname, PLOTDATA in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEN.

Microfiche contains 63 pictures per fiche arranged in 7 rows and 9 columns. These pictures are ordered by columns; the first picture is at the upper right, the second picture is below it, etc.

The procedures which may be used with this device are:

CALL UGENV (OPTIONS, ELEMENT);
CALL UGEPUT (OPTIONS, Y, ELEMENT);
CALL UGELIN (OPTIONS, X, Y, ELEMENT);
CALL UGETIT (OPTIONS, X, Y, TEXT, ELEMENT);
CALL UGETPS (OPTIONS, ARRAY, XARRAY, NPTS, ELEMENT);
CALL UGELNS (OPTIONS, ARRAY, XARRAY, NPTS, BBIT, ELEMENT);

When an element is transmitted to this device, all wink mode and color information is ignored. Intensity levels of VDIM, DMIN, BBIT, and VBIT will be correctly processed. All lines are of the SOLD type. Horizontal text with the VSML, SHAL, LANG, or VING options will be produced by the character generator.

CALL UGAXIS (OPTIONS, LOPTIONS, COPTIONS, INDVAR, LOVAR, HVAR,
             LOLAB, HLAB, NLAB, ELEMENT);
CALL UG3DBS (OPTIONS, LOPTIONS, ARRAY, TRANS, UKAREA, ELEMENT);
CALL UGDATA (OPTIONS, STRING, ELEMENT);

CALL UGOPEN (OPTIONS, IDENT);

The valid items in OPTIONS are:

CAL/16M  Specifies 16mm Unspockeded Film.
CAL/PCH  Specifies Microfiche Film.
DDNAME=<value>  The ddname of the output data set. The
default value is PLOTDATA.
FULSCR, LANGSCR  Flags which specify the size of the
image on the output. For 16mm the default is a square
picture which is the correct size for viewing on SLAC's
microfilm viewers. FULSCR means that the full
rectangular area visible with the viewer may be used.
LANGSCR means that the largest possible square picture
will be produced. Finally, FULSCR and LANGSCR together
give the maximum rectangular plotting area on the film.
For microfiche, the default is a square picture and
FULSCR produces the larger rectangular plotting area.
LEADER=<value>  The number of frames of leader which will
be put at the beginning and end of film output. The
default value is 20. The programmer should not
normally change the default value.
TITLE=<value>  A character string of at most 24
characters which will be used to identify the output.
These 24 characters will appear at the top of each
microfiche and part of them (characters 1-9 and 10-16)
is used in the leader for film output. The default is
a title composed of the job name and output bin number.
The programmer should not normally change the default
value.
CALL UGCLOSE(IDENT);
CALL UGSCLCT(IDENT);
CALL UGSCLD(OPTIONS,REGCLE,DEVCLE);
The default values for the programmer and device limits for
16mm unsprocketed files are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
4260 & 12123 \\
4260 & 12123 \\
\end{bmatrix}
\]

When FULSCR is used, the values are:

\[
\begin{bmatrix}
-0.10\times10^2 & 1.10\times10^2 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
3440 & 12943 \\
4260 & 12123 \\
\end{bmatrix}
\]

When LARGSCH is used, the values are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
2785 & 13598 \\
2785 & 13598 \\
\end{bmatrix}
\]

When FULSCR and LARGSCH are used, the values are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
-0.09868 & 1.09868 \\
\end{bmatrix}
\begin{bmatrix}
1638 & 14581 \\
\end{bmatrix}
\]

The default values for the programmer and device limits for
microfiche files are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
4260 & 12123 \\
4260 & 12123 \\
\end{bmatrix}
\]

When FULSCR is used, the values are:

\[
\begin{bmatrix}
-0.10\times10^2 & 1.10\times10^2 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
3440 & 12943 \\
4260 & 12123 \\
\end{bmatrix}
\]

CALL UGINFO(STRING,IDENT,ARRAY);
The array ARRAY will contain the following:
ARRAY(1) The number of words used in the largest graphic
element that has been transmitted to the device.
ARRAY(2) ...ARRAY(5) Zero.
CALL UGEPUT(OPTIONS,IDENT,ELEMENT);
For this device, the contents of both OPTIONS and IDENT are
ignored.
CALL UGPPCT(OPTIONS,IDENT);
The only valid item in options is CLEAR. The contents of IDENT
is ignored and assumed to be zero. This clearing operation
causes the file to advance.

CALL UGDXDY(LODATA,HICLTA,MINLAB,MAXLAB,LOLAB,HILAB,NLAB);
CALL UGVWEB(OPTIONS,REPP,VDIB,EDIB,SCRD,SCRE,TRANS);
CALL UGSORTB(OPTIONS,REPP,VDIB,EDIB,SCRD,SCRE,TRANS);
CALL UGPROJ(TRANS,PT3D,PT2D);
CALL UGCTOL(OPTIONS,TRIT,ARRAY,YARRAY,NPTS,BBITS);
CALL UGEDIIV(ELEMENT1,ELEMENT2);
CALL UGPT02(CPP,CPS,GC);
CALL UGXTOP(CPP,CPS,GC);
CALL UGI901(LEVEL,HAME,INDEX);
CALL UGI902(OPTIONS,INDATA,HIDATA);

In addition to the errors described with each procedure, there are some other errors which may be detected. These errors are identified by a procedure name of UGCALH02 and their description is:

* E(N): (Where N is greater than 255.) An extremely unusual error condition has been detected.

### SECTION A.3: WYLBUR DISPLAY FILES FOR THE TRESTHONIX 4013

The TRESTHONIX 4013 is an interactive storage display terminal with an eight inch wide by six inch high screen. This section describes a method of using this device in an essentially noninteractive way in conjunction with the SLAC/WYLBUR Text Editing System.

To use the TRESTHONIX 4013 in this manner, the programmer runs a job which uses this device sub-system. That job will write its pictures into a card image partitioned data set with each picture as a separate member. The user may then sign onto WYLBUR from a TRESTHONIX 4013 terminal and issue a 'USE' command to obtain a member of the data set as the active file. The WYLBUR command 'LIST UNUMBERED CLEAN' will then cause the screen to be cleared and the picture drawn. The audible alarm will sound when the picture is complete.

These units can display points, lines, and characters at a single intensity level. The character generator can produce horizontal characters of a single size. The following table gives the between character spacing and suggested between lines spacing assuming the default programmer coordinate system.

<table>
<thead>
<tr>
<th>Character Size</th>
<th>Character Spacing</th>
<th>Line Spacing</th>
<th>Characters per Line</th>
<th>Lines per Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMD</td>
<td>0.01797</td>
<td>0.02824</td>
<td>56</td>
<td>35</td>
</tr>
</tbody>
</table>

When the PULSCHR option is used in procedure UGOPEN, then 74 characters will fit on a single line.

A DD card is necessary at execution time to define the output data set. A typical DD card is:

```bash
// WYLPCT DD DSN=NAME,DISP=(NEW,KEEP),
// VOLUME=SER=N1001,UNIT=2314,SPACE=(TEN), (5,1,3))
// DCEP=(BECFP=F,F,LEN=80,_BLKSIZE=1600)
```
The ddname, WILSPICT in the example, must correspond to the value given in the DDBNAME item in the options list of UGOPEN. The BLKSIZE parameter in the DCB can be any multiple of 80.

The members of the output data set contain information in card images with at most 72 characters per card. The first card contains a title identifying it as a WILBUR display file, a clear screen command, and some idle characters. The second card contains only idle characters. The picture itself starts on the third card. If you are displaying the picture on a dial-up terminal (300 baud) you may delete the second card before listing the file. If you are using a hard-wired terminal (1200 baud) you will need the extra time supplied by the additional idles in the second card for the screen to fully clear.

The procedures which may be used with this device are:

CALL UGEINT(OPTIONS,ELEMENT);
CALL UGEINT(OPTIONS,X,Y,ELEMENT);
CALL UGELIN(OPTIONS,X,Y,ELEMENT);
CALL UGETXY(OPTIONS,X,Y,TXT,ELEMENT);
CALL UGETPTS(OPTIONS,XAB000,YAB000,EPJTS,ELEMENT);
CALL UGELNS(OPTIONS,XAB000,YAB000,EPJTS,EPJTS,ELEMENT);

When an element is transmitted to this device, all intensity level, blink mode, and color information (with one exception) is ignored. The exception is that points with the DINH or VDIN option are plotted as a single raster point while all other points are plotted as four raster points. All lines are of the SOLD type. Horizontal text with the VSAL, SMAL, LARG, or VINT options will be produced by the character generator.

CALL UGAXIS(OPTIONS,LOPTIONS,COPTIONS,INDVAX,LOVAX,LIVAX,
LOLAB,ILAB,ELAB,ELEMENT);
CALL UG3DHS(OPTIONS,LOPTIONS,ARRAY,TRANS,UKAREA,ELEMENT);

CALL UGOPEN(OPTIONS,IDENT);

The valid items in OPTIONS are:

WIL4013 Specifies that WILBUR Display Files for the
TEKTRONIX 4013 are to be produced.

DDBNAME=<value> The ddname of the partitioned data set.
The default value is WILSPICT.

PICTID=<value> The first four characters of the member
names in the PDS. If the given value is less than four
characters long, it is padded on the right with zeros.
The default value is PICT.

PICTSQ=<value> An integer which will be converted to
three numeric digits and concatenated to the PICTID
value to form the complete MEMBER NAME in the PDS.
This value is incremented after each picture. The
default value is 1.

FULLSCR A flag which specifies that the picture is to be
scissored at the screen boundaries and not at a square
area within the screen.

If the full member name of the picture is the same as a pre-
existing member of the data set, then the member is replaced;
otherwise the new member is added to the data set. The disk
space occupied by replaced members is not made available for
re-use until the data set is compressed. UGPICY may be used to
supply an alias for the name of a member.
CALL UGCLOS(IDENT);
CALL UGSCTT(IDENT);
CALL UGSCAL(OPTIONS,PROGLH,DEVCLH);
The default values for the programmer and device limits are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
122 & 901 \\
0 & 779 \\
\end{bmatrix}
\]

The default values when PULSCR is used are:

\[
\begin{bmatrix}
-0.15661 & 1.15661 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 1023 \\
0 & 779 \\
\end{bmatrix}
\]

CALL UGINFO(STRING,IDENT,ARRAY);
The array ARRAY will contain the following:
ARRAY(1) The number of words used in the largest graphic
element that has been transmitted to the device.
ARRAY(2) ...ARRAY(4) Zero.
ARRAY(5) The number of card images which have been
produced.

CALL UGEPUT(OPTIONS,IDENT,ELEMENT);
For this device, the contents of both OPTIONS and IDENT are
ignored.
CALL UGPICT(OPTIONS,IDENT);
The only valid items in OPTIONS are CLEAR and ALIAS. The
contents of IDENT is ignored and assumed to be zero. This
clearing operation causes a new picture to be started.

CALL UGDEXY(LODATA,HIDATA,HILAB,HAXLAB,LOLAB,HLAB,MLEL);
CALL UGVIEW(OPTIONS,HEPP,VDIA,NDIA,SCRD,SCRZ,TRANS);
CALL UGORTH(OPTIONS,HEPP,VDIA,NDIA,SCRD,SCRZ,TRANS);
CALL UGPROJ(TRANS,PT3D,PT2D);
CALL UGCTOL(OPTIONS,TEIT,XARRAY,YARRAY,WPTS,BEITS);
CALL UGHDIV(ELEMENT1,ELEMENT2);

CALL UGPOTOI(CFP,CPS,EC);
CALL UGSTOP(CFP,CPS,EC);

CALL UGX001(LEVEL,NAIL,INDEX);
CALL UGX002(OPTIONS,INCATA,EXDATA);

In addition to the errors described with each procedure, there
are some other errors which may be detected. These errors are
identified by a procedure name of UGNUX02 and their description
is:

0(N): Not enough directory blocks are available in the output
data set.
N(N): (Where N is greater than 255.) An extremely unusual
error condition has been detected.
SECTION A.4: SAVING PICTURES IN A PARTITIONED DATA SET

This section does not describe a display device but instead describes a method by which pictures may be saved in a partitioned data set. The graphic elements are written into the PDS without modification. A method is provided to read the graphic elements from the PDS. A program which makes use of this feature will appear the same as a program which prepares pictures for any non-interactive display device; only the call to UGOPEN will be different.

