# 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 SUBROUTINES</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>A DETAILED DESCRIPTION OF THE SUBROUTINES</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>SUBROUTINE UGELMT</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2</td>
<td>SUBROUTINE UGELMT</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3</td>
<td>SUBROUTINE UGELMT</td>
<td>7</td>
</tr>
<tr>
<td>2.1.4</td>
<td>SUBROUTINE UGELMT</td>
<td>8</td>
</tr>
<tr>
<td>2.1.5</td>
<td>SUBROUTINE UGELMT</td>
<td>9</td>
</tr>
<tr>
<td>2.1.6</td>
<td>SUBROUTINE UGELMT</td>
<td>10</td>
</tr>
<tr>
<td>2.1.7</td>
<td>SUBROUTINE UGAXIS</td>
<td>11</td>
</tr>
<tr>
<td>2.1.8</td>
<td>SUBROUTINE UG3DHS</td>
<td>13</td>
</tr>
<tr>
<td>2.1.9</td>
<td>SUBROUTINE UG3DHS</td>
<td>15</td>
</tr>
<tr>
<td>2.1.10</td>
<td>SUBROUTINE 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>SUBROUTINE UGOPEN</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>SUBROUTINE UCLOSE</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3</td>
<td>SUBROUTINE UGSLCT</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4</td>
<td>SUBROUTINE UGSCLL</td>
<td>18</td>
</tr>
<tr>
<td>2.2.5</td>
<td>SUBROUTINE UGINF0</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>SUBROUTINE UGREAD</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2</td>
<td>SUBROUTINE UGWRITE</td>
<td>21</td>
</tr>
<tr>
<td>2.3.3</td>
<td>SUBROUTINE UGCTRL</td>
<td>22</td>
</tr>
<tr>
<td>2.3.4</td>
<td>SUBROUTINE UGHEIGHT</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>SUBROUTINE UGREAD</td>
<td>26</td>
</tr>
<tr>
<td>2.4.2</td>
<td>SUBROUTINE UGDATA</td>
<td>26</td>
</tr>
<tr>
<td>2.4.3</td>
<td>SUBROUTINE UGDATA</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>KEYBOARD INPUT BUFFER CONTROL</td>
<td>27</td>
</tr>
<tr>
<td>2.5.1</td>
<td>SUBROUTINE UGINTER</td>
<td>28</td>
</tr>
<tr>
<td>2.5.2</td>
<td>SUBROUTINE UGWRITE</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>LIGHT PEN INPUT BUFFER CONTROL</td>
<td>29</td>
</tr>
<tr>
<td>2.6.1</td>
<td>SUBROUTINE UGINTER</td>
<td>29</td>
</tr>
<tr>
<td>2.6.2</td>
<td>SUBROUTINE UGWRITE</td>
<td>30</td>
</tr>
<tr>
<td>2.7</td>
<td>MISCELLANEOUS SUBROUTINES</td>
<td>31</td>
</tr>
<tr>
<td>2.7.1</td>
<td>SUBROUTINE UGID</td>
<td>31</td>
</tr>
<tr>
<td>2.7.2</td>
<td>SUBROUTINE UGVIEW</td>
<td>32</td>
</tr>
<tr>
<td>2.7.3</td>
<td>SUBROUTINE UGORTHE</td>
<td>33</td>
</tr>
<tr>
<td>2.7.4</td>
<td>SUBROUTINE UGPROJ</td>
<td>34</td>
</tr>
<tr>
<td>2.7.5</td>
<td>SUBROUTINE UGCTOL</td>
<td>35</td>
</tr>
<tr>
<td>2.7.6</td>
<td>SUBROUTINE UGDIV</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>USING THE SLAC UNIFIED GRAPHICS SYSTEM</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>PROGRAMMING EXAMPLES</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>PLOTTING A MESH SURFACE</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>INTERACTIVE LISSAJOUS FIGURES</td>
<td>40</td>
</tr>
</tbody>
</table>
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>46</td>
</tr>
<tr>
<td>A.1</td>
<td>THE CALCOMP DRUM PLOTTERS (MODELS 564/565)</td>
<td>46</td>
</tr>
<tr>
<td>A.2</td>
<td>THE CALCOMP MICROFILM PLOTTER (MODEL 1675)</td>
<td>48</td>
</tr>
<tr>
<td>A.3</td>
<td>WILBUR DISPLAY FILES FOR THE TEKTRONIX 4013</td>
<td>51</td>
</tr>
<tr>
<td>A.4</td>
<td>SAVING PICTURES IN A PARTITIONED DATA SET</td>
<td>54</td>
</tr>
<tr>
<td>B</td>
<td>INTERACTIVE DISPLAY DEVICES</td>
<td>57</td>
</tr>
<tr>
<td>B.1</td>
<td>THE IBM 2250 DISPLAY CONSOLE</td>
<td>57</td>
</tr>
<tr>
<td>B.2</td>
<td>THE GIF-IDITCH DISPLAY CONSOLE</td>
<td>60</td>
</tr>
<tr>
<td>B.3</td>
<td>THE TEKTRONIX 4013 DISPLAY TERMINAL</td>
<td>66</td>
</tr>
<tr>
<td>C</td>
<td>THE CHARACTER STICK GENERATING SUB-SYSTEM</td>
<td>70</td>
</tr>
<tr>
<td>C.1</td>
<td>MODULE UGX103A</td>
<td>70</td>
</tr>
<tr>
<td>C.2</td>
<td>SUBROUTINES UGPTOX AND UGXTOP</td>
<td>74</td>
</tr>
<tr>
<td>C.3</td>
<td>MODULE UGX103Z</td>
<td>75</td>
</tr>
<tr>
<td>D</td>
<td>THE ERROR PROCESSING SUB-SYSTEM</td>
<td>77</td>
</tr>
<tr>
<td>D.1</td>
<td>SUBROUTINE UGX001</td>
<td>78</td>
</tr>
<tr>
<td>D.2</td>
<td>SUBROUTINE UGR203</td>
<td>79</td>
</tr>
<tr>
<td>D.3</td>
<td>MODULE UGX001A</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 UGX112Z</td>
<td>81</td>
</tr>
<tr>
<td>F</td>
<td>THE OPTIONS SCANNING SUB-SYSTEM</td>
<td>83</td>
</tr>
<tr>
<td>F.1</td>
<td>SUBROUTINE UGX002</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.
SECTION 1: INTRODUCTION TO THE SLAC UNIFIED GRAPHICS SYSTEM

The SLAC Unified Graphics System is a collection of PL/I 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 FORTRAN 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-IDIION 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 subroutines: UGINT, UGEPET, UGELIN, UGEXIT, UGOPEN, UGCLOS, UGEPUT, and UGPICT. If you are interested in interactive graphic devices, then you should also read about: UGCCTRL, UGEBATN, UCDAATH, UGBATN, UGEPUT, and UGEBAT.
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 FORTRAN programmer has had a set of subroutines available for using each of these devices. One of the problems has been that the subroutines available for one device have been totally different from those of another device. For instance, if a programmer had mastered the subroutines for producing CALCOMP plots, it was necessary to learn an entirely different set of subroutines 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 subroutines 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 FORTRAN programmer will use exactly the same set of subroutines 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 subroutines 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-IDIION 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 FORTRAN 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 SUBROUTINES

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 subroutines 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 subroutine 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. Subroutines are supplied in this
system for enabling and disabling attentions. Another subroutine
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
subroutines in this system communicate with the active device
only. A subroutine is available to make any display device the
active one.
SECTION 2: A DETAILED DESCRIPTION OF THE SUBROUTINES

This section gives a complete description of each of the subroutines provided in the FORTRAN version of the SLAC Unified Graphics System. The names of these subroutines, and all other external names, start with the letters "UG" (standing for Unified Graphics). If the user of these subroutines 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 subroutines is a character string (named OPTNS) which is used to specify information to the system which may be optical 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. The character string must be terminated by an asterisk. Thus the line:

'SHAL, ANGLE=5.3, DDNAME=EQRS, 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 subroutines on the following pages include all of the items which can be utilized by any of the devices. Exactly which items in OPTNS 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 subroutines 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 subroutine 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 subroutine include a list of all the error conditions, for all devices, which may be produced by the subroutine. Both the index and level number (in parentheses) as well as a short description are
Many of the subroutines have arguments which are described in this document as character strings. These arguments may be INTEGER or LOGICAL arrays of any precision.

SECTION 2.1: GRAPHIC ELEMENT GENERATION

The subroutines 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:

```
INTEGER ELENDT(500)
```

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

The first of the graphic element generation subroutines is UGEN 수. This subroutine is used to initialize a graphic element before any picture description data is added to it. The next three subroutines, UGENF, UGENL, and UGEIN, 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 subroutines, UGEPIS and UGELNS, are similar to UGENF and UGENL except that each call to these two subroutines will add an array of points or line segments to the graphic element. Subroutines UGAXIS and UG3DNS 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 subroutines UGENBV and UGENA 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 OPTINS argument in these subroutines 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:

- **VDIM, DBMS, DDRT, BERT**: Intensity levels. Any of four different intensity levels may be given. The key words stand for "very dim", "dim", "bright", and "very bright".

- **VINK, STDY**: Vink 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, CCL3, COL4**: Color codes. Any of four colors may be specified.

- **SOLD, DASH, DDAN, 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".

VSNL,SHAL,LARG,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 SHAL.

In addition to using these subroutines 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 subroutines described in this section are completely independent of the rest of the subroutines 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: SUBROUTINE UGEINT

This subroutine may be used to clear and initialize a graphic element. After this subroutine has been called, other subroutines 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:
CALL UGEINT (OPTNS, ELEMT, ELTSIZ)

The parameters in the calling sequence are:
OPTNS A character string which may contain one of the following items:
  CLEAR This is the normal use of this subroutine. The element is cleared and made ready to accept more picture description data.
  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.
  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.