Each graphic element will be a separate record in the output data set and each picture will be a separate member. In addition, the programmer scaling and scissoring units are saved in the data set when the first element is written. Therefore, under most circumstances, the programmer should create the pictures so that they all depend on the same scaling limits and these limits should be set by UGSCAL before the first element is saved in the data set.

To read a data set written in this manner, the programmer should open the data set with the INPUT flag and use UGEOPT to obtain the graphic elements. When the first graphic element of a picture is read, the programmer scaling and scissoring limits are also read. The programmer may then use UGSCAL to retrieve these limits.

A DD card is necessary at execution time to define the output data set. A typical DD card is:

```
//PICTURES DD DSNNAME=VYL.QQ.UU.NAME,DISP=(NEW,KEEP),
// VOLUME=SER=VYL001,UNIT=2314,SPACE=(TRK,(5,1,3)),
// DCB=(RECFM=U,RECSIZE=4000)
```

The ddname, PICTURES in the example, must correspond to the value given in the DSNNAME item in the options list of UGOPEN. The RECSIZE parameter in the DCB must be at least as large as the largest graphic element which will be written into the data set. Remember that the RECSIZE value is in bytes while graphic element length is declared in words.

The procedures which may be used with this device are:

- CALL UGEINT(OPTIONS,ELEMENT);
- CALL UGEINIT(OPTIONS,X,Y,ELEMENT);
- CALL UGELIN(OPTIONS,X,Y,LEFT,ELEMENT);
- CALL UGEXIT(OPTIONS,X,Y,RIGHT,ELEMENT);
- CALL UGEPTS(OPTIONS,XARRAY,YARRAY,NPTS,ELEMENT);
- CALL UGELS(OPTIONS,XARRAY,YARRAY,NPTS,BRITS,ELEMENT);

When an element is transmitted, all intensity level, wink mode, color, character size, and line structure information is transmitted without modification.

- CALL UGAXIS(OPTIONS,LOPTIONS,COPTIONS,INDVAR,LOVAR,HIVAR,
  LOLAB,HI LAB,LAB,ELEMENT);
- CALL UG3DMS(OPTIONS,LOPTIONS,ARRAY,TRANS,UKAREA,ELEMENT);

- CALL UGOPEN(OPTIONS,IDENT);

The valid items in OPTIONS are:
DISKPDS specifies that the pictures are to be saved or retrieved from a partitioned data set in un-modified form.
 DDNAME=<value> The ddname of the partitioned data set. The default value is PICTURES.
 PICTID=<value> The first four characters of the member names in the PDS. If the given value is less than four characters long, it is padded on the right with zeros. The default value is PICT.
 PICTSQ=<value> An integer which will be converted to three numeric digits and concatenated to the PICTID value to form the complete member name in the PDS. This value is incremented after each picture. The default value is 1.
 INPUT A flag which specifies that the partitioned data set is to be read. The default is to open the data set for output.
 If the full member name of the picture is the same as a pre-existing member of the data set, then the member is replaced; otherwise the new member is added to the data set. The disk space occupied by replaced members is not made available for re-use until the data set is compressed. UGPPICT may be used to supply an alias for the name of a member.
 CALL UGCLOS(IDENT);
 CALL UGSLET(IDENT);
 CALL UGSCAL(OPTIONS, PICCLE, DEVCLN);
 The default values for the programmer and device limits are:

\[
\begin{array}{c}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{array}
\]

CALL UGENLIN(STRING, IDENT, ARRAY);
 The array ARRAY will contain the following:
 ARRAY(1) The number of words used in the largest graphic element that has been transmitted to the device.
 ARRAY(2) ... ARRAY(5) Zero.

CALL UGSPUT(OPTIONS, IDENT, ELEMENT);
 For this device, the contents of both OPTIONS and IDENT are ignored.
 CALL UGPPICT(OPTIONS, IDENT);
 The only valid items in OPTIONS are CLEAR and ALIAS. The contents of IDENT is ignored and assumed to be zero. This clearing operation causes a new picture to be started.
 CALL UGSPGET(OPTIONS, ELEMENT);

CALL UGDXYDY(LODATA, HIFDATA, HINLAB, HALAB, LOLAB, HILLAB, HLAB);
 CALL UGVIEW(OPTIONS, REFP, VDIR, HDIR, SCRD, SCRE, TRANS);
 CALL UGORTH(OPTIONS, REFP, VDIR, HDIR, SCRD, SCRE, TRANS);
 CALL UGPRESQ(TRANS, PT3D, PT2D);
 CALL UGCTOL(OPTIONS, THET, XARRAY, YARRAY, NPTS, BEHTS);
 CALL UGDIV(ELEMT1, ELEMT2);

CALL UGTOX(CFP, CPS, EC);
 CALL UGTOP(CFP, CPS, EC);
CALL UGX001(LEVEL,NAME,INDEX);
CALL UGX002(OPTIONS,INDATA,EXDATA);

In addition to the errors described with each procedure, there are some other errors which may be detected. These errors are identified by a procedure name of UGBPDS02 and their description is:

0(4): Not enough directory blocks are available in the output data set.
1(4): An error has been detected while trying to read the data set.
N(4): (Where N is greater than 255.) An extremely unusual error condition has been detected.
This section gives a detailed description of each of the interactive devices that are supported by this system. For each device there is a description of the hardware including its features and limitations, and a review of the capabilities of each procedure as it applies to the device. In particular the valid and default items in the OPTIONS list for each procedure are given.

SECTION B.1: THE IBM 2250 DISPLAY CONSOLE

The IBM 2250 is an on-line interactive display console. There are three IBM 2250/Model 2 consoles at SLAC. These devices consist of a CRT with a 10 inch by 10 inch working area, an alphanumeric keyboard, and a light pen.

The IBM 2250's are connected to the central computer system through an IBM 2840 display controller. This unit contains a 16,384 byte buffer which is shared by the three display consoles. A program running on the central computer system and using an IBM 2250 can write display orders into an area of the buffer allocated to the display. The orders present in the buffer constitute a "display file". In order to keep a picture on the screen, it is re-generated from the display file 40 times a second.

These units can display points, lines, and characters at a single intensity level. The character generator can produce horizontal characters of two sizes. The following table gives the between character spacing and suggested between lines spacing assuming the default programmer coordinate system.

<table>
<thead>
<tr>
<th>Character Size</th>
<th>Character Spacing</th>
<th>Line Spacing</th>
<th>Characters per Line</th>
<th>Lines per Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALL</td>
<td>0.01367</td>
<td>0.01953</td>
<td>74</td>
<td>52</td>
</tr>
<tr>
<td>LARGE</td>
<td>0.02051</td>
<td>0.02930</td>
<td>49</td>
<td>35</td>
</tr>
</tbody>
</table>

A DD card is necessary at execution time to select the display console to be used. A typical DD card is:

```
//SCOPES50 DD UNIT=0D1
```

The ddname, SCOPES50 in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEM. The device address, 0D1 in the example, is the address of the display console assigned to the "0" job class.
The keyboard on the IBM 2250's contains a number of special keys. The key labeled JUMP may be used to move the cursor from one keyboard input buffer to another. There are also keys labeled BACKSPACE and ADVANCE for moving the cursor within a keyboard input buffer without changing the content of the buffer. If the CONTINUE key is held down and another key is depressed, the function of the second key is repeated 40 times a second. The CONTINUE key is especially useful when used with the BACKSPACE or ADVANCE keys to move the cursor around. The keyboard attention is generated by holding the ALT key down and depressing the key labeled END.

The IBM 2250's at SLAC have a serious problem that the console operator should be aware of. If the cursor is not present and a key on the keyboard is depressed, the entire unit becomes locked and nothing will operate correctly including the light pen. The unit may be unlocked by holding the SHIFT key down and depressing the ALT key.

The light pen attention is generated by pointing at a light pen detectable element and depressing the foot pedal. The programmer should be aware that the non-detectability of elements, on this device, is produced by software. This means that if the console operator points at a detectable element which is very close to a non-detectable element, the hardware-software combination may decide that the console operator was pointing at the non-detectable element and discard the attention. For this reason, the programmer should keep detectable elements clear of all other elements.

The procedures which may be used with this device are:

CALL UGEINT(OPTIONS,ELEMENT);
CALL UGEFNT(OPTIONS,X,Y,ELEMENT);
CALL UGEFIN(OPTIONS,X,Y,EVENT,ELEMENT);
CALL UGETT(OPTIONS,X,Y,TEXT,ELEMENT);
CALL UGEPNTS(OPTIONS,YARAY,YARAY,PTS,ELEMENT);
CALL UGELES(OPTIONS,YARAY,YARAY,PTS,BBITS,ELEMENT);
When an element is transmitted to an IBM 2250, all intensity level, wink mode, and color information is ignored. All lines are of the SOLD type. Horizontal text with the VSHL or SHAL options will be produced in the small size and horizontal text with the LANG or VLRG option will be produced in the large size.

CALL UGAXIS(OPTIONS,LOCTIONS,COPTIONS,INDVAR,LOYVAR,HIVAR,
LOLAB,NILAB,ELAB,ELEMENT);
CALL UG3DMS(OPTIONS,LOCTIONS,ARRAY,TRANS,SHAPE,ELEMENT);

CALL UGOPEN(OPTIONS,IDENT);
The valid items in OPTIONS are:
IBM2250 Specifies the IBM 2250.
MAXIMUM=<value> The maximum number of elements which may be present in a single picture. The default value is 25.
MAXSIZE=<value> The maximum number of bytes that may be in an element after it has been translated to device
orders. The default value is 2000.

MAXDFIL=<value> The maximum number of bytes that may be
in a complete display file (in the IBM 2640) at one
time. The default value is 4096.

DDNAME=<value> The dname of the display unit. The
default value is SCOPE50.

CALL UGCLOS(IDENT);
CALL UGSLCT(IDENT);
CALL UGSCAL(OPTIONS,PROG,DEV,VLDB);

The default values for the programmer and device limits for
this device are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 4095 \\
0 & 4095 \\
\end{bmatrix}
\]

CALL UGINFO(STRING,IDEN,ARRAY);

The array ARRAY will contain the following:

ARRAY(1) The number of words used in the largest graphic
element that has been transmitted to the device.

ARRAY(2) The largest number of bytes in an element after
it has been translated to device orders.

ARRAY(3) The maximum number of elements which were
present in a single picture.

ARRAY(4) The maximum number of bytes that were in a
single display file.

ARRAY(5) When procedure UGPUT transmits an element to
the display device, this location is set to the length
of the element after it has been translated to device
orders. Other procedures do not modify this value.

CALL UGPUT(OPTIONS,IDEN,ELEMENT);

The items in OPTIONS which will be recognized and processed are
DET, OMIT, ON, and CPP.

CALL UGPRINT(OPTIONS,IDEN);

The items in OPTIONS which will be recognized and processed are
CLEAR, DET, NDDET, OMIT, WNER, ON and OMIT. Elements may be
fully manipulated.

CALL UGCRTY(OPTIONS,IODATA);

The only item in OPTIONS which will be recognized is BEEP.

CALL UGNUM(OPTIONS);

OPTIONS may contain CPP, KEDB, LPRM, or FRM. The number of
display regeneration cycles after a frame interrupt is fixed at
one.

CALL UGDATE(OPTIONS);

CALL UGNUM(TIME,ATTBD);

CALL UGPUT(OPTIONS,X,Y,IDEN,STRING);

Character size VSAL and NHAL result in small sized characters
while LARG and VIG give large sized characters. All other
items in OPTIONS are ignored. The input buffer should consist
of an even number of characters.

CALL UGGET(IDEN,STRING,ICUB);

CALL UGDXYO(IDATA,HIEATA,HILAB,HILAB,LOLAB,HILAB,LLAB);

CALL UGVENV(OPTIONS,BEPP,VDIM,NDIM,SCRD,SCRB,TRANS);

CALL UGORDER(OPTIONS,BEPP,VDIM,NDIM,SCRD,SCRB,TRANS);
CALL UGPROJ(TRANS,PT3D,PT2D);
CALL UGCSTOL(OPTIONS,TEXT,ARRAY,ARRAY,NPTS,BBITS);
CALL UGEDIV(ELEMENT1,ELEMENT2);

CALL UGPTOX(CPP,CPS,EC);
CALL UGTOP(CPP,CPS,EC);

CALL UGX004(LEVEL,NAME,INDEX);
CALL UGX002(OPTIONS,INDATA,EIDATA);

In addition to the errors described with each procedure, there are some other errors which may be detected. These errors are identified by a procedure name of UGIB5002 and their description is:

0(4): An asynchronous buffer error has occurred; the display unit is malfunctioning.

N(4): (Where N is greater than 255.) An extremely unusual error condition has been detected.

SECTION B.2: THE GIF-IDIIOM DISPLAY CONSOLE

The GIF-IDIIOM is an on-line interactive display console. The Graphic Interpretation Facility (GIF) is operated by the SLAC Computation Group as a research facility for on-line graphic systems. The facility consists of a Varian 620/I computer and an IDIIOM display console with a 21 inch CRT, an alphanumeric keyboard, a light pen, a function keyboard with 32 buttons, and a number of other special devices that have been developed.

The Varian 620/I computer is connected to the central computer system through an IBM 2701 Parallel Data Adaptor. A program running on the central computer system and using the GIF-IDIIOM will transmit display orders through the IBM 2701 to the Varian 620/I. The Varian 620/I assembles these orders into a valid display file and transmits this picture to the IDIIOM display console 60 times a second. The three sense switches on the Varian 620/I console control the mode in which the Varian 620/I is running. For normal operation these switches should be [Down-Down-Down]. To change these switches the console operator should do the following: (1) bring the program to a state where no I/O between the central computer and the Varian 620/I is in progress, (2) change the sense switch setting and, (3) push the bailt button on the IDIIOM console twice to signal the program of the change.

The GIF-IDIIOM can display points, lines, and characters at four intensity levels. Lines may be drawn with four structures: solid, dashed, dot-dashed, and dotted. The character generator can produce horizontal or vertical characters of four sizes. The following table gives the between character spacing and suggested between lines spacing assuming the default programmer coordinate system.
A DD card is necessary at execution time to define the display console. A typical DD card is:

```
//GIFLINK DD UNIT=GIF
```

The ddname, GIFLINK in the example, must correspond to the value given in the DDUHHE item in the options list of UGOPEN. The device address, GIF in the example, is the symbolic address of the port on the IBM 2701 connected to the Varian 620/I. Jobs using the GIF_IDIION run in the "G" job class.

The keyboard on the IDIION can be used to perform a number of special operations on a keyboard input buffer in addition to simply entering information. Many of these operations are performed by holding the CONTROL key down and depressing an alphabetic key. These operations are:

- **FS**: Advance the cursor to the next character without changing the current character.
- **(CONTROL)+A**: Same as FS.
- **BS**: Backspace the cursor without changing the current character.
- **(CONTROL)+B**: Same as BS.
- **(CONTROL)+W**: Move the cursor to the next keyboard input buffer.
- **(CONTROL)+N**: Move the cursor to the beginning of the current keyboard input buffer.
- **(CONTROL)+C**: Move the cursor to the beginning of the current keyboard input buffer and reset the entire keyboard input buffer to blanks.
- **(CONTROL)+D**: Delete the character at the current cursor position and move the following characters one space to the left.
- **(CONTROL)+I**: Move the character over the cursor and the following characters one space to the right and insert a blank at the current cursor position.
- **(CONTROL)+S**: Set a tab at the current cursor position.
- **(CONTROL)+T**: Tab the cursor to the next position with a set tab.
- **(CONTROL)+U**: Clear the current position of a set tab.
- **(CONTROL)+V**: Clear all set tabs.

The keyboard attention is generated by the ESCAPE key.

The light pen attention is generated by pointing at a light pen detectable element and pressing the button on the side of the pen. On the IDIION, a light pen detectable element will brighten on the screen when the light pen is pointing at it.

A tracking pattern (a small cross, about one half inch tall and wide) may be put on the screen. When the light pen is pointing at this tracking pattern, the pattern will center itself in the field of view of the light pen. When the console operator moves the light pen around the screen, the tracking pattern will follow...
the light pen. The programmer may determine the position of the tracking pattern at any time. In addition, the light pen and tracking pattern may be used to draw free-hand curves on the screen. This is done by adding a light pen input buffer to the display file and putting the IDIION into the drawing mode. In this mode, the console operator may draw curves on the screen by holding the button on the light pen down and moving the tracking pattern about. When the IDIION is put into the erase mode and the light pen drawing buffer is made light pen detectable, then the console operator may delete segments of these curves by holding the button on the light pen down and pointing to the segments to be deleted. Finally, a light pen input buffer may be read from the display.

In the IDIION, a display element may act as a subroutine. This type of element does not appear on the screen unless it is invoked by another element. In this way, a sub-picture may appear on the screen many times while the orders to draw that sub-picture are in the display file only once. Invocations of element-subroutines may be nested to any depth.

One of the special devices connected to the GIF-IDIION is a 16 millimeter, Model 16S ARRIFLEX motion picture camera with a standard motor and an animation motor. An interface between the movie camera and the Varian 620/I has been designed and built at SLAC. Through this interface, the Varian 620/I can control the animation motor and sense when the shutter is open. The movie camera may be operated in any of three modes. These modes, and the corresponding positions of the sense switches on the Varian 620/I are:

1. Over the Shoulder Mode [Down-Down-Up]: In this mode, the movie camera uses the standard motor and runs at its normal speed of 24 frames per second. The Varian 620/I continually senses the shutter open signal from the camera. When the shutter initially opens, the Varian 620/I will start the display. After going through the display file once, the display is turned off until the next time the shutter opens. This mode may, therefore, be used to take flicker-free movies of a person operating the console. The operation of any existing interactive program may be filmed.

2. Single Pulse Animation Mode [Down-Up-Down]: In this mode, the Varian 620/I sends signals to the camera telling it when to open its shutter. After the shutter has opened, the display will be started. The length of time that the shutter remains open is controlled by a switch on the animation motor. This switch may be set to give exposure times of 1/8, 1/4, and 1/2 seconds. A single pulse animation program is written in the following manner:

A. The FRAW attention should be enabled and the regeneration count set to a small value (usually 2, 3, or 4) with the FRAW option of procedure UGCTRL.

B. The elements for a single frame should be transmitted to the GIF-IDIION and the display should be started.
C. The occurrence of the FRAM attention indicates that one frame of movie film has been exposed. Step B may then be repeated. This type of animation program may be checked out as a normal interactive display program. The only difference is that the final movie will run at 24 frames per second, while many factors can affect the rate at which the frames appear on the CRT.

3. Double Pulse Animation Mode [Down-Up-Up]: This mode is a generalization of the previous mode. It covers the case where a single frame is composed of many display files. Thus, a single frame of movie film may contain a more complex picture than can be viewed in the normal interactive mode. A double pulse animation program is written in the following manner:

A. The FRAM attention should be enabled and the regeneration count should be set.
B. The shutter should be opened with the SHOPEN option in procedure UGCTRL.
C. The elements for a single display file should be transmitted to the GIF-IDITION and the display should be started.
D. The occurrence of the FRAM attention indicates that the display file has been exposed. Step C may then be repeated.
E. When all display files have been exposed, the shutter should be closed with the SECLOS option in procedure UGCTRL. Step B may then be repeated.

This type of animation may also be run and checked out in the normal interactive mode because the shutter open and close commands are ignored if the sense switches are not in the double pulse animation mode. However, the console operator may have to use considerable imagination to visualize the final movie.

Another of the special devices which is connected to the GIF-IDITION is a 3-D viewer. This viewer is a device which contains a motor driven rotating disk. The disk contains clear and opaque areas. When the console operator looks through this device, the left and right eyes will alternately be blocked by the opaque areas on the disk. An interface has been built at SLAC which allows the Varian 620/I to sense when the left or right eye has a clear view. By flashing different images on the screen when each eye is open, it is possible to present the user with a stereoscopic picture. A program which uses the 3-D viewer should obey the following conventions: (1) elements which are to be presented to the left eye only should have identifications which are less than -1000, (2) elements which are to be presented to the right eye only should have identifications which are greater than +1000, (3) elements which are to be presented to both eyes should have identifications between -1000 and +1000 (inclusive). In addition, the three sense switches on the Varian 620/I should be in the [Up-Down-Down] position. This setting of the sense switches signals the program that it is to synchronize the display with the 3-D viewer.
On-line animation may be done by simply re-generating and re-transmitting a changing graphic element at regular intervals. Normally when an element is being changed, the VAXim 620/1 will turn off the CRT while the display file is being changed. However this can cause an annoying amount of flicker when on-line animation is being attempted. A solution is to use a special setting of the sense switches, [Up-Down-Up], and assure, by using the MINRESIZE option in procedure UGEPUT, that the element being changed is of constant size. Under these conditions, the display is turned off for a considerably shorter time and the element being changed can simply be read into the same space occupied by its earlier version.

The procedures which may be used with this device are:

CALL UGEINIT(OPTIONS, ELEMENT);
CALL UGEDET(OPTIONS, X, Y, ELEMENT);
CALL UGELIN(OPTIONS, X, Y, BIT, ELEMENT);
CALL UGETXT(OPTIONS, X, Y, TEXT, ELEMENT);
CALL UGEPST(OPTIONS, XARRAY, YARRAY, NP3, ELEMENT);
CALL UGELMS(OPTIONS, XARRAY, YARRAY, NP3, BBITS, ELEMENT);

The IDIIGN will correctly process the intensity levels VDIM, DMIN, BNT, VBT; the wink mode specifications WINK and STDY; and the line structure specifications SOLD, DASH, DDSN, and DOTS. Color information is ignored. Character sizes of VSHL, SHAL, LANG, and VLRG may be used. The intensity level and line structure information is not very useful because of minor design flaws in the early model IDIIGN that we have at SLAC.

CALL UGAXIS(OPTIONS, LCTFORM, COPTIONS, INDVAR, LOVAR, HVAR,
LOLAB, NCLAB, NLAB, ELEMENT);
CALL UG3DMS(OPTIONS, LCTFORM, ARRAY, TRANS, SHAAREA, ELEMENT);
CALL UGEXDV(OPTIONS, X, Y, IDIERT, ELEMENT);

The ANGLE and SCALE data will be ignored and assumed to be 0.0 and 1.0 respectively.

CALL UGEDITA(OPTIONS, STRING, ELEMENT);

CALL UGOOPEN(OPTIONS, IDENT);

The valid items in OPTIONS are:

GIF/IDI Specifies the GIF-IDIIGN.
MAXENVE=<value> The maximum number of elements which may be present in a single picture. The default value is 25.
MAXESIZ=<value> The maximum number of bytes that may be in an element after it has been translated to device orders. The default value is 2000.
DDNAME=<value> The dname of the display unit. The default value is GIFLINK.

CALL UGCLOS(IDENT);
CALL UGSLCT(IDENT);
CALL UGSCAL(OPTIONS, PBCLI, DEVCLEI);

The default values for the programmer and device limits for this device are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1023 \\
0 & 1023 \\
\end{bmatrix}
\]
CALL UGINFO(STRING,IDENT,ARRAY);

The array ARRAY will contain the following:

ARRAY(1) The number of words used in the largest graphic element that has been transmitted to the device.
ARRAY(2) The largest number of bytes in an element after it has been translated to device orders.
ARRAY(3),ARRAY(4) ZERO.
ARRAY(5) When procedure UGEPUT transmits an element to the display device, this location is set to the length of the element after it has been translated to device orders. Other procedures do not modify this value.

CALL UGEPUT(OPTIONS,IDENT,ELEMENT);

The items in OPTIONS which will be recognized and processed are DBTC, ONIT, ON, OFF, HCHP, DTTC, ESUB, and HINESIE. HCHP signifies that the element is to be translated to non-compact orders. This option makes it possible to determine the exact point or line that was pointed to in a light penned element.

CALL UGPICT(OPTIONS,IDENT);

The items in OPTIONS which will be recognized and processed are CLEAR, DTTC, NDIT, ONIT, INCL, ON and OFF. Elements may be fully manipulated.