ELEMNT  The graphic element which is to be initialized.
ELTSIZ  The number of words in ELEMNT (INTEGER*4).

The errors detected by this subroutine are:
1(3): The length of the array ELEMNT is too small.

SECTION 2.1.2: SUBROUTINE UGEPNT

This subroutine 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 UGEPNT (OPTNS, X, Y, ELEMNT)

The parameters in the calling sequence are:
OPTNS  A character string which may contain any of the 
       following items:
       VDIN,DIMN,BRIN,VBRT  Intensity levels.
       MFRN,STDI  Link mode specification.
       COL1,COLF,COL3,COLF  Color codes.
       X  X coordinate of the point (REAL*4).
       Y  Y coordinate of the point (REAL*4).
       ELEMNT  The graphic element which will have the point added to 
it.

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

SECTION 2.1.3: SUBROUTINE UGELIN

This subroutine 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 UGELIN (OPTNS, X, Y, BEIT, ELEMNT)

The parameters in the calling sequence are:
OPTNS  A character string which may contain any of the 
       following items:
SOLD, DASH, DDDS, DOTS Line structure specification.

VDIM, DIMH, BBIT, VERT Intensity levels.

WINK, STBY Wink mode specification.

COL1, COL2, COL3, COL4 Color codes.

X X coordinate of an end point of a line (REAL*4).

Y Y coordinate of an end point of a line (REAL*4).

BBIT The blanking bit. The line segment formed in moving to the given point is drawn if BBIT is a one, and blanked if BBIT is a zero (INTEGER*4).

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

The errors detected by this subroutine are:

9(1): Not enough room was available in ELEMENT to contain the new end point.

SECTION 2.1.4: SUBROUTINE UGETIT

This subroutine 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 \( \lceil \frac{N}{4} \rceil + 3 \) words of information are added to the graphic element where \( N \) is the number of characters in the text and \( \lceil X \rceil \) is the greatest integer in \( X \); otherwise \( \lceil \frac{N}{4} \rceil + 7 \) words are added.

The calling sequence is:

CALL UGETIT (OPTNS, X, Y, TEXT, NTXT, ELEMENT)

The parameters in the calling sequence are:

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

VSBL, SHAL, LANG, VLEG 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.

ISPACING=\langle value \rangle Suggested character spacing. The device character generator will be used if the between character spacing will approximately match the given value.

SPACING=\langle value \rangle Character spacing. The programmed stroke generator must be used.

ANGLE=\langle value \rangle The angle that the characters make with the horizontal. The angle is measured in the counter-clockwise direction.

VDIM, DIMH, BBIT, VERT Intensity levels.

WINK, STBY 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: X coordinate of the center of the first character (REAL*4).
Y: Y coordinate of the center of the first character (REAL*4).
TEXT: The string containing the characters to be added to the graphic element.
NUM: The number of characters in TEXT (INTEGER*4).
ELEMENT: The graphic element which will have the text information added to it.

The errors detected by this subroutine 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: SUBROUTINE UGETS

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

The calling sequence is:
CALL UGETS(OPTMS,YABRAY,YABRAY,NPNTS,ELEMENT)

The parameters in the calling sequence are:
OPTMS: A character string which may contain any of the following items:
       VDIM, DIM, BRIT, VERT Intensity levels.
       WINK, STBY Wink mode specification.
       CO1, CO2, CO3, CO4 Color codes.
YABRAY: An array containing the X coordinates of the points (REAL*4, DIMENSION=NPTs).
YABRAY: An array containing the Y coordinates of the points (REAL*4, DIMENSION=NPTs).
NPNTS: The number of points in YABRAY and YABRAY (INTEGER*4).
ELEMENT: The graphic element which will have the points added to it.

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

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

The calling sequence is:

CALL UGEINS (OPTNS, XARRAY, YARRAY, NPTS, BBITS, NBBITS, ELEMNT)

The parameters in the calling sequence are:

OPTNS A character string which may contain any of the following items:
SOLID, DASH, DOTS Line structure specification.
VDIM, DIMN, 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 (REAL*, DIMENSION=NPTS).

YARRAY An array containing the Y coordinates of the end points of the line segments (REAL*, DIMENSION=NPTS).

NPTS The number of end points in XARRAY and YARRAY (INTEGER*4).

BBITS The blanking bits. The bits are used cyclically from BBITS. A single blanking bit of one would result in a continuous curve; two blanking bits of one-zero results in a line being drawn between the first and second points, the third and fourth, etc. (INTEGER*4, DIMENSION=NBBITS or (31-NBBITS)/32).

NBBITS The number of blanking bits in BBITS (INTEGER*4). If NBBITS is positive, the bits must be given one to a word; if NBBITS is negative, its absolute value is used for the count and the bits must be packed 32 to a word. When unpacked bits are used, the first blanking bit is the low order bit of BBITS(1); when packed bits are used, the first blanking bit is the high order bit of BBITS(1).

ELEMNT The graphic element which will have the line segments added to it.

The errors detected by this subroutine are:

9(1): Not enough room was available in ELEMNT to contain the new line segments. The element is unchanged.
SECTION 2.1.7: SUBROUTINE UGAXIS

This subroutine may be used to add the description of a horizontal or vertical axis with labels and tic marks to a graphic element. This subroutine adds the actual data to the graphic element by calling subroutines UGELIN and UGETIT. If either of these subroutines signal 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 (OPTNS, LOPNS, CLOPTNS, INEVAR, LOVAR, HIVAL, LOLAB, HILAB, WLAB, HIENT)

The parameters in the calling sequence are:

OPTNS 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.

HINTV 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.

XOLABL 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; the given value must be between 1 and 9.

IDEC=<value> The number of places to the right of the decimal point in the labels, the default value is 2; the given value must be between 0 and 9.

IXOFF=<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.

IYOFF=<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.

NSTM=<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.

DUNIT=<value> This subroutine uses the library subroutine F10999 to create the labels. F10999 requires a dummy unit number, the default value is 99.

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

LOPTNS An options list that will be passed to UGELIN.

COPTNS An options list that will be passed to UGETIT.

INDVAR For a horizontal axis this is the X coordinate of the axis in the programmer coordinate system, for a vertical axis this is the Y coordinate (REAL*4).

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 (REAL*4).

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 (REAL*4).

LOLAB The label to be placed at the low end of the axis (REAL*4).

HILAB The label to be placed at the high end of the axis (REAL*4).

NLAB The number of labels and primary tic marks to be put on the axis (INTEGER*4).

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

The reader should refer to subroutine UGDXY in the section on miscellaneous Subroutines for help in assigning values to LOLAB, HILAB, and NLAB which results in "round numbers" being used for labels.

The errors detected by this subroutine are:

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

This subroutine 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 Z 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 subroutine adds the actual data to the graphic element by calling subroutine 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 subroutine can generate a large amount of data which it will try to add to the graphic element.

This subroutine 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 (OPTNS, LOPTNS, ARRAY, HDIM, NDIM, TRANS, USEAREA, ELEM, ELEMT)

The parameters in the calling sequence are:
OPTNS 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 subroutine are necessary to generate the full view of a surface.
NOCONN 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.
LOPTNS 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. \((\text{REAL}^n, \text{DIMENSION}=(\text{MDIM}, \text{NDIM}))\). The format of ARRAY is:

\[
\begin{align*}
&\quad \text{X1} \quad \text{X2} \ldots \text{XN} \\
&\quad \text{Y1} \quad \text{Z11} \quad \text{Z21} \ldots \text{ZN1} \\
&\quad \text{Y2} \quad \text{Z12} \quad \text{Z22} \ldots \text{ZN2} \\
&\quad \ldots \\
&\quad \text{YN} \quad \text{Z1M} \quad \text{Z2M} \ldots \text{ZNM}
\end{align*}
\]

The sequences \((\text{X1}, \text{X1}, \ldots, \text{XN})\) and \((\text{Y1}, \text{Y2}, \ldots, \text{YN})\) 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.

NDIM The first dimension of ARRAY (INTEGER*4).

NDIM The second dimension of ARRAY (INTEGER*4).

TRANS An array containing the transformation as produced by subroutine UGVIEW or UGORTH \((\text{REAL}^n, \text{DIMENSION}=(31))\). 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 EDIS.

UAREA An array which will be used as a work area \((\text{REAL}^n, \text{DIMENSION}=(\text{LDIM}))\). 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.

LDIM The dimension of UAREA (INTEGER*4).

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

The errors detected by this subroutine are:

1(3): The array UAREA is not large enough.

9(1): Not enough room was available in ELEMNT 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 UG3DMS is based on the information in [3,4].


5. M. L. Prueitt, Fantastic Computer Pictures Give Us a
SECTION 2.1.9: SUBROUTINE UGEINV

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 subroutine may be used to invoke an element-subroutine from another element.

The calling sequence is:
CALL UGEINV (OPTNS,X,Y,IDENT,ELEMNT)

The parameters in the calling sequence are:

OPTNS A character string which may contain any of the following items:
- SOLD,DASH,DDSH,DOTS Line structure specification.
- UDIN,DIHN,BRT,VERT Intensity levels.
- WINK,STDY Wink mode 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 (REAL*4).

Y Y coordinate of the relative origin of the element-subroutine (REAL*4).

IDENT The identification of the element-subroutine to be invoked (INTEGER*4).

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

The errors detected by this subroutine are:

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

This subroutine 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 utilization cannot be accomplished with the information in this document.