CALL UGCTRL(OPTIONS,ICETATA);

The valid items in OPTIONS are:

FRAN The regeneration count for the FRAN attention may be any positive value.
LITB<=<value> If the given bit string does not contain 32 bits, it is padded on the right with zeros. The function keyboard is arranged in 4 rows of 8 buttons. Button number one is in the top row on the left, button number two is in the top row next to button number one, etc.
TPPUT,TPGET,TPHNV The tracking pattern may be put on the screen, its position may be obtained, and it may be removed from the screen.
TPDCC,TPDNV,TPSBS The IDIION may be put into the inactive, draw, or erase mode with respect to light pen input buffers.
BSON,BSOFF,BSTRC The IDIION may be instructed to check for or ignore element-subroutines, and the programmer may request an element-subroutine trace.
SHOPEN,SHCLOS The shutter on the synchronized movie camera may be opened and closed in the double pulse animation mode.

CALL UGRATH(OPTIONS);

OPTIONS may contain OFF, KERD, LPEN, FRAN, PPEN, PPBR, PPLH, PPBR, LPEN, or LPNB.

CALL UGDATH(OPTIONS);

CALL UGWRITE(TIME,ATTNE);

CALL UGEPUT(OPTIONS,X,Y,IDENT,STRING);

The character sizes VSHE, SHEI, LARG, and VLRE; and the intensity levels DEXH, DIXH, BBHT, and VBBT will be correctly processed. The input buffer should consist of an even number of characters.

CALL UGWRITE(IDENT,STRING,ICUR);
CALL UG12PUT(OPTIONS, IDENT, LENGTH);
The intensity levels VDIM, DIMH, BRIT, and VBBT; and the line structure specifications SOLB, DASH, DDSH, and DOTS will be correctly processed.
CALL UGLCRT(IDENT, XARRAY, YARRAY, NPTS, BEITS);

CALL UGDXDY(LODATA, HIDATA, HILAB, NILAB, LOLAB, HILAB, NLAB);
CALL UGVIEW(OPTIONS, BEIT, VDIR, RDIR, SCRD, SCRZ, TRANS);
CALL UGPROJ(TRANS, PT3D, PT2D);
CALL UGCTOL(OPTIONS, TEXT, XARRAY, YARRAY, NPTS, BEITS);
CALL UGEDIV(ELHENT1, ELHENT2);

CALL UGPTOX(CFP, CPS, EC);
CALL UGXTOP(CFP, CPS, EC);

CALL UGX001(LEVEL, NAME, INDEX);
CALL UGX002(OPTIONS, INDATA, HIDATA);

In addition to the errors described with each procedure, there are some other errors which may be detected. These errors are identified by a procedure name of UGGFPI02 and their description is:

1(4): The light pen input buffer in the Varian 620/I is full.
N(4): (where N is greater than 255.) An extremely unusual error condition has been detected.

SECTION B.3: THE TECTRONIX 4013 DISPLAY TERMINAL

The TECTRONIX 4013 is an interactive storage display terminal with an eight inch wide by six inch high screen and a keyboard. The CRT on the TECTRONIX 4013 differs from the CRT's on the previous devices in that partial erasure of a picture is impossible; once a display item is drawn on the screen it remains there until the entire screen is cleared.

The TECTRONIX 4013 terminals are under the control of the MILTEM Terminal Manager. A program which sends information to and receives information from a terminal must communicate with MILTEM. To run an interactive program using the TECTRONIX 4013 the user should:

1. Sign on the terminal and begin using the WYLBUR Text Editing System in the usual manner.
2. Submit a job which runs in one of the interactive partitions and wait until it identifies itself as a sub-system to MILTEM. The command SHOW SYSTEMS will list the currently running systems. The name of your sub-system will be the same as the job name.
3. When your sub-system is known to MILTEM, you may type its name to connect your terminal to your sub-system. However before this connection is made, MILTEM will ask for the sub-system's Magic Word (see options under
UGOPEN). Once you have correctly entered the Magic Word, no other terminal will be allowed to use the sub-system.

4. When running in the interactive mode, the keyboard attention is generated by the RETURN key. If you have been typing some information and wish to cancel what you have done and start all over, you may hit the BREAK key. In addition, you may temporarily return to UYLEUR by typing $YUL and hitting the RETURN key. When you are finished using UYLEUR you may type the name of your sub-system to resume.

The Tektronix 4013 can display points, lines, and characters at a single intensity level. The character generator can produce horizontal characters of a single size. The following table gives the between character spacing and suggested between lines spacing assuming the default programmer coordinate system.

<table>
<thead>
<tr>
<th>Character Size</th>
<th>Character Spacing</th>
<th>Line Spacing</th>
<th>Characters per Line</th>
<th>Lines per Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAL</td>
<td>0.01797</td>
<td>0.02824</td>
<td>56</td>
<td>35</td>
</tr>
</tbody>
</table>

When the PULSCH option is used in procedure UGOPEN, then 74 characters will fit on a single line.

A DD card is not needed unless you are using the IDSCOR option in UGOPEN. In this case your JCL should include:

```ruby
//ANYNAME DD DUMMY
```

The ddname, ANYNAME in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEN.

Only one keyboard input buffer can be on the screen at a time. If you attempt to put more than one on the screen, only the last one will be active.

These display units have a limited tracking facility. A set of vertical and horizontal cross-hairs may be put on the screen and moved by the console operator with the thumb wheels on the right hand side of the keyboard. When the cross-hairs are on the screen the console operator may strike a printing key, not the carriage return, to cause the coordinates to be sent to the central computer and the cross-hairs removed from the screen. Since the position is sent when the console operator strikes a key, it means that the keyboard cannot be used to enter text while the cross-hairs are present.

The procedures which may be used with this device are:

```ruby
CALL UGEINT(OPTIONS, ELEMENT);
CALL UGEPIR(OPTIONS, X, Y, ELEMENT);
CALL UGELIN(OPTIONS, X, Y, BEIT, ELEMENT);
CALL UGEFRT(OPTIONS, X, Y, TEXT, ELEMENT);
CALL UGEPTS(OPTIONS, XBAR, YBAR, NPTS, ELEMENT);
CALL UGEPLS(OPTIONS, XBAR, YBAR, NPTS, BEITS, ELEMENT);
```

When an element is transmitted to this device, all intensity
level, wink mode, and color information (with one exception) is ignored. The exception is that points with the DINH or VDIN option are plotted as a single raster point while all other points are plotted as four raster points. All lines are of the SOLD type. Horizontal text with the VSHE, SHAL, LANG, or VLRG options will be produced by the character generator.

CALL UGANIS(OPTIONS, LOCATIONS, OPTIONS, IDVAR, LOVAR, MIVAR, LOLAB, XLAB, YLAB, ELEMENT);
CALL UG3DMS(OPTIONS, LOCATIONS, ARRAY, TRANS, WAREA, ELEMENT);

CALL UGOPEN(OPTIONS, IDENT);
The valid items in OPTIONS are:

TEK4013 Specifies the TEKTRONIX 4013.

MAGICWD=<value> The Magic Word that HILTEM will request.
   The Magic Word may have from one to eight characters
   and has a default value of XXXXXX.

DDNAME=<value> The ddname of the display unit when
   IDSCOPE is used. No default value is supplied.

FULSCR A flag which specifies that the picture is to be
   scissored at the screen boundaries and not at a square
   area within the screen.
When UGOPEN is called, your program will identify itself to
   HILTEM as a sub-system and wait until you have made the
   connection between your sub-system and your terminal.

CALL UGCLOS(IDENT);
CALL UGSLCLT(IDENT);
CALL UGSCAL(OPTIONS, DECGLN, DEVGLN);
The default values for the programmer and device limits are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
122 & 901 \\
0 & 779 \\
\end{bmatrix}
\]

The default values when FULSCR is used are:

\[
\begin{bmatrix}
-0.15661 & 1.15661 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 1023 \\
0 & 779 \\
\end{bmatrix}
\]

CALL UGINF0(STRING, IDENT, ARRAY);
The array ARRAY will contain the following:
   ARRAY(1) The number of words used in the largest graphic
   element that has been transmitted to the device.
   ARRAY(2) ...ARRAY(5) Zero.

CALL UGEPUT(OPTIONS, IDENT, ELEMENT);
For this device, the contents of both OPTIONS and IDENT are
   ignored. Hitting the BREAK key while a picture is being drawn
   will terminate that call to UGEPUT with no other effect on the
   program.

CALL UGPICT(OPTIONS, IDENT);
The only valid item in OPTIONS is CLEAR. The contents of IDENT
   is ignored and assumed to be zero.

CALL UGCTRL(OPTIONS, IDDATA);
The only valid items in OPTIONS are BEEP and TRCCTL. TRCCTL
   causes the cross-hairs to be put on the screen and the console
   to be put into the wait state. When the console operator
   strikes a printing key, the cross-hair coordinates are read and
   UGCTRL returns to the calling procedure.
CALL UGREATH(OPTIONS);
The only item in OPTIONS which will be recognized is KBRD.
CALL UGDATE(OPTIONS);
CALL UGREATH(TIME,ATTED);

CALL UGKPUT(OPTIONS,I1,Y,IDENT,STRING);
OPTIONS may contain LCASE; all character size information is
ignored. The contents of STRING will not be displayed because
the typed characters would not then be readable on the storage
scope.
CALL UGKGET(IDENT,STRING,ICUB);
The characters that are read are exactly those characters that
the console operator has typed.

CALL UGDSDKY(LODATA,HIDATA,HILAB,HILAB,LOLAB,HILAB,HILAB);
CALL UGVIREU(OPTIONS,REFP,VDIR,EDIR,SCRD,SCRE,TRANS);
CALL UGORTN(OPTIONS,REFP,VDIR,EDIR,SCRD,SCRE,TRANS);
CALL UGPLOJ(TRANS,PT3D,PT2D);
CALL UGCTRUL(OPTIONS,TEXT,ARRAY,ARRAY,NPTS,BBITS);
CALL UGEDIV(EBLEN1,EBLEN2);

CALL UGPTOX(CPP,CPS,EC);
CALL UGXTOP(CPP,CPS,EC);

CALL UIG001(LEVEL,NAME,INDEX);
CALL UIG002(OPTIONS,INDATA,HIDATA);

In addition to the errors described with each procedure, there
are some other errors which may be detected. These errors are
identified by a procedure name of UGTK1302 and their description is:

N(N): (where N is greater than 255.) An extremely unusual
error condition has been detected.
The Character Stroke Generator Sub-System is that part of this system which converts text data into a graphic element into short line segments forming the characters. The stroke generator will be used in the following cases:

1. The programmer has specified that it must be used by giving the SPACING option when the data was added to the graphic element.
2. The programmer has specified that either the stroke generator or the device character generator may be used by giving the ISPACING option, but either the device has no character generator or it has one which cannot satisfy all of the requirements.

The stroke generator is a program module which contains tables to describe how each character is formed from line segments. The standard stroke generator is able to process all of the characters on the IBM 029 keypunch in addition to the ten plotting symbols. Also the PECDIC codes for lower case letters are produced as upper case. The characters produced by the standard stroke generator are shown in lines one, five, and six of Figure C.1.1.

SECTION C.1: MODULE UGX103A

The characters produced by the standard stroke generator should be sufficient for most applications. However, some applications can require additional characters such as Greek letters, superscripts, or subscripts. For this reason an alternate program module is available. If the programmer creates the load module with the LINK-EDITOR and includes the card:

INCLUDE SYSLIB(UGX103A)

in the SYSIN data set, then this alternate module will be used. This alternate module is larger than the standard module; it has the following additional capabilities:

1. Upper and lower case Roman and Greek letters may be produced.
2. Many additional special characters are available.
3. Characters are available which cause the stroke generator to enter superscript or subscript mode. Superscripts or subscripts may themselves have superscripts or subscripts.

This alternate module differs from the standard stroke generator in one other important respect. The standard module generates characters which all have the same width (their width is two-thirds of the value given by the SPACING or XPARCS option in UGETM and their height is the same as that value). This alternate module generates characters of differing widths and heights, thus the upper case letter "N" is twice as wide as an
"I", and most lower case letters are three-fourths as wide as most upper case letters. This results in text with a more pleasing appearance, but also causes a problem: if, for instance, a letter is to carry both a superscript and subscript, something equivalent to a backspace would be necessary, but the amount backspaced would depend on the characters in the superscript (or subscript). To overcome this problem, a character has been introduced which causes the stroke generator to save its current position and state. Another special character in a later part of the string can cause the earlier state of the character generator to be restored. There are three independent save-restore character pairs available.

The following table gives the hexadecimal equivalent of each character. The meaning and use of the character pairs will be described in a following section on procedures UGPTOX and UGTOP.

<table>
<thead>
<tr>
<th>Hex. Char Code Pair</th>
<th>Character Description</th>
<th>Hex. Char Code Pair</th>
<th>Character Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>mS</td>
<td>08</td>
<td>QG</td>
</tr>
<tr>
<td>01</td>
<td>AG</td>
<td>09</td>
<td>IG</td>
</tr>
<tr>
<td>02</td>
<td>BG</td>
<td>0A</td>
<td>IS</td>
</tr>
<tr>
<td>03</td>
<td>GG</td>
<td>0B</td>
<td>XS</td>
</tr>
<tr>
<td>04</td>
<td>DG</td>
<td>0C</td>
<td>:S</td>
</tr>
<tr>
<td>05</td>
<td>EG</td>
<td>0D</td>
<td>ES</td>
</tr>
<tr>
<td>06</td>
<td>EG</td>
<td>0F</td>
<td>PS</td>
</tr>
<tr>
<td>07</td>
<td>EG</td>
<td>0F</td>
<td>DS</td>
</tr>
<tr>
<td>10</td>
<td>?S</td>
<td>18</td>
<td>RG</td>
</tr>
<tr>
<td>11</td>
<td>KG</td>
<td>19</td>
<td>SG</td>
</tr>
<tr>
<td>12</td>
<td>LG</td>
<td>1A</td>
<td>UA</td>
</tr>
<tr>
<td>13</td>
<td>MG</td>
<td>1B</td>
<td>DA</td>
</tr>
<tr>
<td>14</td>
<td>MG</td>
<td>1C</td>
<td>LA</td>
</tr>
<tr>
<td>15</td>
<td>XG</td>
<td>1D</td>
<td>BA</td>
</tr>
<tr>
<td>16</td>
<td>OG</td>
<td>1E</td>
<td>EA</td>
</tr>
<tr>
<td>17</td>
<td>PG</td>
<td>1F</td>
<td>Unassigned</td>
</tr>
<tr>
<td>20</td>
<td>NX</td>
<td>28</td>
<td>YG</td>
</tr>
<tr>
<td>21</td>
<td>NX</td>
<td>29</td>
<td>WG</td>
</tr>
<tr>
<td>22</td>
<td>NX</td>
<td>2A</td>
<td>IX</td>
</tr>
<tr>
<td>23</td>
<td>AX</td>
<td>2B</td>
<td>UX</td>
</tr>
<tr>
<td>24</td>
<td>TG</td>
<td>2C</td>
<td>&lt;X</td>
</tr>
<tr>
<td>25</td>
<td>TG</td>
<td>2D</td>
<td>&gt;X</td>
</tr>
<tr>
<td>26</td>
<td>TG</td>
<td>2E</td>
<td>LX</td>
</tr>
<tr>
<td>27</td>
<td>CG</td>
<td>2F</td>
<td>FY</td>
</tr>
<tr>
<td>30</td>
<td>0C</td>
<td>38</td>
<td>8C</td>
</tr>
<tr>
<td>31</td>
<td>1C</td>
<td>39</td>
<td>9C</td>
</tr>
<tr>
<td>32</td>
<td>2C</td>
<td>3A</td>
<td>Unassigned</td>
</tr>
<tr>
<td>33</td>
<td>3C</td>
<td>3B</td>
<td>Unassigned</td>
</tr>
<tr>
<td>34</td>
<td>4C</td>
<td>3C</td>
<td>Unassigned</td>
</tr>
<tr>
<td>35</td>
<td>5C</td>
<td>3D</td>
<td>Unassigned</td>
</tr>
<tr>
<td>36</td>
<td>6C</td>
<td>3E</td>
<td>EC</td>
</tr>
<tr>
<td>37</td>
<td>7C</td>
<td>3F</td>
<td>FC</td>
</tr>
</tbody>
</table>

Note: The symbols listed in the table are hexadecimal values and represent the characters to be used. The meanings of the symbols are as follows:

- HS: Null
- AG: L.C. Alpha
- BG: L.C. Beta
- GG: L.C. Gamma
- DG: L.C. Delta
- EG: L.C. Epsilon
- KG: L.C. Zeta
- LG: L.C. Eta
- MG: L.C. Nu
- DG: L.C. Nu
- XG: L.C. Xi
- OG: L.C. Omicron
- PG: L.C. Pi
- NX: Membership Symbol
- WX: Membership Negation
- XX: Existential Quant.
- AX: Universal Quant.
- TG: L.C. Tau
- UG: L.C. Upsilon
- FG: L.C. Phi
- CG: L.C. Chi
- OC: Enter Superscr Mode
- 1C: Leave Superscr Mode
- 2C: Enter Subscr Mode
- 3C: Leave Subscr Mode
- 4C: Save Position-1
- 5C: Restore Pos-1
- 6C: Save Position-2
- 7C: Restore Pos-2
- 8C: Save Position-3
- 9C: Restore Pos-3
- EC: Increase Size
- FC: Decrease Size
<table>
<thead>
<tr>
<th>Hex. Char Code Pair</th>
<th>Character Description</th>
<th>Hex. Char Code Pair</th>
<th>Character Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Blank</td>
<td>48</td>
<td>U.C. Theta</td>
</tr>
<tr>
<td>41</td>
<td>U.C. Alpha</td>
<td>49</td>
<td>U.C. Iota</td>
</tr>
<tr>
<td>42</td>
<td>U.C. Beta</td>
<td>4A</td>
<td>Cent Sign</td>
</tr>
<tr>
<td>43</td>
<td>U.C. Gamma</td>
<td>4B</td>
<td>Period</td>
</tr>
<tr>
<td>44</td>
<td>U.C. Delta</td>
<td>4C</td>
<td>Less Than</td>
</tr>
<tr>
<td>45</td>
<td>U.C. Epsilon</td>
<td>4D</td>
<td>Left Parenthesis</td>
</tr>
<tr>
<td>46</td>
<td>U.C. Zeta</td>
<td>4E</td>
<td>Plus Sign</td>
</tr>
<tr>
<td>47</td>
<td>U.C. Eta</td>
<td>4F</td>
<td>Vertical Line</td>
</tr>
<tr>
<td>50</td>
<td>&amp;</td>
<td>58</td>
<td>U.C. Rho</td>
</tr>
<tr>
<td>51</td>
<td>U.C. Kappa</td>
<td>59</td>
<td>U.C. Sigma</td>
</tr>
<tr>
<td>52</td>
<td>U.C. Lambda</td>
<td>5A</td>
<td>Exclamation Mark</td>
</tr>
<tr>
<td>53</td>
<td>U.C. Nu</td>
<td>5B</td>
<td>Dollar Sign</td>
</tr>
<tr>
<td>54</td>
<td>U.C. Xi</td>
<td>5C</td>
<td>Asterisk</td>
</tr>
<tr>
<td>55</td>
<td>U.C. Xi</td>
<td>5D</td>
<td>Right Parenthesis</td>
</tr>
<tr>
<td>56</td>
<td>U.C. Omicron</td>
<td>5E</td>
<td>Semi-Colon</td>
</tr>
<tr>
<td>57</td>
<td>U.C. Pi</td>
<td>5F</td>
<td>Not</td>
</tr>
<tr>
<td>60</td>
<td>-</td>
<td>6A</td>
<td>Dagger</td>
</tr>
<tr>
<td>61</td>
<td>/</td>
<td>6B</td>
<td>Comma</td>
</tr>
<tr>
<td>62</td>
<td>+</td>
<td>6C</td>
<td>Percent</td>
</tr>
<tr>
<td>63</td>
<td>-</td>
<td>6D</td>
<td>Underline</td>
</tr>
<tr>
<td>64</td>
<td>*</td>
<td>6E</td>
<td>Greater Than</td>
</tr>
<tr>
<td>65</td>
<td>*</td>
<td>6F</td>
<td>Question Mark</td>
</tr>
<tr>
<td>66</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Unassigned</td>
<td>78</td>
<td>Left Angle Bracket</td>
</tr>
<tr>
<td>71</td>
<td>Unassigned</td>
<td>79</td>
<td>Right Angle Bracket</td>
</tr>
<tr>
<td>72</td>
<td>Unassigned</td>
<td>7A</td>
<td>Colon</td>
</tr>
<tr>
<td>73</td>
<td>Unassigned</td>
<td>7B</td>
<td>Pound Sign</td>
</tr>
<tr>
<td>74</td>
<td>Unassigned</td>
<td>7C</td>
<td>At Sign</td>
</tr>
<tr>
<td>75</td>
<td>Unassigned</td>
<td>7D</td>
<td>Apostrophe</td>
</tr>
<tr>
<td>76</td>
<td>Unassigned</td>
<td>7E</td>
<td>Equals Sign</td>
</tr>
<tr>
<td>77</td>
<td>Unassigned</td>
<td>7F</td>
<td>Quote Marks</td>
</tr>
<tr>
<td>80</td>
<td>Unassigned</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>AL</td>
<td>81</td>
<td>L.C. A</td>
</tr>
<tr>
<td>82</td>
<td>BL</td>
<td>82</td>
<td>L.C. B</td>
</tr>
<tr>
<td>83</td>
<td>CL</td>
<td>83</td>
<td>L.C. C</td>
</tr>
<tr>
<td>84</td>
<td>DL</td>
<td>84</td>
<td>L.C. D</td>
</tr>
<tr>
<td>85</td>
<td>EL</td>
<td>85</td>
<td>L.C. E</td>
</tr>
<tr>
<td>86</td>
<td>FL</td>
<td>86</td>
<td>L.C. F</td>
</tr>
<tr>
<td>87</td>
<td>GL</td>
<td>87</td>
<td>L.C. G</td>
</tr>
<tr>
<td>90</td>
<td>Unassigned</td>
<td>90</td>
<td>L.C. Q</td>
</tr>
<tr>
<td>91</td>
<td>JL</td>
<td>91</td>
<td>L.C. R</td>
</tr>
<tr>
<td>92</td>
<td>KL</td>
<td>92</td>
<td>Unassigned</td>
</tr>
<tr>
<td>93</td>
<td>LL</td>
<td>93</td>
<td>Right Brace</td>
</tr>
<tr>
<td>94</td>
<td>ML</td>
<td>94</td>
<td>Unassigned</td>
</tr>
<tr>
<td>95</td>
<td>NL</td>
<td>95</td>
<td>Unassigned</td>
</tr>
<tr>
<td>96</td>
<td>OL</td>
<td>96</td>
<td>Plus or Minus</td>
</tr>
<tr>
<td>97</td>
<td>PL</td>
<td>97</td>
<td>Unassigned</td>
</tr>
<tr>
<td>Hex. Code</td>
<td>Char Pair</td>
<td>Character Description</td>
<td>Hex. Code</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>A0</td>
<td>0S</td>
<td>Unassigned</td>
<td>A8</td>
</tr>
<tr>
<td>A1</td>
<td>0S</td>
<td>Degrees</td>
<td>A9</td>
</tr>
<tr>
<td>A2</td>
<td>SL</td>
<td>L.C. S</td>
<td>AA</td>
</tr>
<tr>
<td>A3</td>
<td>TL</td>
<td>L.C. T</td>
<td>AB</td>
</tr>
<tr>
<td>A4</td>
<td>UL</td>
<td>L.C. U</td>
<td>AC</td>
</tr>
<tr>
<td>A5</td>
<td>VL</td>
<td>L.C. V</td>
<td>AD</td>
</tr>
<tr>
<td>A6</td>
<td>UL</td>
<td>L.C. W</td>
<td>AE</td>
</tr>
<tr>
<td>A7</td>
<td>VL</td>
<td>L.C. X</td>
<td>AF</td>
</tr>
<tr>
<td>B0</td>
<td>0F</td>
<td>Plot Syn V Cross</td>
<td>B8</td>
</tr>
<tr>
<td>B1</td>
<td>1P</td>
<td>Plot Syn D Cross</td>
<td>B9</td>
</tr>
<tr>
<td>B2</td>
<td>2P</td>
<td>Plot Syn Diamond</td>
<td>BA</td>
</tr>
<tr>
<td>B3</td>
<td>3P</td>
<td>Plot Syn Square</td>
<td>BB</td>
</tr>
<tr>
<td>B4</td>
<td>4P</td>
<td>Plot Syn F Diamond</td>
<td>BC</td>
</tr>
<tr>
<td>B5</td>
<td>5P</td>
<td>Plot Syn F Square</td>
<td>BD</td>
</tr>
<tr>
<td>B6</td>
<td>6P</td>
<td>Plot Syn F V Cross</td>
<td>BE</td>
</tr>
<tr>
<td>B7</td>
<td>7P</td>
<td>Plot Syn F D Cross</td>
<td>BF</td>
</tr>
<tr>
<td>C0</td>
<td></td>
<td>Unassigned</td>
<td>C8</td>
</tr>
<tr>
<td>C1</td>
<td>A</td>
<td>U.C. I</td>
<td>C9</td>
</tr>
<tr>
<td>C2</td>
<td>B</td>
<td>U.C. B</td>
<td>CA</td>
</tr>
<tr>
<td>C3</td>
<td>C</td>
<td>U.C. C</td>
<td>CB</td>
</tr>
<tr>
<td>C4</td>
<td>D</td>
<td>U.C. D</td>
<td>CC</td>
</tr>
<tr>
<td>C5</td>
<td>E</td>
<td>U.C. E</td>
<td>CD</td>
</tr>
<tr>
<td>C6</td>
<td>F</td>
<td>U.C. F</td>
<td>CE</td>
</tr>
<tr>
<td>C7</td>
<td>G</td>
<td>U.C. G</td>
<td>CF</td>
</tr>
<tr>
<td>D0</td>
<td>0U</td>
<td>Backwards Blank</td>
<td>D8</td>
</tr>
<tr>
<td>D1</td>
<td>J</td>
<td>U.C. J</td>
<td>D9</td>
</tr>
<tr>
<td>D2</td>
<td>K</td>
<td>U.C. K</td>
<td>DA</td>
</tr>
<tr>
<td>D3</td>
<td>L</td>
<td>U.C. L</td>
<td>DB</td>
</tr>
<tr>
<td>D4</td>
<td>M</td>
<td>U.C. M</td>
<td>DC</td>
</tr>
<tr>
<td>D5</td>
<td>N</td>
<td>U.C. N</td>
<td>DD</td>
</tr>
<tr>
<td>D6</td>
<td>O</td>
<td>U.C. O</td>
<td>DE</td>
</tr>
<tr>
<td>D7</td>
<td>P</td>
<td>U.C. P</td>
<td>DF</td>
</tr>
<tr>
<td>E0</td>
<td>/S</td>
<td>Backwards Slash</td>
<td>E8</td>
</tr>
<tr>
<td>E1</td>
<td></td>
<td>Unassigned</td>
<td>E9</td>
</tr>
<tr>
<td>E2</td>
<td>S</td>
<td>U.C. S</td>
<td>EA</td>
</tr>
<tr>
<td>E3</td>
<td>T</td>
<td>U.C. T</td>
<td>EB</td>
</tr>
<tr>
<td>E4</td>
<td>U</td>
<td>U.C. U</td>
<td>EC</td>
</tr>
<tr>
<td>E5</td>
<td>V</td>
<td>U.C. V</td>
<td>ED</td>
</tr>
<tr>
<td>E6</td>
<td>W</td>
<td>U.C. W</td>
<td>EE</td>
</tr>
<tr>
<td>E7</td>
<td>X</td>
<td>U.C. X</td>
<td>EF</td>
</tr>
<tr>
<td>F0</td>
<td>0U</td>
<td>Numeral 0</td>
<td>F8</td>
</tr>
<tr>
<td>F1</td>
<td>1U</td>
<td>Numeral 1</td>
<td>F9</td>
</tr>
<tr>
<td>F2</td>
<td>2U</td>
<td>Numeral 2</td>
<td>FA</td>
</tr>
<tr>
<td>F3</td>
<td>3U</td>
<td>Numeral 3</td>
<td>FB</td>
</tr>
<tr>
<td>F4</td>
<td>4U</td>
<td>Numeral 4</td>
<td>FC</td>
</tr>
<tr>
<td>F5</td>
<td>5U</td>
<td>Numeral 5</td>
<td>FD</td>
</tr>
<tr>
<td>F6</td>
<td>6U</td>
<td>Numeral 6</td>
<td>FE</td>
</tr>
<tr>
<td>F7</td>
<td>7U</td>
<td>Numeral 7</td>
<td>FF</td>
</tr>
</tbody>
</table>
Figure C.1.1 shows examples of all of the characters that this alternate module is capable of producing.

```
ABCDEFGHJKLMNPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

0123456789 +*<>:)(tltX~O

-------
```