The calling sequence is:
CALL UGEDTA (OPTNS, STRING, NSTR, ELEMNT)

The parameters in the calling sequence are:
- OPTNS   A character string which may contain any of the following items:
  SOLD, DASH, DDSH, DOTS Line structure specification.
  VDIN, DIMM, BBIT, VBNT Intensity levels.
  WINK, STBY Wink mode specification.
  COL1, COL2, COL3, COL4 Color codes.
- STRING A character string containing the information to be added to the display element.
- NSTR   The number of characters in STRING (INTEGER).
- ELEMNT The graphic element which will have the given information added to it.

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

SECTION 2.2: GRAPHIC DATA SET CONTROL

The subroutines in this section perform basic operations on a graphic device. Subroutine UGOPEN is used to open a graphic device, while UGCLOS 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: SUBROUTINE UGOPEN

This subroutine 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 device.
The calling sequence is:
```
CALL UGOPEN (OPTUS, IDENT)
```

The parameters in the calling sequence are:
- **OPTUS**: 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/60H The CALCOMP Microfilm Recorder with 16mm unimicrofilmed film.
  - CAL/FCH The CALCOMP Microfilm Recorder with Microfiche film.
  - NYL4013 NYLBR Display Files for the TERTRONIX 4013 are saved in a partitioned data set.
- **DISKPDS**: Graphic elements, in their device independent form, may be written to, or read from, a partitioned data set.
- **IBM2250**: The IBM 2250 Display Console.
- **GIP/IDI**: The GIP-IDIION Display Console.
- **TER4013**: The TERTRONIX 4013 Display Terminal. This item refers to the terminals which are under the control of the MFTEN 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>. Subroutine UGOPEN will find the internal record of the DD card whose name is given and determine the type of output device to be used.
- **IDSCOPE**: This item selects a refresh display scope: either an IBM 2250 or the GIP-IDION.
- **IDSCOPE**: This item selects an interactive display scope: either an IBM 2250, the GIP-IDION, or a TERTRONIX 4013 running under MFTEN.

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.
- **ISYSTEM=<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 (INTEGER*4).

The errors detected by this subroutine 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: SUBROUTINE UGCLOS

This subroutine 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 (INTEGER*4).

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

SECTION 2.2.3: SUBROUTINE UGSICT

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

The calling sequence is:
CALL UGSICT(IDENT)

The parameter in the calling sequence is:
IDENT The identification of the device which is to become the active device (INTEGER*4).

The errors detected by this subroutine are:
1(3): The identification is invalid; no such device is open.

SECTION 2.2.4: SUBROUTINE UGSCAL

This subroutine 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 (OPTNS, PREGLE, DEVCLN)

The parameters in the calling sequence are:

OPTNS A character string which may contain any of the following items:
GET,PUT Flags to indicate if the limits are to be retrieved or changed.
PREGLE A flag to indicate that the programmer limits are to be retrieved or changed.
DEVCLN A flag to indicate that the device limits are to be retrieved or changed.

PREGLE An array for the programmer scaling limits (REAL*4, DIMENSION=(2,2)).
DEVCLN An array for the device scaling limits (INTEGER*4, DIMENSION=(2,2)).

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

SECTION 2.2.5: SUBROUTINE UGINFO

This subroutine 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, eight characters long, which will be set to contain the specific device identifications described in subroutine UGOPEN.
IDENT This will be set to the identification of the device (INTEGER*4).
ARRAY This array will be set to the device dependent information (INTEGER*4, DIMENSION=5).

The errors detected by this subroutine are:
1(3): No device is active.
SECTION 2.3: DISPLAY DEVICE CONTROL

This section describes those subroutines that are directly related to controlling the picture and the graphic device itself. Subroutine UGEPIT will transmit a graphic element to a display device. Subroutine UGPICT gives the programmer control over the picture. Subroutine UGETBL allows the programmer to control the display device itself. Finally, subroutine UGETGRT 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 "cmit" state (as opposed to the "include" state).

SECTION 2.3.1: SUBROUTINE UGEPIT

This subroutine 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 UGEPIT(OPTNS, IDENT, ELEMNT)
```

The parameters in the calling sequence are:

- **OPTNS** A character string which may contain any of the following items:
  - DETC Make the element light pen detectable.
  - CMIT Leave the element in the cmit state.
  - NCHP Send non-compact orders to the device.
  - CH,OFF Leave the display in the "on" or "off" state.
  - NTN 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, sink node, 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.

**HINESIZE=<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 (IDENTISION).

**ELEMENT** The graphic element which is to be transmitted.

The errors detected by this subroutine 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 subroutine is not supported for the active device.

**SECTION 2.3.2: SUBROUTINE UGPIC**

This subroutine 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 subroutine as long as the operations are not contradictory.

The calling sequence is:

CALL UGPIC (OPTNS, IDENT)

The parameters in the calling sequence are:

**OPTNS** 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.
- **DET,CEDIT** 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 (INTEGRAL).

The errors detected by this subroutine are:

1(3): No device is active.

99(2): This subroutine is not supported for the active device.

SECTION 2.3.3: SUBROUTINE UGCTRL

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

The calling sequence is:

CALL UGCTRL(OPTNS, IARRAY, XARRAY)

The parameters in the calling sequence are:

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

BEEP Sound the audible alarm on the display device.

FRAME The value in IARRAY(1) gives the number of regeneration cycles that should be performed before issuing a FRAME 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 IARRAY(1) and IARRAY(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 IARRAY(1) and IARRAY(2); the coordinates in the device coordinate system are put into IARRAY(1) and IARRAY(2). If the tracking pattern is not on the screen, then IARRAY(1) and IARRAY(2) are set to minus one.

TPENV Remove the tracking pattern from the
display device.

TPSHC 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.

TPDHE Put the display device into the draw mode with respect to light pen input buffers and set the segment length from IABRAY(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.

TPERR 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 TPCHC item.

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

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

ESTBC If the LFEN 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 IABRAY 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.

SCLOS Close the shutter on the synchronized movie camera.

IABRAY An input and output array that is used when certain
items are given in OPThS (INTEGER*4, DIMENSION=10).

XARRAY  An input and output array that is used when certain
items are given in OPThS (REAL*4, DIMENSION=2).

WARNING...The arguments XARRAY and XARRAY must be dimensioned at
least 10 and 2 respectively, even if you are not supplying
information in the arrays and do not expect to receive any
information in them.

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

SECTION 2.3.4: SUBROUTINE UCSGET

This subroutine 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 UCSPUT with a device type of
DISKPS. This subroutine is usable only when the active device
type is DISKPS with the INPUT option.

The calling sequence is:
   CALL UCSGET (OPTHS, ELEMNT, ELTSIZ)

The parameters in the calling sequence are:
   OPTHS  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.
   ELEMNT 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 subroutine UCSGET should be
   called with the RESET option before the new
   information is added to the element.
   ELTSIZ The number of words in ELEMNT (INTEGER*4).

The errors detected by this subroutine 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 ELEMNT.
   99(2) : This subroutine is not supported for the active device.
SECTION 2.4: ATTENTION CONTROL

This section describes subroutines which allow the programmer to control attentions from an interactive display device. It is by means of these subroutines that the programmer synchronizes the execution of the program with the actions of the console operator. Subroutine UGRATH may be used to enable attentions from a control unit (such as the light pen) on a display device. Subroutine UGRATH 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, subroutine UGRATH 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 subroutine. When an attention is reported, this code as well as other useful information is made available to the program. This information is inserted into attention data arrays which must have the following declarations:

```
INTEGER*4 AICODE
INTEGER*4 IABRAY(5)
REAL*4 XABRAY(2)
```

The following list gives (1) the valid attention codes (4 characters returned in AICODE), (2) the contents of IABRAY(1), and (3) a description of the attention and the contents (if any) of the rest of IABRAY and XABRAY.

- **NONE** 0 This is not an attention code but is a possible return value from UGRATH when no attention is available to report.
- **KBED** 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. IABRAY(2) gives the identification of the element, IABRAY(3) gives the byte offset of the item in the element that was pointed to, IABRAY(4) and IABRAY(5) give the X and Y coordinates of the item pointed to in the device coordinate system while IABRAY(1) and XABRAY(2) give the coordinates in the programmer coordinate system in effect when the attention is reported.
- **PRAM** 3 This attention is generated after a specified number of display regeneration cycles.
- **PFKH** 4 Program Function Keyboard Hake. IABRAY(2) gives the button number which caused the attention and IABRAY(3) will contain a one if the button had its light on or a zero if its light was off.
- **PFKB** 5 Program Function Keyboard Break. IABRAY(2) and IABRAY(3) are the same as for PFKH.
PFLM 6 The same as PFRM except that the attention is reported only if the button's light is on.
PFLB 7 The same as PFRB 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 subroutine UGEATH 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: SUBROUTINE UGEATH

This subroutine 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 UGEATH (OPTNS)

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

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

SECTION 2.4.2: SUBROUTINE UGDATH

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

The calling sequence is:
CALL UGDATH (OPTNS)

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

The errors detected by this subroutine are:
1(3): No device is active.
99(2): This subroutine is not supported for the active device.
SECTION 2.4.3: SUBROUTINE UGREAT

This subroutine may be used to retrieve an attention record from the attention queue of an interactive display device. When this subroutine is called and the queue is empty, there are a number of possible options for the programmer to specify. These options are (1) UGREAT 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 UGREAT(TIME, ATCODE, IARRAY, JARRAY)

The parameters in the calling sequence are:

TIME A zero or positive value indicates the wait time (in seconds) for this subroutine. On return, TIME contains the unexpired time. If TIME has a negative value the subroutine will wait indefinitely (REAL*8). Note that if TIME has a positive value when this subroutine is called, its value will be changed. Therefore, in this case, the programmer should use a variable and not a constant for TIME.

ATCODE Four characters of information which specify the type of attention (INTEGER*4).

IARRAY Attention data (INTEGER*4, DIMENSION=5).

JARRAY Attention data (REAL*8, DIMENSION=2).

The errors detected by this subroutine are:

1(3): No device is active.

99(3): This subroutine 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. Subroutine UGIPUT will put a keyboard input buffer on an interactive device and subroutine UGKGET 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: SUBROUTINE UGKPUT

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

The calling sequence is:
CALL UGKPUT(OPTNS,X,Y,IDENT,STRING,NSTR)

The parameters in the calling sequence are:
OPTNS A character string which may contain any of the following items:
  VSHEL,SHAL,LANG,VLEG Character size.
  VDIN,DMIN,BRT,VBRT 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.

X X coordinate of the center of the first character (REAL*4).
Y Y coordinate of the center of the first character (REAL*4).
IDENT The identification of the keyboard input buffer (INTEGER*4).
STRING A character string which contains the initial values for the keyboard input buffer. This may be a string of blanks.
NSTR The number of characters in STRING (INTEGER*4).

The errors detected by this subroutine are:
  1(3): No device is active.
  11(2): STRING is longer than 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 subroutine is not supported for the active device.

SECTION 2.5.2: SUBROUTINE UGKGET

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

The calling sequence is:
CALL UGKGET(IDENT,STRING,NSTR,NSTR,ICUR)
The parameters in the calling sequence are:

<table>
<thead>
<tr>
<th>IDENT</th>
<th>The identification of the keyboard input buffer to be read (INTEGER*4).</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>A character string which will have the keyboard input buffer placed in it.</td>
</tr>
<tr>
<td>NSTR</td>
<td>The maximum number of characters which can be placed in STRING (INTEGER*4).</td>
</tr>
<tr>
<td>NSTR (actual)</td>
<td>The number of characters actually placed in STRING (INTEGER*4).</td>
</tr>
<tr>
<td>ICUR</td>
<td>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 (INTEGER*4).</td>
</tr>
</tbody>
</table>

The errors detected by this subroutine 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 subroutine 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. Subroutine UGLPUT will add a light pen input buffer to the display file and subroutine 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-IDION Display Console.

SECTION 2.6.1: SUBROUTINE UGLPUT

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

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

The parameters in the calling sequence are:

| OPTBS | A character string which may contain any of the following items: |
SOLD, DASH, DBN, DBS, DOTS Line structure specification.
VDIM, VDIN, VBRT, VBUT Intensity levels.
COL1, COL2, CCI3, COL4 Color codes.

IDENT   The identification of the light pen input buffer (INTEGER*4).
LENGTH   The length (in bytes) of the light pen input buffer (INTEGER*4).

The errors detected by this subroutine 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 subroutine is not supported for the active device.

SECTION 2.6.1: SUBROUTINE UGLGET

This subroutine 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 almost the same as that required by subroutine UGBLNS.

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

The parameters in the calling sequence are:
IDENT   The identification of the light pen input buffer to be read (INTEGER*4).
NSIZE   The number of available entries in XARRAY, YARRAY, and BBITS (INTEGER*4).
XARRAY  An array where the X coordinates of the end points of the line segments will be stored (REAL*4, DIMENSION=NSIZE).
YARRAY  An array where the Y coordinates of the end points of the line segments will be stored (REAL*4, DIMENSION=NSIZE).
NPTS    The number of end points that were stored in XARRAY and YARRAY (INTEGER*4).
BBITS   An array where the blankings bits will be stored with 32 bits per word (INTEGER*4, DIMENSION=(NSIZE-31)/32).

The errors detected by this subroutine 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 subroutine is not supported for the active device.
SECTION 2.7: MISCELLANEOUS SUBROUTINES

This section contains the descriptions of subroutines which do not properly fit into any of the earlier sections.

SECTION 2.7.1: SUBROUTINE UGIDX

This subroutine is an aid in using the subroutine 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 subroutine 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, HILAB, and NLAB in subroutine UGAXIS which assures that all labeled tic marks are labeled with "round numbers".

The calling sequence is:

CALL UGIDXY (LODATA, HIDATA, NINLAB, NAHLAB, LOLAB, HILAB, NLAB)

The parameters in the calling sequence are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LODATA</td>
<td>The low extent of the data (REAL*4).</td>
</tr>
<tr>
<td>HIDATA</td>
<td>The high extent of the data (REAL*4).</td>
</tr>
<tr>
<td>NINLAB</td>
<td>The minimum acceptable number of labeled tic marks (INTEGER*4).</td>
</tr>
<tr>
<td>NAHLAB</td>
<td>The maximum acceptable number of labeled tic marks (INTEGER*4).</td>
</tr>
<tr>
<td>LOLAB</td>
<td>A computed value which will be LODATA reduced to a &quot;round number&quot; (REAL*4).</td>
</tr>
<tr>
<td>HILAB</td>
<td>A computed value which will be HIDATA increased to a &quot;round number&quot; (REAL*4).</td>
</tr>
<tr>
<td>NLAB</td>
<td>A computed value which will make all of the labels &quot;round numbers&quot; (INTEGER*4).</td>
</tr>
</tbody>
</table>

No error messages are produced by this subroutine.

Algorithms of this nature have been described in the publications listed below. The algorithms used in UGIDX are a modification of the one described in [2].

3. C. R. Lewart, Algorithms 463: Algorithms SCALE1, SCALE2,

SECTION 2.7.2: SUBROUTINE UGVVIEW

This subroutine may be used to define a projective transformation from three space into two space. The transformation may be used with subroutine UGPROJ to project a point in three dimensional space into the programmer coordinate system. The meaning of the input to this subroutine 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 SCRE and its horizontal axis is defined by the vector HDIR. Finally an eye point, E, is defined by moving a distance NYED 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 UGVVIEW (OPTNS, REFP, VDIR, HDIR, SCRD, SCRE, TRANS)

The parameters in the calling sequence are:

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

NYED=<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.

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 projective reference point (REAL*4, DIMENSION=3).

VDIR The view direction (REAL*4, DIMENSION=3).

HDIR The horizontal direction of the projection screen (REAL*4, DIMENSION=3). 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 (REAL*4).
SCRN The size of the screen (REAL*4).
TRANS An array which will be set to the projective transformation (REAL*4, DIMENSION=3). This array will contain a 3 by 4 projection matrix as well as the given projective parameters.

No error messages are produced by this subroutine.

Figure 2.7.1: The definition of a Projective Transformation.

SECTION 2.7.3: SUBROUTINE UGORTH

This subroutine may be used to define an orthogonal transformation from three space into two space. The transformation may be used with subroutine UGPROJ to project a point in three dimensional space into the programmer coordinate system. The meaning of the input to this subroutine is essentially the same as for UGVIRE. 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 REFP is not critical, it may be moved along VDIR without changing the transformation. Also the value of SCRD 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 subroutines.

The calling sequence is:
CALL UGORTH (OPTNS, REFP, VDIR, BDIR, SCRD, SCRE, TRANS)

The parameters in the calling sequence are:
OPTNS A character string which may contain any of the following items:
  ILO=<value> The X coordinate at the left hand side of the screen. The default value is 0.0.
  LHI=<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 (REAL*4, DIMENSION=3).
  VDIR The view direction (REAL*4, DIMENSION=3).
  BDIR The horizontal direction of the projection screen (REAL*4, DIMENSION=3). If this vector is given as (0.0,0.0,0.0,0.0), then a vector parallel to the X-Y plane will be supplied.
  SCRD The distance from REFP to the screen (REAL*4).
  SCRE The size of the screen (REAL*4).
  TRANS An array which will be set to the orthogonal transformation (REAL*4, DIMENSION=3). This array will contain a 3 by 4 projection matrix as well as the given orthogonal parameters.

No error messages are produced by this subroutine.

SECTION 2.7.4: SUBROUTINE UGPROJ

This subroutine uses a transformation defined by subroutine UGVIEW 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 (REAL*4, DIMENSION=3).
PT3D The coordinates of the given three dimensional point (REAL*4, DIMENSION=3).
PT2D The projected point (REAL*4, DIMENSION=2).

No error messages are produced by this subroutine.
SECTION 2.7.5: SUBROUTINE UGCTOL

This subroutine 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 almost the same as that required by subroutine UGEHNS. Before passing these line segments on to UGEHNS, 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 (OPTNS, TEXT, NTEXT, NSIZE, XARRAY, YARRAY, NPTS, EBITS)

The parameters in the calling sequence are:

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

I=<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 0.0367.

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.

NTEXT The number of characters in TEXT (INTEGER*4).

NSIZE The number of available entries in XARRAY, YARRAY, and EBITS (INTEGER*4).

XARRAY An array where the X coordinates of the end points of the line segments will be stored (REAL*4, DIMENSION=NSIZE).

YARRAY An array where the Y coordinates of the end points of the line segments will be stored (REAL*4, DIMENSION=NSIZE).

NPTS The number of end points that were stored in XARRAY and YARRAY (INTEGER*4).

EBITS An array where the blankings bits will be stored with 32 bits per word (INTEGER*4, DIMENSION=(NSIZE+31)/32).

The errors detected by this subroutine are:

1(2): The string TEXT contained more than 255 characters.

2(2): There is not enough room in XARRAY, YARRAY, and EBITS for the line segment data.

SECTION 2.7.6: SUBROUTINE UGEDIIV

This subroutine may be used to divide a previously generated element into two smaller elements. The primary purpose of this subroutine is to divide elements that have been retrieved by
subroutine UGENET from a partitioned data set. Such an element may be too large for display on a given display device. This subroutine 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 UGENET (BLNNT1, BLNNT2)

The parameters in the calling sequence are:

BLNNT1 An array which contains the original element on input, and the first part of the divided element on output.

BLNNT2 An array which will contain the second part of the divided element on output. On input, this array must have been cleared by subroutine UGENET. In certain cases this array will have to be almost as large as the graphic element in BLNNT1 although it usually need only be about half as large.

The errors detected by this subroutine 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/LEADER, and the other contains modules which must be made available at execution time.

The first of these program libraries is **UGL.CG.RCB.UGFLIB.** In this library are all of the routines that have been described in addition to some second level routines that the programmer does not normally need. If the LINK-EDITOR is used, this dataset should be concatenated with the SYSLIB data sets in the LINK step. If the LEADER 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 **UGL.CG.RCB.UGFLIB.** 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.

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 UG1000 must always be in the root segment.
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 subroutine is used; this is described in the section on the Error Processing Sub-System. Without this error processing subroutine, the array ELEMENT would have to be approximately seven times larger than its present size.