Figure C.1.1: The Extended Character Set.

SECTION C.2: PROCEDURES UGPTOX AND UGITOP

To create text containing these additional characters requires that the programmer generate character strings whose individual characters have unusual bit patterns. These strings can then be packed into a graphic element with procedure UGPTXIT and then transmitted to a display device with procedure UGETPUT. The procedures UGPTOX and UGITOP are made available to help the programmer create and manipulate character strings containing these unusual characters.

The arguments basically consist of three character strings, a string of primary characters, a string of secondary characters, and a string of extended characters. Procedure UGPTOX uses the primary and secondary pairs to generate the extended string and procedure UGITOP does the inverse operation. For an example using UGPTOX suppose that the character for a lower case Greek delta is to be put into the third position in the extended string. The table in the preceding section shows that the character pair for this character is "DG". Therefore, before UGPTOX is called, the programmer must put a "D" in the third position of the primary character string and a "G" in the third position of the secondary character string. The symbol " " in the table stands for a blank. Notice that there is a pattern to the secondary characters; lower case Roman letters are designated by an "L", lower case Greek by a "G", upper case Greek by a "H", control functions by a "C", plotting symbols by a "P", and special characters by "A", "S", or "I". Finally, the standard characters are all designated by a blank secondary character.

The calling sequences are:

```
CALL UGPTOX (CPP, CPS, EC);
CALL UGITOP (CPP, CPS, EC);
```
The parameters in the calling sequence are:
- **CPP**: A character string containing the primary character pairs.
- **CPS**: A character string containing the secondary character pairs.
- **EC**: A character string containing the extended characters.

The entry declarations for these procedures are:
```c
DECLARE UGITOP EXTERNAL ENTRY(CHARACTER(*),
    CHARACTER(*),CHARACTER(*));
DECLARE UGITOP EXTERNAL ENTRY(CHARACTER(*),
    CHARACTER(*),CHARACTER(*));
```

No error messages are produced by these procedures.

Figures C.2.1 and C.2.2 give examples of the use of the alternate character stroke generator. Each figure shows the primary and secondary character strings and the corresponding line of text.

**Figure C.2.1: Some Simple Examples.**

```
ADG adg AΔΓ αδδ 2^{2-1}=8
```

```
PRIMARY...ADG ADG ADG ADG 202021-11=8
SECONDARY... LLL HHH 656 C C C C
```

**Figure C.2.2: A More Elaborate Example.**

```
(n-1)!\ne\int_0^\infty e^{-t}t^{n-1}dt
```

```
PRIMARY...(n-1)!=6(n)=\{I40052F05 ED-TITON-11DT
SECONDARY... L L L SCGSEC C LC LCLCL CLL
```
SECTION C.3: MODULE UGX103X

It is possible to write programs for graphic devices which never use the stroke generator; either the device character generator is sufficient or the pictures do not contain any text material. For these programs it is wasteful to have the stroke generator using core space at execution time. Therefore, a second alternate program module is available. If the programmer creates the load module with the LINK-EDITOR and includes the card:

INCLUDEN SYSLIB (UGX103X)

in the SYSIN data set, then this alternate module will be used. This alternate module is much smaller than the standard module. If a program with this alternate module tries to use the stroke generator, then that text material will not appear in the picture.
When the Unified Graphics System detects an error, it has three pieces of information available to it. The first is the name of the procedure that detected the error, the second is an index whose value describes the error, and the third item is the severity level of the error. There are four levels of errors which may be reported. These levels and the actions of the error processor are:

1. Minor Errors: The procedure name and index are saved in some error indicators.
2. Errors: The error indicators are saved and a message is printed. The message contains the procedure name and index.
3. Severe Errors: A message is printed and the program is terminated without a core dump.
4. Terminal Errors: A message is printed and the program is terminated with a core dump.

At execution time, the programmer can check for the first two types of errors and try to recover from them. One way to do this is to check the error indicators after a procedure has been called. The error indicators are declared by the statement:

```plaintext
DECLARE 1 UGHAB STATIC EXTERNAL,
    2 UGHAB CHARACTER(8),
    2 UGIND FIXED BINARY(15,0);
```

When a Minor Error or Error has occurred, UGHAB will contain the procedure name and UGIND will contain the index. If no error has occurred, UGHAB will contain blanks and UGIND will contain zero.

A second and more versatile way of processing errors is by means of a user-written error processing procedure called UGEHRR. If the user includes a procedure with this name, it will be called whenever any error is detected. The skeleton for the procedure is:

```plaintext
UGEHRR: PROCEDURE(LEVEL,NAME,INDEX);
DECLARE (LEVEL,INDEX) FIXED BINARY(15,0);
DECLARE NAME CHARACTER(8);
```

This procedure can do almost anything to try to recover from the error including calling other procedures in this system. However, if other procedures are called, procedure UGEHRR should be declared RECURSIVE since another error may arise before UGEHRR has returned for the first time. Procedure UGEHRR may also change its arguments; in particular, LEVEL may be set to zero to cause the system to suppress further error action.

An example of a non-trivial use of an error processing procedure will now be given. Suppose a program has been written which generates large complicated pictures for the CALCOMP Drum Plotter and suppose it is written as follows:

1. Before each plot, the graphic element is cleared by
calling procedure UGEINT with the CLEAR option.

2. The graphic element generators are used to create the graphic element.

3. The graphic element is transmitted to the CALCOMP.

Eventually this simple program will fail at Step 2 because the graphic element will become full. However, this program can be made to work by doing the following:

1. Make the graphic element an external variable.

2. Prepare a UGIXERR procedure which recognizes a graphic element overflow error as signalled by the graphic element generators (these are distinguished by an index number of 9). In this case UGIXERR should transmit the graphic element to the CALCOMP by calling procedure UGEPUT and then clear the graphic element by calling procedure UGEINT with the CONTINUE option. The CONTINUE option is needed here because of the way procedure UGEINT works. It draws from the last point to the given point. The CONTINUE option causes the last point to be saved. Procedure UGIXERR may then reset the level number to zero and return.

With these modifications and additions, the original program will work because of a property of the basic graphic element generators. When they find that all of the new data will not fit, they report this condition to the error processor. If the error processor returns, the element generators again try to put the data into the graphic element before returning.

SECTION D.1: PROCEDURE UGIX001

When a procedure in this system detects an error, it is reported by calling a procedure named UGIX001. A programmer may also make use of this procedure if he wishes to process errors in this same manner.

The calling sequence is:

CALL UGIX001(LEVEL,NAME,INDEX);

The parameters in the calling sequence are:

LEVEL    The level number of the error. A value of zero will cause the error flags to be cleared.
NAME     The name of the procedure identifying the error.
INDEX    The index of the error.

The entry declaration for this procedure is:

DECLARE UGIX001 EXTERNAL ENTRY(FIXED BINARY(15,0),
    CHARACTER(0), FIXED BINARY(15,0)) ;
This section describes two related functions. Scaling is the operation of taking the given floating point coordinates in the programmer coordinate system and transforming them to the device coordinate system. Scissoring is the operation of deleting those points, parts of lines, and characters which lie outside a rectangular area. Scaling is always necessary and scissoring can be important because a distorted picture can result if coordinates are given which extend beyond the physical limits of a graphic device.

The programmer and device limits are specified as 2 by 2 matrices and are manipulated by procedure USCAL. These matrices have the following form:

\[
\begin{bmatrix}
  p_{x0} & p_{xh} \\
  p_{y0} & p_{yh}
\end{bmatrix}
\quad
\begin{bmatrix}
  d_{x0} & d_{xh} \\
  d_{y0} & d_{yh}
\end{bmatrix}
\]

The programmer limits are given in floating point form and the device limits are in fixed point form. To scale the X coordinate, \( p_{x0} \) maps into \( d_{x0} \), \( p_{xh} \) maps into \( d_{xh} \), and intermediate values are linearly interpolated. Y coordinates are mapped similarly. The picture is scissored at the device limit matrix.