```
//uunuMSURF JOB 'uunu$pp',CLASS=B
// EXEC FORTGCLC,
//      ILREDLIB='WYL.CG.RCB.UGFLIB',
//      GOSLIB='WYL.CG.RCB.UGFLIB'
//FORT.SYSIN DD *
C SAMPLE PROGRAM: PLOTTING A MESH SURFACE
C
REAL*4  REPP(3)/50.0, 75.0, 40.0/
REAL*4  VDIR(3)/-50.0,-75.0,-30.0/
REAL*4  HDIR(3)/0.0,0.0,0.0/
REAL*4  TRANS(31)
REAL*4  ARRAY(32,52)
REAL*4  UKABEA(780)
COMMON  ELEMENT(1000)
INTEGER*4 C20(5)
REAL*4  L,Y,Z
INTEGER*4 I,J
C
C INITIALIZE THE PROGRAM: OPEN THE GRAPHIC DEVICE AND
C ASSURE THAT A MESH PLOTTING AREA IS AVAILABLE, FILL
C IN THE ARRAY WHICH DEFINES THE SURFACE, AND GENERATE
C THE PROJECTIVE TRANSFORMATION.
CALL UGOPEN('CAL/10D*','99')
CALL UGPICT('CIEH#1',0)
DO 102 I=2,52
   I=I-27
   ARRAY(I,I)=1
DO 101 J=2,32
```

Y=J-17
ARRAY(J,1)=Y
R=SQRT(I*2+J*2)
ARRAY(I,1)=(750.0/(R*2+75.0))+5.0*COS(0.4*I)

101 CONTINUE
102 CONTINUE
CALL USGINV("**",REGION,VDIR,EDIR,100.0,75.0,TRANS)

C CREATE THE PICTURE: THE GRAPHIC ELEMENT IS CLEARED,
C A BORDER IS ADDED TO THE ELEMENT, A TITLE IS ADDED,
C THE SURFACE ITSELF IS ADDED TO THE ELEMENT, AND
C FINALLY THE GRAPHIC ELEMENT IS TRANSMITTED TO THE
C GRAPHIC DEVICE.
CALL USGINT("CLEAR",ELENT,1000)
CALL USGELIN("**",0.0,0.0,0.0,ELENT)
CALL USGELIN("**",1.0,0.0,0.0,ELENT)
CALL USGELIN("**",1.0,1.0,0.0,ELENT)
CALL USGELIN("**",0.0,1.0,0.0,ELENT)
CALL USGELIN("**",0.0,0.0,0.0,ELENT)
CALL USGPTOX("HESH SURFACE EXAMPLE",
K
LL LLLL LLLL LLLL",C20,20)
CALL USGRTX("SPACING=0.04",0.2,0.8,C20,20,ELENT)
CALL USG3DMS("UPPER","**",ARRAY,32,52,TRANS,
X,UKAREA,700,ELENT)
CALL USG3DMS("LOWER,NOCONNECT","**",ARRAY,32,52,TRANS,
X,UKAREA,700,ELENT)
CALL USGPTQ("**",0,ELENT)

C TERMINATE THE PROGRAM: THE GRAPHIC DEVICE IS CLOSED
C AND THE PROGRAM RETURNS TO THE SYSTEM.
CALL USGCLOS(99)
RETURN
C
END
C ERROR PROCESSING SUBROUTINE
C
SUBROUTINE USGERR(LEVEL,NAM,INDEX)
C
INTEGER*4 LEVEL,NAM(2),INDEX, ELENT(1000)
C
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.EQ.9) GO TO 101
CALL USGPUT("**",0,ELENT)
CALL USGINT("CONTINUE","ELENT,1000
LEVEL=0
101 CONTINUE
RETURN
C
END
//LKD.SYSIN  DD *
INCLUDE SYSLIE(UGX103A)
//GO.PLOTTAPE DD DSNAME=&GLOT,DISP=(NEW,PASS),
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
//uuuLSAJP JOB 'uuu$99',CLASS=0
// EXEC PORTCG,
// LKEDLIB='WYL.CG.RCB.UGFLIB',
// GOSL='WYL.CG.RCB.UGFLIB'
//PORT.SYSIN DD *
C SAMPLE PROGRAM: INTERACTIVE LISSAJOUS FIGURES
C
EXTERNAL ERRSUB
COMMON ERRFLG
INTEGER*4 ERRPLC
REAL*8 ERR215

INTEGER*4 LPEN,'LPEN'/
INTEGER*4 ATCODE,IARRAY(5)
REAL*4 IARRAY(2)
INTEGER*4 PARRS(5)
INTEGER*4 ELEHNT(1000)
INTEGER*4 BBIT
REAL*4 A,T,X,I
INTEGER*4 M,P,HAL,HT,H,I,J

C INITIALIZE THE PROGRAM: OPEN THE GRAPHIC DEVICE, STOP
C THE DISPLAY, ENABLE THE LIGHT PEN, GENERATE THE TITLE,
C ESTABLISH INTERNAL I/O BUFFER (WITH LIBRARY SUBROUTINE
C FIO999), AND INITIALIZE THE LISSAJOUS FIGURE PARAMETERS.
CALL UGOPEN('IBH2250',99)
CALL UGPICT('OFF',0)
CALL UGEATH('LENH,OFF')
CALL UGEHNT('CLEAR',ELEHNT,1000)
CALL UGETXT('LARG',0.248,0.97,
'LISSAJOUS FIGURE GENERATOR',26,ELEHNT)
CALL UGELIN('**',0.0,0.05,0,ELEHNT)
CALL UGELIN('**',1.0,0.05,1,ELEHNT)
CALL UGEPUT('**',0,ELEHNT)
CALL FIO999(90,PARRS,5,5)

101 H=2
H=3
A=0.25
C
```
C PUT PARAMETER REQUEST ON THE SCREEN: THE DISPLAY
C CONSISTS OF A DESCRIPTION OF THE LISSAJOUS FIGURES, A
C KEYBOARD INPUT BUFFER FOR ENTERING THE PARAMETERS, AND
C A LIGHT PEN MESSAGE FOR TERMINATING THE PROGRAM.

102 CALL UGETIT('CLEAR*', ELEMENT, 1000)
   CALL UGETIT('SHAL*', 0.199, 0.7,
   X 'THE EQUATION OF A LISSAJOUS FIGURE IS:*, 38, ELEMENT)
   CALL UGETIT('SHAL*', 0.269, 0.68,
   X 'X=COS(2*PI*(H*1/N+A))', 21, ELEMENT)
   CALL UGETIT('SHAL*', 0.269, 0.66,
   X 'Y=COS(2*PI*7)*13, ELEMENT)
   CALL UGETIT('SHAL*', 0.199, 0.5,
   X 'YOU MAY NOW CHANGE THE PARAMETERS H, N, AND A*. 45,
   X ELEMENT)
   CALL UGETIT('SHAL*', 0.199, 0.48,
   X 'AND GENERATE A KEYBOARD ATTENTION; OR YOU MAY*', 45,
   X ELEMENT)
   CALL UGETIT('SHAL*', 0.199, 0.46,
   X 'LIGHT PEN THE TERMINATE MESSAGE TO QUIT.*', 40,
   X ELEMENT)
   CALL UGETPUT('**', 10, ELEMENT)
   WRITE(90,109) H,N,A
   CALL UGETPUT('SHAL*', 0.371, 0.3, 11, PARNS, 20)
   CALL UGETIT('CLEAR*', ELEMENT, 1000)
   CALL UGETIT('SHAL*', 0.7, 0.025, 'TERMINATE*', 9, ELEMENT)
   CALL UGETPUT('CH, DETC*', 1, ELEMENT)

C PROCESS OPERATOR ACTION: THE KEYBOARD IS ENABLED AND
C THE PROGRAM WAITS FOR AN ATTENTION. WHEN AN ATTENTION
C IS RECEIVED, THE KEYBOARD IS DISABLED. A LIGHT PEN
C ATTENTION CAUSES THE PROGRAM TO TERMINATE. A KEYBOARD
C ATTENTION CAUSES THE INPUT BUFFER TO BE READ, THE
C PICTURE TO BE CLEARED, AND THE INPUT BUFFER TO BE
C CONVERTED TO NUMERICIS AND CHECKED. THE EXTENDED ERROR
C HANDLER IS USED TO CHECK FOR CONVERSION ERRORS.

   CALL UGEATH('KRED,OFF**')
   CALL UGEATH(-1.0, ATCODE, IABRAV, XARRAY)
   CALL UDATH('KRED**')
   IF (ATCODE.EQ.LPEN) GO TO 108
   CALL UGGET(11, PARNS, 20, I,J)
   CALL UGPIC('CLEAR***', 1)
   CALL UGPIC('CLEAR***', 11)
   CALL UGPIC('CLEAR***', 10)
   CALL ERRSAV(215, ERR215)
   CALL ERRSET(215, 256, -1, 1, ERRSUB)
   ERRFLG=0
   READ(90,109) H,N,A
   CALL ERSER(215, ERR215)
   IF (ERRFLG.EQ.0) GO TO 101
   IF (($HL.EQ.0) OR ($HL.EQ.0)) GO TO 101

C CREATE THE LISSAJOUS FIGURE: THE FIGURE IS GENERATED
C BY LETTING THE PARAMETER T RUN FROM ZERO TO (N DIVIDED
C BY THE GREATEST COMMON DIVISOR OF N AND M) THEN A
C HORIZONTAL AND VERTICAL AXIS AND TWO LIGHT PEN
C MESSAGES ARE ADDED TO THE PICTURE.
MT=H
C

C PROCESS OPERATOR ACTION: THE PROGRAM WAITS FOR AN
C ATTENTION. A LIGHT PEN DETECT ON THE 'DELETE AXES'
C MESSAGE CAUSES THE AXES ONLY TO BE DELETED, OTHERWISE
C THE ENTIRE PICTURE IS DELETED AND THE FIRST DISPLAY IS
C RESTORED.

C 106 CALL UGDRAW((-1,0,ATCODE,ARRAY,ARRAY))
C CALL UGPICT('CLEAR*',2)
C CALL UGPICT('CLEAR*',11)
C IF (ARRAY(2).NE.2) GO TO 107
C CALL UGPICT('CH*',0)
C GO TO 106
C
C 107 CALL UGPICT('CLEAR*',1)
C CALL UGPICT('CLEAR*',10)
C GO TO 102
C
C C TERMINATE THE PROGRAM: THE GRAPHIC DEVICE IS CLOSED
C AND THE PROGRAM RETURNS TO THE SYSTEM.
C 108 CALL UGCLOS(99)
C RETURN
C
C 109 FORMAT(215,F10.4)
C
C END
C
C THIS SUBROUTINE IS USED BY THE EXTENDED ERROR HANDLER TO
C SIGNAL CONVERSION ERRORS TO THE MAIN PROGRAM.
Lissajous Figure Generator

The equation of a Lissajous figure is:
\[ X = \cos(2\pi t (M+N+A)) \]
\[ Y = \cos(2\pi t T) \]

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 subroutine as it applies to the device. In particular the
valid and default items in the OPTNS list for each subroutine 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:

```
   //PLOTTAPE CD DSNAME=G6PLOT,DISP=(NEW,FAS),
   //      VOLUME=SRR=PLOTAP,UNIT=T7,LABEL=(1,NL),
   //      DCB=(RECFM=U,BSKSIZE=3000,REXN=0)
```

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

The subroutines which may be used with this device are:
CALL UGREN(T(OPTNS, ELEMNT, ELTSIZ)
CALL UGEPUT(OPTNS, I, Y, ELEMNT)
CALL UGETLIN(OPTNS, I, Y, BBIT, ELEMNT)
CALL UGETIT(OPTNS, I, Y, TEXT, NTEXT, ELEMNT)
CALL UGETPTS(OPTNS, XARRAY, YARRAY, NPTS, ELEMNT)
CALL UGETLNS(OPTNS, XARRAY, YARRAY, NPTS, BBITS, MBBITS, ELEMNT)

When an element is transmitted to this device, all intensity level, blink mode, and color information is ignored. All lines are of the Sold type. Since the device has no character generator, only that text with the XSPACING or SPACING options will appear in the picture.

CALL UGAXIS(OPTNS, OPTNS, OPTNS, INDVAR, LOVAR, HIVAR,
LOLAB, HILAB, BLAB, ELEMNT)
CALL UG3DHS(OPTNS, OPTNS, ARRAY, MDIM, NDIM, TRANS,
UKAREA, LDIM, ELEMNT)

CALL UGOPEN(OPTNS, IDENT)
The valid items in OPTNS are:
CAL/10D  Specifies the 10 inch plotter.
CAL/29D  Specifies the 29 inch plotter.
DDNAME=<value> The ddname of the output tape. The
default value is PLATTAPE.
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 UGSLET(IDENT)
CALL UGSCAL(OPTNS, PROGLN, DEVCLN)
The default values for the programmer and device limits for the
10 inch plotter are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 1000 \\
0 & 1000 \\
\end{bmatrix}
\]

Those for the 29 inch plotter are:

\[
\begin{bmatrix}
0.0 & 1.0 \\
0.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
0 & 5800 \\
0 & 5800 \\
\end{bmatrix}
\]

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(OPTNS, IDENT, ELEMNT)
For this device, the contents of both OPTNS and IDENT are
ignored.

CALL UGPICT(OPTNS, 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.

CALL UGIDX(Y(LCDATA,HICDATA,HRILAB,HIRES,LOLAB,HIRES,LRILAB,HILM)
CALL UGVIEW(OPTES,REPP,VDIR,EDIR,SCED,SCRE,TRANS)
CALL UGORTH(OPTES,REPP,VDIR,EDIR,SCED,SCRE,TRANS)
CALL UGPROJ(TRANS,PT3D,PT2D)
CALL UGCTOL(OPTES,TEXT,TEXT,ENSIZE,ARRAY,ARRAY,NPTS,BBPTS)
CALL UGDIV(BLEN1,BLEN2)

CALL UGTOPI(CPP,CPB,EC,NSTR)
CALL UGTOP(CPP,CPB,EC,NSTR)

CALL UGEOO1(LEVEL,DNUM,INDEX)
CALL UGEOO2(OPTES,INDATA,OUTDATA)
CALL UGE203(NAME,COUNT,DNUM)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGCALD02 and their description is:

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

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

The CALCCMP 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>SMALL</td>
<td>0.000916</td>
<td>0.01500</td>
<td>109</td>
<td>66</td>
</tr>
</tbody>
</table>

When the FULSCR option is used in procedure UGOPEN, then 132 characters will fit on a single line. The LASTSCR 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:

```plaintext
//PLOTDATA DD DSNAME=OPTNS,DISP=(NEW,PASS),
  VOLUME=SH=00000,UNIT=T=1600, LABEL=(1,3L),
  DCB=(RECFM=F,LRECL=1500,BLKSIZE=1500,FORM=3)
```

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 subroutines which may be used with this device are:

- CALL UGHINT(OPTNS,ELEMT,ELTSIZ)
- CALL UGHPRINT(OPTNS,X,Y,ELEMT)
- CALL UGELIN(OPTNS,X,Y,BBIT,ELEMT)
- CALL UGEXTX(OPTNS,X,Y,TEXT,NTEXT,ELEMT)
- CALL UGEPLOT(OPTNS,XARRAY,YARRAY,NPTS,ELEMT)
- CALL UGELNS(OPTNS,XARRAY,YARRAY,NPTS,BBIT,MBBIT,ELEMT)

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

- CALL UGAXIS(OPTNS,LOPTNS,COPTNS,INDVAR,LOVAR,HVAR,
  LQUAL,HQUAL,HQUAL,ELEMT)
- CALL UGTEXT(OPTNS,LOPTNS,XARRAY,HDIM,HDIM,TRANS,
  LQUAL,ELEMT)
- CALL UGDATA(OPTNS,STRG,NSTP,ELEMT)
- CALL UGOPEN(OPTNS,ID Eng)

The valid items in OPTNS are:

- CAL/65M specifies 16mm Unasprocketed Film.
- CAL/PCF specifies Microfiche Film.

**DDNAME=**<value> The ddname of the output data set. The
default value is PLOTDATA.

**FULSCR, LARGSCR** 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.
**LARGSCR** means that the largest possible square picture
will be produced. Finally, **FULSCR** and **LARGSCR** together
give the maximum rectangular plotting area on the file.
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 1-8 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 UGCLOS(IDENT)
CALL UGSLCT(IDENT)
CALL UGSCAL(OPTNS,PROGLN,DEVCLN)

The default values for the programmer and device limits for 16mm unsprocketed film are:

| 0.0   | 1.0   | 4260 | 12123 |
| 0.0   | 1.0   | 4260 | 12123 |

When PULSCR is used, the values are:

| -0.10429 | 1.10429 | 3440 | 12943 |
| 0.0       | 1.0     | 4260 | 12123 |

When LARGSCB is used, the values are:

| 0.0   | 1.0   | 2785 | 13598 |
| 0.0   | 1.0   | 2785 | 13598 |

When PULSCR and LARGSCB are used, the values are:

| 0.0   | 1.0   | 2785 | 13598 |
| -0.09868 | 1.09868 | 1638 | 14561 |

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

| 0.0   | 1.0   | 4260 | 12123 |
| 0.0   | 1.0   | 4260 | 12123 |

When PULSCR is used, the values are:

| -0.10429 | 1.10429 | 3440 | 12943 |
| 0.0       | 1.0     | 4260 | 12123 |

CALL UGINFO(STRING,IDNT,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(OPTNS,IDNT,ELEMENT)
For this device, the contents of both OPTNS and IDNT are ignored.

CALL UGSPICT(OPTNS,IDNT)
The only valid item in options is CLEAR. The contents of IDNT is ignored and assumed to be zero. This clearing operation causes the film to advance.