The scissoring of points and characters, those produced by the device character generator, is simple and fast because the item is either in the picture or out of it. The case for line segments is not so simple; a line may be in the picture, partially in, or out of it. A line segment that is partially in the picture may intersect the device limits rectangle once or twice.

An example of a non-trivial use of the scaling and scissoring system will now be given. Suppose a program is to be written which generates 10 inch by 25 inch plots on the 10 inch CALCONF Drum Plotter and the programmer wishes to specify the coordinates on the paper in terms of inches. The proper matrices are:

\[
\begin{bmatrix}
  0.0 & 25.0 \\
  0.0 & 10.0
\end{bmatrix}
\quad
\begin{bmatrix}
  0 & 2500 \\
  0 & 1000
\end{bmatrix}
\]

Suppose, at some later time, the programmer wishes to display these pictures on an IBM 2250 with the image using the full width of the screen and centered on the screen vertically. One inch on the CALCONF will correspond to 4095/25 raster units on the IBM 2250. Therefore the height of the image is 10*(4095/25)=1638 raster units. Since the height of the picture on the IBM 2250 is 4095 raster units, this leaves (4095-1638)/2=1229 raster units above and below the image. Therefore, the proper matrices for the IBM 2250 are:
\[
\begin{bmatrix}
0.0 & 25.0 \\
0.0 & 10.0 
\end{bmatrix}
\begin{bmatrix}
0 & 4095 \\
1229 & 2867 
\end{bmatrix}
\]

The same graphic elements that were generated for the CALCOMP could be transmitted to the IBM 2250 to produce the identical, but smaller, image with these limits. Scissoring would take place in exactly the same way in both cases.

Suppose further, that the programmer now wished to enlarge the upper right 3 inch by 3 inch section of the CALCOMP plot and display it full screen on the IBM 2250. These matrices are:

\[
\begin{bmatrix}
22.0 & 25.0 \\
7.0 & 10.0 
\end{bmatrix}
\begin{bmatrix}
0 & 4095 \\
0 & 4095 
\end{bmatrix}
\]

After these matrices are made available to the system, the graphic elements could again be transmitted to produce the desired picture.

Some warnings must be given in regard to this manipulation of scaling and scissoring limits:

1. The manipulations described in the preceding example are completely device dependent. The same manipulations could, however, be done in a much more device independent manner. This can be done by first reading the default device limits to get the upper and lower limits of the device. For example if the retrieved default upper Y limit of the IBM 2250 were used instead of 4095 in the above calculation, then the resulting program would be valid for both the IBM 2250 and the GIF-IDIDC.

2. No check is made in procedure UGSCAL to see if the given device limits are valid for the device. For example, on the IBM 2250, negative lower limits or upper limits larger than 4095 are not detected as being erroneous.

3. In the horizontal direction, the number of device units per programmer unit is \((DXHI-DXLO)/(PXHI-PXLO)\). In the vertical direction this ratio is \((DYHI-DYLO)/(PYHI-PYLO)\). Under normal conditions these two ratios should be the same. If the ratios are not the same, certain operations performed by the character stroke generator are ambiguous and the programmer could be surprised or unhappy over the result.

SECTION E.1: MODULE UG1X112Z

It is possible to write programs for a graphic device which never need the scissoring system because the programmer has assured that nothing will ever be positioned outside the device limits. For these programs it is wasteful to have the scissoring module using core space and slowing the computation at execution time. Therefore an alternate program module is available. If the
programmer creates the load module with the LINK-EDITOR and includes the card:

  INCLUDE SYSLIB(UGX112Z)

in the SYSIN data set, then this alternate module will be used. This alternate module is much smaller and considerably faster because no scissoring is done.
Many of the procedures in this system contain a character string argument to specify optional or device-dependent information. This section describes the procedure that is used to scan these option lists to obtain the information from them. The programmer may make use of this procedure.

SECTION F.1: PROCEDURE UGX002

This procedure may be used to scan a character string for certain items. The items to be searched for are described in an input structure, and the results of the scan are inserted into an output structure.

The calling sequence is:
CALL UGX002 (OPTIONS, INDATA, ENDATA);

The parameters in the calling sequence are:
OPTIONS The options list which will be scanned.
INDATA An input structure which specifies the items to be scanned for and the format of the output array. The items which may be scanned for are of five types:
1. A simple flag.
2. A flag, followed by an equals sign, followed by a fixed point number.
3. A flag, followed by an equals sign, followed by a floating point number.
4. A flag, followed by an equals sign, followed by a string of characters (this string of characters cannot include blanks or commas).
5. A flag, followed by an equals sign, followed by a string of bits (zeros or ones).

INDATA consists of halfwords and character strings intermixed. Its basic format consists of:
1. The number of descriptors in INDATA.
2. Descriptors of the items to be scanned for.
The descriptors have the following format:
1. The descriptor type (1 thru 5).
2. The number of bytes in the flag.
3. A byte offset in the output array where the result of finding a match is to be stored.
4. A type-dependent value (see below).
5. The flag (padded on the right to an even number of bytes).
The type-dependent value for the 5 types has the following meaning:
1. A value which is stored into the two bytes in the output structure if the byte offset is positive, or a value which is OR'ed into
the two bytes if the byte offset is not positive.
2. Not used. The number is stored in the output structure as a 2 byte integer.
3. Not used. The number is stored in the output structure as a 4 byte floating point number.
4. The number of bytes in the output structure which are available to hold the characters. Unspecified characters are set to blanks.
5. The number of bytes in the output structure which are available to hold the bits. Unspecified bits will be set to zero.

EXDATA An output structure which will have the processed information stored in it.
The input and output structures should be passed as the first element in the structure.

The entry declaration for this procedure is:
DECLARE UG1002 EXTERNAL ENTRY(CHARACTER(*),..):

No error messages are produced by this procedure. Any unrecognizable items in the options list are ignored.

The following example may help explain this procedure. Suppose we wish to scan for items of the form 'CAL/10D', or 'DDNAME=<string>'. First let us specify the output structure. In this structure we need a two byte integer for the first item and an eight byte string for the second item. Therefore:

DECLARE 1 EXDATA,
  2 FLAG FIXED BINARY(15,0),
  2 DNUM CHARACTER(6);

Now that the byte offsets in the output array are known, we can specify the input array as follows:
DECLARE 1 INDATA STATIC,
  2 BYTE FIXED BINARY(15,0) INITIAL(2),
  2 NT1A(4) FIXED BINARY(15,0) INITIAL(1,7,0,5),
  2 NT1B CHARACTER(8) INITIAL('CAL/10D'),
  2 NT2A(4) FIXED BINARY(15,0) INITIAL(4,6,2,0),
  2 NT2B CHARACTER(6) INITIAL('DDNAME');

With these structures defined, we may now write code to initialize the output array and call the options scanning routine as follows:

EXDATA.FLAG=0; EXDATA.DNUM=(8) ' ';
CALL UG1002(OPTIONS,INDATA,BNEN,EXDATA.FLAG);

If OPTIONS contained 'CAL/10D' then EXDATA.FLAG will be changed to 5; if OPTIONS contained 'DDNAME=<string>' then EXDATA.DNUM will be changed so that it contains the given string.
This section describes, in detail, the content of a graphic element. Normally the user will construct graphic elements using the procedures described in the section on Graphic Element Generation and will not need the information in this section. If the programmer uses this information to construct graphic elements directly, it should be remembered that little checking for invalid graphic elements is done at execution time.

A graphic element is an array of full words reserved to contain picture description information. Basically, the content of a graphic element is a few items of information at the beginning, followed by "blocks" of picture data. Each block begins with a "block specification" and contains either points, lines, text, or other miscellaneous information. In detail, this is:

ELEMENT(1): Current length of the graphic element (I).
ELEMENT(2): The index of the start of the first block (J).
ELEMENT(3): The index of the start of the last block (K).

... ELEMENT(J): The start of the first block.
... ELEMENT(K): The start of the last block.
... ELEMENT(I+1): The maximum length of the graphic element.

Notice that if a graphic element is to be saved in a data set, ELEMENT(I) gives the number of words which must be written. The value in ELEMENT(I+1) need not be saved because its value will depend on the size of the array that will contain the graphic element when it is retrieved.

When procedure UGIENT is called with the CLEAR option, it resets the graphic element to the following:

ELEMENT(1): 3
ELEMENT(2): 4
ELEMENT(3): 0
ELEMENT(4): HBOUND(ELEMENT,1)-1

When the RESET option is used, UGIENT sets:

ELEMENT(ELEMENT(1)+1): HBOUND(ELEMENT,1)-1

The CONTINUE option instructs UGIENT to do the same thing as for CLEAR unless the last block specification was for line data. In that case it sets:

ELEMENT(1): 1
ELEMENT(2): 4
ELEMENT(3): 4
ELEMENT(4)...ELEMENT(I): A copy of the block specification for the lines and the last end point from the line data with the blanking bit set to '0'.
ELEMENT(I+1): HBOUND(ELEMENT,1)-1

The graphic element generators work by moving the words with the maximum length down, and inserting a block of data into the element. ELEMENT(1), and if necessary ELEMENT(3), are then
reset. The graphic element generators use the pointer to the last block to avoid adding a new block specification when it is not needed.

A point data block consists of the following:

- **ELEMENT(I):** The first halfword contains the characters 'P' and the second halfword contains the number of full words in the block.

- **ELEMENT(I+1):** The intensity, color, and wink bits. Bits 4 through 7 are reserved for VERT, BRT, DINT, and VDIR; bits 8 through 11 are for COL0, COL3, COL2, and COL1; bit 15 is reserved for WINK.

Following this are the X and Y coordinates in floating point form.

A line data block consists of the following:

- **ELEMENT(I):** The first halfword contains the characters 'L' and the second halfword contains the number of full words in the block.

- **ELEMENT(I+1):** The intensity, color, wink, and line structures bits. Bits 28 through 31 are reserved for DOTS, DDSH, DASH, and SOLD.

Following this are the X and Y coordinates in floating point form of the end points of the line segments. The blanking bit is in bit 31 of the Y coordinate.

A text data block consists of the following:

- **ELEMENT(I):** The first character contains 'C'; the second contains '1' (for stroke generator only), '2' (for character generator if possible), or '3' (for character generator only); and the second halfword contains the number of full words in the block.

- **ELEMENT(I+1):** The intensity, color, and wink bits.

- **ELEMENT(I+2):** The rotation angle in floating point form.

- **ELEMENT(I+3):** Either the character spacing for the stroke generator in floating point form or the character sizes VLINE, LANGE, SHAL, and VSHE coded in bits 28 through 31.

Following this may be an arbitrary number of sub-blocks consisting of the following:

- **ELEMENT(I):** X coordinate of the center of the first character in floating point form.

- **ELEMENT(I+1):** Y coordinate of the center of the first character in floating point form.

- **ELEMENT(I+2):** The start of the characters, four to a word and one to a byte, with the first byte containing the number of characters.

A data block for invoking element-subroutines consists of the following:

- **ELEMENT(I):** The first halfword contains the characters 'I' and the second halfword contains the number of full words in the block.

- **ELEMENT(I+1):** The intensity, color, wink, and line structure bits.
ELEMENT(I+2): The rotation angle in floating point form.
ELEMENT(I+3): The scaling value in floating point form.
Following this may be an arbitrary number of sub-blocks consisting of the following:
ELEMENT(I): X coordinate of the relative origin of the element-subroutine in floating point form.
ELEMENT(I+1): Y coordinate of the relative origin of the element-subroutine in floating point form.
ELEMENT(I+2): The first halfword is not used; the second halfword contains the identification of the element-subroutine.

A data block containing information which is to be transmitted to the display device in unmodified form consists of the following:
ELEMENT(I): The first halfword contains the characters 'X' and the second halfword contains the number of full words in the block.
ELEMENT(I+1): The intensity, color, wink, and line structure bits.
Following this may be an arbitrary number of sub-blocks consisting of the following.
ELEMENT(I): The start of the information, four bytes to the word, with the first byte containing the number of bytes in the sub-block.
INDEX

A
Active Device, 3, 18
ALIAS=value (options item), 22, 54, 56
ANGL=value (options item), 6, 15, 36, 65

B
Barlow, J., 14
BEEP (options item), 22, 60, 69

C
CALCOMP Drum Plotter, 1-2, 39-41, 47-69, 78-79, 80
CALCOMP Microfilm Plotter, 49-52
CAL/PCH (options item), 17, 50
CAL/10D (options item), 17, 48
CAL/29D (options item), 17, 48
CAL/65H (options item), 17, 50
Character Size Options, 3, 5, 8, 29, 49-50, 52-53, 58-59, 60, 62, 65, 66, 68-69
Character Stroke Generator, 3, 8, 30-35, 71-76
CLEAR (options item), 6, 22, 48, 51, 54, 56, 60, 66, 69, 79, 85
Color Code Options, 5, 7, 8, 9, 10, 15, 16, 29, 30
CONTINUE (options item), 6, 79, 85

D
DDNAME=value (options item), 17, 48, 50, 53, 56, 60, 65, 69
Default Values, 5, 48, 51, 54, 56, 60, 65, 69
DEFDC (options item), 21, 22, 60, 66
DEVC (options item), 19
DISK (options item), 17, 24, 56
Dixon, W. J., 32

E
Element, see Graphic Element
Element-Subroutine, 15, 21, 63
Error Processor, 4, 39, 70-79
ESEQ (options item), 23, 66
ESOS (options item), 23, 66
ESTBC (options item), 24, 66
EUSB (options item), 27, 66
EYED=value (options item), 33

F
FRAME (attention), 26, 60, 63-64, 66
FRAME (options item), 22, 66
Franke, B., 14
FULSCC (options item), 49-51, 52-54, 68-69

G
GET (options item), 19
Giamb, T., 32
GIF/UI (options item), 17, 65
GIF/UI Display Console, 2, 61-67, 81
Giulieri, S., 14
Graphic Element, 2, 5-16, 20, 85-87

H
Hidden Line Problem, 13
HIT=value (options item), 11

I
IBM 2250 Display Console, 1-2, 42-46, 58-61, 80-81
IBM2250 (options item), 17, 59
IDI0N Display Console, see
GIF/UI Display Console
IDSC (options item), 17, 68
INCL (options item), 22, 60, 66
INPUT (options item), 24, 55, 56
Intensity Level Options, 5, 7, 8, 9, 10, 15, 16, 29, 30, 50, 53, 65, 66, 67, 69
KBBD (attention), 26, 60, 66, 70
Keyboard Input Buffer, 28-30, 59, 62, 68
Kronmal, R. A., 32
Kubert, B., 14

L
LARGSCR (options item), 49-51
LCASE (options item), 29, 70
LDEC=<value> (options item), 11
LEADER=<value> (options item), 50
Levart, C. B., 32
Light Pen Input Buffer, 30-31, 62-63
Line Structure, 5, 7, 10, 15, 16, 30, 65, 67
LINK-EDITOR, 38, 71, 77, 82
LITES=<value> (options item), 23, 66
LOTH=<value> (options item), 11
LOWER (options item), 13
LPEH (attention), 26, 60, 66
LPSB (attention), 26, 66
LPSE (attention), 26, 66
LVID=<value> (options item), 11
LXOFF=<value> (options item), 11
LYOFF=<value> (options item), 11

M
MAGICUD=<value> (options item), 69
MAXDIL=<value> (options item), 60
MAXHUM=<value> (options item), 59, 65
MAXRESZ=<value> (options item), 59, 65
MILLEN Terminal Manager, 67-69
MINHREZ=<value> (options item), 21, 65, 66
MINHREV (options item), 11
Motion Picture Camera, 24, 63-64

N
NAME=<value> (options item), 25
NECHP (options item), 21, 66
NEHFT (options item), 22, 60, 66
NOCCOM (options item), 13
NOLABEL (options item), 11
NONE, 26
NSTH=<value> (options item), 11

O
OFF (options item), 21, 22, 26, 27, 60, 66
OMIT (options item), 21, 22, 60, 66
ON (options item), 21, 22, 60, 66
OPTIONS Argument, 4, 5, 83-84
Orthogonal Transformation, 13-14, 34-35
Overlay Structures, 38

P
PFEB (attention), 26, 66
PFEN (attention), 26, 66
PFEL (attention), 26, 66
PFEN (attention), 26, 66
PICTID=<value> (options item), 53, 56
PICTSQ=<value> (options item), 53, 56
PROGL (options item), 19
Programmer Coordinate System, 2, 80-81
Projective Transformation, 13-14, 33-34
Prueitt, H. L., 14
PUT (options item), 19

R
RDSCOPE (options item), 17
RESET (options item), 6, 25
Round Numbers, 12, 32
RTNC (options item), 21, 66

S
SCALE=<value> (options item), 15, 65
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCAL8</td>
<td>Scaling and Scissoring, 2-3, 19, 55, 86-81</td>
</tr>
<tr>
<td>SECLOS</td>
<td>(options item), 24, 64, 66</td>
</tr>
<tr>
<td>SHOPEF</td>
<td>(options item), 24, 64, 66</td>
</tr>
<tr>
<td>SPACING=value</td>
<td>(options item), 8, 36, 48, 71</td>
</tr>
</tbody>
</table>

**T**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERTMONII 4013 Display</td>
<td>Terminal, 52-54, 67-70</td>
</tr>
<tr>
<td>TER4013</td>
<td>(options item), 17, 69</td>
</tr>
<tr>
<td>TITLE=value</td>
<td>(options item), 49, 50</td>
</tr>
<tr>
<td>TPCTL</td>
<td>(options item), 23, 69</td>
</tr>
<tr>
<td>TPDES</td>
<td>(options item), 23, 66</td>
</tr>
<tr>
<td>TPERR</td>
<td>(options item), 23, 66</td>
</tr>
<tr>
<td>TPGET</td>
<td>(options item), 23, 66</td>
</tr>
<tr>
<td>TPIAC</td>
<td>(options item), 23, 66</td>
</tr>
<tr>
<td>TPPUT</td>
<td>(options item), 23, 66</td>
</tr>
<tr>
<td>TPBIV</td>
<td>(options item), 23, 66</td>
</tr>
</tbody>
</table>

**U**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UGAXIS</td>
<td>5, 11-12, 32, 48, 50, 53, 55, 59, 65, 69</td>
</tr>
<tr>
<td>UGCALD02</td>
<td>49</td>
</tr>
<tr>
<td>UGCALM02</td>
<td>52</td>
</tr>
<tr>
<td>UGCLOS</td>
<td>16, 18, 48, 51, 54, 56, 68, 69, 70</td>
</tr>
<tr>
<td>UGCSTL</td>
<td>36, 49, 51, 54, 56, 61, 67, 70</td>
</tr>
<tr>
<td>UGCTVL</td>
<td>20, 22-24, 60, 64, 66, 69</td>
</tr>
<tr>
<td>UGBATH</td>
<td>25, 27, 60, 66, 70</td>
</tr>
<tr>
<td>UGDBPS02</td>
<td>57</td>
</tr>
<tr>
<td>UGDXI</td>
<td>12, 32, 49, 51, 54, 56, 60, 67, 70</td>
</tr>
<tr>
<td>UGBATN</td>
<td>25-27, 60, 66, 70</td>
</tr>
<tr>
<td>UGBDIV</td>
<td>37, 49, 52, 54, 56, 61, 67, 70</td>
</tr>
<tr>
<td>UGDITA</td>
<td>5, 16, 50, 65</td>
</tr>
<tr>
<td>UGDGET</td>
<td>20, 24-25, 37, 55, 56</td>
</tr>
<tr>
<td>UGDMF</td>
<td>5, 6, 25, 48, 50, 53, 55, 59, 65, 68, 79, 85</td>
</tr>
<tr>
<td>UGDMV</td>
<td>5, 15, 65</td>
</tr>
<tr>
<td>UGDLIN</td>
<td>5, 7-8, 11-12, 13, 42, 50, 53, 55, 59, 65, 68, 79</td>
</tr>
<tr>
<td>UGDLMS</td>
<td>5, 10, 31, 36, 48, 50, 53, 55, 59, 65, 68</td>
</tr>
<tr>
<td>UGEPN</td>
<td>5, 7, 48, 50, 53, 55, 59, 65, 68</td>
</tr>
<tr>
<td>UGEPIS</td>
<td>5, 9, 48, 50, 53, 55, 59, 65, 68</td>
</tr>
<tr>
<td>UGEPUT</td>
<td>20-21, 48, 51, 54, 56, 60, 65, 66, 69, 75</td>
</tr>
<tr>
<td>UGEXN</td>
<td>5, 8-9, 11-12, 48, 50, 53, 55, 59, 65, 68, 71, 75</td>
</tr>
<tr>
<td>UGGFI002</td>
<td>67</td>
</tr>
<tr>
<td>UGI85002</td>
<td>61</td>
</tr>
<tr>
<td>UGINFO</td>
<td>16, 19-20, 48, 51, 54, 56, 60, 65, 69</td>
</tr>
<tr>
<td>UGIGET</td>
<td>20, 29-30, 60, 66, 70</td>
</tr>
<tr>
<td>UGIPUT</td>
<td>20-29, 60, 66, 70</td>
</tr>
<tr>
<td>UGIXET</td>
<td>30, 31, 67</td>
</tr>
<tr>
<td>UGIPUT</td>
<td>30-31, 67</td>
</tr>
<tr>
<td>UGIVHE</td>
<td>16-18, 48, 50, 53, 55, 59, 65, 69</td>
</tr>
<tr>
<td>UGIXET</td>
<td>14, 30-35, 49, 51, 54, 56, 60, 66, 70</td>
</tr>
<tr>
<td>UGIPCT</td>
<td>20, 22, 48, 51, 54, 56, 60, 66, 69</td>
</tr>
<tr>
<td>UGIPROJ</td>
<td>33, 34, 35-36, 49, 51, 54, 56, 61, 67, 70</td>
</tr>
<tr>
<td>UGIPTOX</td>
<td>49, 52, 54, 56, 61, 67, 70, 75-76</td>
</tr>
<tr>
<td>UGIBTH</td>
<td>25-26, 27-28, 60, 66, 70</td>
</tr>
<tr>
<td>UGISCAL</td>
<td>16, 19, 48, 51, 54, 55, 56, 60, 65, 69, 80</td>
</tr>
<tr>
<td>UGISECT</td>
<td>16, 18-19, 48, 51, 54, 56, 60, 65, 69</td>
</tr>
<tr>
<td>UGIE1302</td>
<td>70</td>
</tr>
<tr>
<td>UGIEVIEW</td>
<td>14, 33-34, 35, 49, 51, 54, 56, 60, 67, 70</td>
</tr>
<tr>
<td>UGIXL02</td>
<td>54</td>
</tr>
<tr>
<td>UGIXERR</td>
<td>70-79</td>
</tr>
<tr>
<td>UGIXTOP</td>
<td>49, 52, 54, 56, 61, 67, 70, 75-76</td>
</tr>
<tr>
<td>UGIXO00</td>
<td>38</td>
</tr>
<tr>
<td>UGIXO01</td>
<td>49, 52, 54, 57, 61, 67, 70, 79</td>
</tr>
<tr>
<td>UGIXO02</td>
<td>49, 52, 54, 57, 61, 67, 70, 83-84</td>
</tr>
<tr>
<td>UGIX1031</td>
<td>71</td>
</tr>
<tr>
<td>UGIX1032</td>
<td>77</td>
</tr>
<tr>
<td>UGIX1122</td>
<td>81</td>
</tr>
<tr>
<td>UG3DHS</td>
<td>5, 13-15, 48, 50, 53, 55, 59, 65, 69</td>
</tr>
</tbody>
</table>

**V**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIAN</td>
<td>620/I, 61-65</td>
</tr>
<tr>
<td>VERT</td>
<td>(options item), 11</td>
</tr>
</tbody>
</table>
Williamson, H., 14
Wink Node Options, 5, 7, 8, 9, 10, 15, 16, 65
Wright, T., 14
WYLBUR Text Editor, 52-53, 67-68
WYLJ013 (options item), 17, 53

x
I
X=<value> (options item), 36
XHI=<value> (options item), 33, 35
XLO=<value> (options item), 33, 35
XSPACING=<value> (options item), 8, 48, 71
XSYSTEM=<value> (options item), 17

y
Y=<value> (options item), 36
YHI=<value> (options item), 33, 35
YLO=<value> (options item), 33, 35

3-D Viewer, 64