CALL UGDXDY(LCDATA,HICATA,HILAB,HAILAB,LOLAB,HILAB,HLAB)
CALL UGVIEW(OPTNS,REPP,VDIR,HDIR,SCRD,SCFE,TRANS)
CALL UGOETH(OPTNS,BEPP,VDIR,HDIR,SCRD,SCRE,TRANS)
CALL UGPROJ(TRANS,PT3D,PT2D)
CALL UGCTL0L(OPTNS,TEXT,MTXT,NSIZE,XARRAY,YARRAY,HTPS,RRITS)
CALL UGDIV(ELN1T,ELN1T2)

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

CALL UGK001(LEVEL,DYNAVE,INDEX)
CALL UGK002(OPTNS,INDATA,EXDATA)
CALL UGK203(NAME,COUNT,DYNAVE)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGCALM02 and their description is:

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

SECTION A.3: WYLBUR DISPLAY FILES FOR THE TEKTRONIX 4013

The TEKTRONIX 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 non-interactive way in conjunction with the SLAC/WYLBUR Text Editing System.

To use the TEKTRONIX 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 TEKTRONIX 4013 terminal and issue a 'USE' command to obtain a member of the data set as the active file. The WYLBUR command 'LIST UNNUMBERED 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>SHAL</td>
<td>0.01797</td>
<td>0.02824</td>
<td>56</td>
<td>35</td>
</tr>
</tbody>
</table>

When the PULSCRB option is used in subroutine 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:

```
//WYLPICT DD DSN=HWP=WILBUR.DISPLAY,DISP=(NEW,KEEP),
//
// VOLUME=SER=WILSO1,UNIT=2314,SPACE=(THK,(5,1,3)),
//
// DCB=(RECFM=PB,LECSL=80,BLKSIZ=1600)
```

The ddbname, WYLPICT in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEN. The BLKSIZ 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 pictures 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 idle characters in the second card for the screen to fully clear.

The subroutines which may be used with this device are:

```
CALL UGEINT(OPTNS,ELENT,ELTSIZ)
CALL UGEPNT(OPTNS,X,Y,ELENT)
CALL UGELIN(OPTNS,X,Y,EEDIT,ELENT)
CALL UGETTY(OPTNS,X,Y,EDIT,EEDIT,ELENT)
CALL UGETPS(OPTNS,TARRAY,TARRAY,NPTS,ELENT)
CALL UGELNS(OPTNS,TARRAY,TARRAY,NPTS,BBITS,ELENT)
```

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 DIMH or VDIM 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 VHSL, SMAL, LANG, or VLRA options will be produced by the character generator.

```
CALL UGAXIS(OPTNS,LOPTHS,CPTHNS,INDVAR,LOVAR,HIVAR,
LLOL,HIAB,ELAB,ELENT)
CALL UG3DMS(OPTNS,LOPTNS,ARRAY,HDIM,HDIM,TRANS,
HARRAY,LDIM,ELENT)
```

```
CALL UGOPEN(OPTNS,IDENT)
```

The valid items in OPTNS are:

```
WYL4013 Specifies that WILBUR Display Files for the
TEKTRONIX 4013 are to be produced.
DDNAME=<value> The ddbname of the partitioned data set.
   The default value is WYLPICT.
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.
PICTSO=<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.
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.
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. UGPICT may be used to
supply an alias for the name of a member.

CALL UGCLOS(IDENT)
CALL UGSLET(IDENT)
CALL UGSCALE(OPTNS,PROGLN,DEVCLB)
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 FULSCL is used are:

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

CALL UGINFORM徼ING,IDENT,ARRAY)
The array ARRAY will contain the following:

ARRAY(1) The number of images 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(OPTNS,IDENT,ELEMENT)
For this device, the contents of both OPTNS and IDENT are
ignored.
CALL UGPICT(OPTNS,IDENT)
The only valid items in OPTNS 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 UGDIDY(LODATA,HICATA,HILAB,HAILAB,LOLAB,HILAB)
CALL UGVVIEW(OPTNS,REPF,VDIR,HDIM,SCRD,SCRE,TRANS)
CALL UGORTH(OPTNS,REPF,VDIR,HDIM,SCRD,SCRE,TRANS)
CALL UGPROJ(TRANS,PT3D,PT2D)
CALL UGCTOL(OPTNS,TEXT,NTXT,NSIZE,XARRAY,YARRAY,NPTS,BBITS)
CALL UGHDIV(ELHE1,ELHE2)

CALL UGPTX(CPP,CPS,EC,NST)
CALL UGTOP(CPP,CPS,EC,NST)

CALL UGK001(LEVEL,DVNAME,INDEX)
CALL UGK002(OPTNS,INDATA,INDATA)
CALL UGK203(NAME,COUNT,DVNAME)

In addition to the errors described with each subroutine, there
are some other errors which may be detected. These errors are
identified by a subroutine name of UGNLX02 and their description
is:

0(4): Not enough directory blocks are available in the output
data set.
H(4): (Where H is greater than 255.) An extremely unusual error condition has been detected.

SECTION 3.4: 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 UGETOPEN to obtain the graphic elements. Then 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 DSNAME=VYL0D99.NAM.ASGR,DISP=(NEW,KEEP).
//    VOLUME=SER=VYL001.UNIT=2314,SPACE=(TRK,(5,1,3)).
//    DCB=(RECFM=U,BLKSIZE=4000)
```

The data name, PICTURES in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEN. The BLESIZE 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 BLKSIZE value is in bytes while graphic element length is declared in words.

The subroutines which may be used with this device are:

```
CALL UGEXD(ODTS,ELEMD,ELTSIG)
CALL UGETPT(ODTS,1,Y,ELEMD)
CALL UGETLIN(ODTS,1,Y,FBIT,ELEMD)
CALL UGETFIT(ODTS,1,Y,TXT,TXT,ELEMD)
CALL UGETPTS(ODTS,ARRAY,ARRAY,BITS,ELEMD)
CALL UGETEMS(ODTS,ARRAY,ARRAY,NTPS,BITS,EBITS,ELEMD)
```

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

CALL UGAXIS(OPTNS,LOPTUS,COPTUS,INDVAR,LOVAR,HIVAR,
LOLAB,HILAB,HLAB,ELENT)
CALL UG3DMS(OPTNS,LOPTUS,ARRAY,NDIM,NDIM,TRANS,
UKAREA,LDIM,ELENT)

CALL UGOPEN(OPTNS,IDENT)

The valid items in OPTNS 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.

PICTSO=<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. UGPICT may be used to supply an alias for the name of a member.

CALL UGCLOS(IDENT)
CALL UGSLCT(IDENT)
CALL UGSCAL(OPTNS,PROGIN,DEVCLH)

The default values for the programmer and device limits are:

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

CALL UGIINFO(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(OPTNS,IDENT,ELENT)

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

CALL UGPICT(OPTNS,IDENT)

The only valid items in OPTNS 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 UGEGET(OPTNS,ELENT,ELTSIZE)

CALL UGBXBY(LCDATA,HIDATA,HINDEX,LINDEX,LINDEX,LINDEX,LINDEX,LINDEX,LINDEX)
CALL UGVIEWS(OPTNS,DEPP,VOIR,HDIM,SCRD,SCSZ,TRANS)
CALL UGORTU(OPTES, REPF, VDIR, HDIR, SCR0, SCE2, TRANS)
CALL UGPROJ(TRANS, PT3D, PT2D)
CALL UGCST01(OPTES, TEXT, NTEXT, NSIZE, XARRAY, YARRAY, NPTS, RBIT0)
CALL UGEDIV(ELMT1, ELMT2)

CALL UGPT03(CFP, CPS, IC, NSER)
CALL UGK001(CFP, CPS, IC, NSER)

CALL UGKD001(LEVEL, DUNABE, INDEX)
CALL UGKD002(OPTES, INDATA, RDATA)
CALL UGKD003(NAME, COUNT, DUNABT)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGDPS02 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 subroutine as it applies to the device. In particular the valid and default items in the OPTINS list for each subroutine 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 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:

```
//SCOPE50 DD UNIT=OD1
```

The ddbname, SCOPE50 in the example, must correspond to the value given in the DDNAME item in the options list of UGOPEN. The device address, OD1 in the example, is the address of the display console assigned to the "O" 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 subroutines which may be used with this device are:

CALL UGEXIT(OPTNS,ELEMT,ELTSIZ)
CALL UGEXIT(OPTNS,OPTNS,ELEMT)
CALL UGEXIT(OPTNS,OPTNS,EDT,ELEMT)
CALL UGEXIT(OPTNS,OPTNS,INPUT,ELEMT)
CALL UGEXIT(OPTNS,XARRAY,XARRAY,PTS,ELEMT)
CALL UGEXIT(OPTNS,XARRAY,XARRAY,PTS,EDT,ELEMT)

When an element is transmitted to an IBM 2250, all intensity level, wink mode, and color information is ignored. All lines are of the SOD type. Horizontal text with the VSML or NML options will be produced in the small size and horizontal text with the LANG or VLBG option will be produced in the large size.

CALL UGAXIXS(OPTNS,LOPTNS,COPYSTNS,INDVAR,LOCVAR,HICVAR,
LOLAB,HILAB,HIAB,ELEMT)
CALL UG3DIN(SOPTNS,LOPTNS,ARRAY,NDIM,NDIM,TRANS,
NEARH,LDIM,ELEMT)

CALL UGOOPEN(OPTNS,IDEN)

The valid items in OPTNS are:

IBMS2250  Specifies the IBM 2250.

MAXIMUM=<value>  The maximum number of elements which may
be present in a single picture. The default value is
25.

MAXBYTES=<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.

MADFIL={value} The maximum number of bytes that may be in a complete display file (in the IBM 2840) at one time. The default value is 4096.

DDNAME={value} The ddbname of the display unit. The default value is SCOPE50.

CALL UGCLOS(IDENT)
CALL UGSCLS(IDENT)
CALL UGSCLC(OPFRS,PROCLN,DEVNLH)

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

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

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) 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 subroutine 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 subroutines do not modify this value.

CALL UGEPUT(OPFRS,IDENT,ELEMENT)
The items in OPFRS which will be recognized and processed are DETC, OMIT, ON, and CPP.

CALL UGETCT(OPFRS,IDENT)
The items in OPFRS which will be recognized and processed are CLEAR, DETC, END, OMIT, INCL, ON and OFF. Elements may be fully manipulated.

CALL UGETRL(OPFRS,IARRAY,YARRAY)
The only item in OPFRS which will be recognized is DEEP.

CALL UGERTS(OPFRS)
OPFRS may contain OFF, KBDE, LFEN, or FRAH. The number of display regeneration cycles after a frame interrupt is fixed at one.

CALL UGDATN(OPFRS)
CALL UGRACT(TEMP,ATCODE,IARRAY,YARRAY)

CALL UGKPUT(OPFRS,X,Y,IDENT,STRING,STB)
Character size VSML and SMALL result in small sized characters while LARG and VLRG give large sized characters. All other items in OPTIONS are ignored. The input buffer should consist of an even number of characters.

CALL UGKGET(IDENT,STRING,STB,STB,ICUB)

CALL UGBKDE(YLONDATA,HICATA,MIRLAB,MAILAB,LOLAB,HLAB,WLAB)
CALL UGVIEW(OPFRS,DEEP,VDIR,HDIR,SCRD,SCRE,TRANS)
CALL UGTERM (OPTES, REFP, VDIR, HDIR, SCR0, SCRH, TRANS)
CALL UGPROJ (TRANS, PT3D, PT2D)
CALL UGCTOL (OPTES, TEXT, TEXT, NSIZE, XARRAY, YARRAY, NPTS, EDITS)
CALL UGDXIV (SLHNT1, SLHNT2)

CALL UGTOX (CFP, CPS, EC, NSER)
CALL UGTOF (CFP, CPS, EC, NSER)

CALL UGIOO1 (LEVEL, DVNAME, INDEX)
CALL UGIOO2 (OPTES, INDATA, EDATA)
CALL UGIOO3 (NAME, COURT, DVNAME)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGIS5002 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-IDION DISPLAY CONSOLE

The GIF-IDION 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 IDION 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-IDION 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 IDION 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 halt button on the IDION console twice to signal the program of the change.

The GIF-IDION 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.

<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>VSHL</td>
<td>0.00781</td>
<td>0.01172</td>
<td>128</td>
<td>85</td>
</tr>
<tr>
<td>SHAL</td>
<td>0.01367</td>
<td>0.02051</td>
<td>73</td>
<td>48</td>
</tr>
<tr>
<td>LARG</td>
<td>0.02051</td>
<td>0.02930</td>
<td>48</td>
<td>34</td>
</tr>
<tr>
<td>VLKG</td>
<td>0.02734</td>
<td>0.04102</td>
<td>36</td>
<td>24</td>
</tr>
</tbody>
</table>

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 DDNAME item in the options list of UGOOPEN. 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)*N**
- Move the cursor to the next keyboard input buffer.

**(CONTROL)*R**
- 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 16H 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 use 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 FRAME attention should be enabled and the regeneration count set to a small value (usually 2, 3, or 4) with the FRAME option of subroutine 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 PHAN 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 PHAN attention should be enabled and the regeneration count should be set.
B. The shutter should be opened with the SHOPEN option in subroutine UGCTRL.
C. The elements for a single display file should be transmitted to the GIF-IDIION and the display should be started.
D. The occurrence of the PHAN 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 SENCLOS option in subroutine 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-IDIION 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 Varian 620/I 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 RED/SIX option in subroutine UGETPUT, 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 subroutines which may be used with this device are:

CALL UGETINT(OPTNS,ELEMENT,ELTSIX)
CALL UGETPUT(OPTNS,X,Y,ELEMENT)
CALL UGELSIX(OPTNS,X,Y,EBIT,ELEMENT)
CALL UGETINT(OPTNS,X,Y,EBIT,WBIT,ELEMENT)
CALL UGETPUT(OPTNS,XARRAY,YARRAY,BITS,ELEMENT)
CALL UGELSIX(OPTNS,XARRAY,YARRAY,BITS,BBITE,BBIT,ELEMENT)

The IDIION will correctly process the intensity levels VDIN, DIMN, BRIT, VBRT; the wink mode specifications WINK and STDY; and the line structure specifications SOLD, DASH, DDSN, and DOIS. Color information is ignored. Character sizes of VSHL, SNAL, LNG, 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 IDIION that we have at SLAC.

CALL UGAXIS(OPTNS,LOPTNS,COPTNS,INDVAR,LOVAR,HIVAR,
LONB,WILAB,HIB,BIT,ELEMENT)
CALL UGDATA(OPTNS,LOPTNS,ARRAY,HDIM,NDIM,TRANS,
UKRAME,EBIT,ELEMENT)
CALL UGEXINFO(OPTNS,X,Y,IDENT,ELEMENT)

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

CALL UGDATA(OPTNS,STRING,ESTB,ELEMENT)

CALL UGOPEN(OPTNS,IDENT)

The valid items in OPTNS are:
GIN/IDI Specifies the GIN-IDIION.
MAXNUM=<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.
DDNAME=<value> The dname of the display unit. The default value is GIPLINK.

CALL USCLOSE(IDENT)
CALL USCLCT(IDENT)
CALL USSCSLAP(OPTNS,PROGLM,DEVCLB)

The default values for the programmer and device limits for this device are:
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 subroutine UGEPUT transmits an element to the display device, this location is set to the length of the element after it has been transmitted to device orders. Other subroutines do not modify this value.

CALL UGEPUT(OPTNS, IDENT, ELEMT)
The elements in OPTNS which will be recognized and processed are DETC, ONIT, ON, OFF, NCHP, DETC, NSUB, and MINSIZE. NCHP 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 pen element.

CALL UGPICT(OPTNS, IDENT)
The elements in OPTNS which will be recognized and processed are CLEAR, DETC, MDEL, ONIT, INCL, ON and OFF. Elements may be fully manipulated.

CALL UGCTRL(OPTNS, IARRAY, JARRAY)
The valid items in OPTNS are:
FRAME The regeneration count for the FRAME attention may be any positive value.
LITES=<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, TPGFT, TPRNV The tracking pattern may be put on the screen, its position may be obtained, and it may be removed from the screen.
TPIC, TPDBN, TPRSR The IDIION may be put into the inactive, draw, or erase mode with respect to light pen input buffers.
ESON, ESCFT, ESTRC 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 UGBATN(OPTNS)
OPTNS may contain OFF, KBRC, LPEN, FRAM, PFKN, PFKB, PFKN, PFLH, LPCH, or LPSC.

CALL UGBATN(OPTNS)

CALL UGPUT(OPTNS, X, Y, IDENT, STRING, NSTB)
The character sizes VSML, SMAL, LARL, and VLRG; and the
intensity levels VEIR, DINH, BRI1, and VBDT will be correctly processed. The input buffer should consist of an even number of characters.

CALL UGGET(IDENT, STRING, NSIZE, NSTR, ICUN)

CALL UGPUT(OPTNS, IDENT, LENGTH)

The intensity levels VEIR, DINH, BRI1, and VBDT; and the line structure specifications SOLD, DASH, DDSH, and DOTS will be correctly processed.

CALL UGGET(IDC0, IDENT, NSIZE, XARRAY, YARRAY, NPTS, BBITS)

CALL UGEDIT(LODATA, HICDATA, MINLAB, MAXLAB, LBLAB, HBLAB, NLAB)

CALL UGKTSN(OPTNS, REFF, VEIR, HDIR, SCRD, SCRE, TRANS)

CALL UGKTOJ(TRANS, PT30, PT20)

CALL UGCTOL(OPTNS, TYP, WHT, HSIZE, XARRAY, YARRAY, NPTS, BBITS)

CALL UGEDITV(ELNHT1, ELNHT2)

CALL UGPT0X(CFP, CPS, EC, NSTR)

CALL UGPT0P(CFP, CPS, EC, NSTR)

CALL UGK007(LEVEL, DBHANE, INDEX)

CALL UGK005(OPTNS, INDATA, XDATA)

CALL UGK203(WAHY, COUNT, DBHANE)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGGIP102 and their description is:

1(N): The light pen input buffer in the Varian 620/1 is full.

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

SECTION B.3: THE TERTRONIX 4013 DISPLAY TERMINAL

The TERTRONIX 4013 is an interactive storage display terminal with an eight inch wide by six inch high screen and a keyboard. The CRT on the TERTRONIX 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 TERTRONIX 4013 terminals are under the control of the MILTEN Terminal Manager. A program which sends information to and receives information from a terminal must communicate with MILTEN. To run an interactive program using the TERTRONIX 4013 the user should:

1. Sign on the terminal and begin using the WYLBUG 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 MILTEN. 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 HILTON, you may type its name to connect your terminal to your sub-system. However before this connection is made, HILTON 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 UYLEURE by typing $WYL and hitting the RETURN key. When you are finished using UYLEURE 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>0.02024</td>
<td>56</td>
<td>35</td>
</tr>
</tbody>
</table>

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

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

```
//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 subroutines which may be used with this device are:
CALL TGODINT(OPTNS, ELEMNT, ELTSIZ)
CALL TGODEMT(OPTNS, I, Y, ELEMNT)
CALL TGODELT(OPTNS, X, Y, BBIT, ELEMNT)
CALL TGODNIT(OPTNS, X, Y, TEN, BBIT, ELEMNT)
CALL TGODEETS(OPTNS, XARRAY, YARRAY, ENTS, ELEMNT)
CALL TGODELNS(OPTNS, XARRAY, YARRAY, BBITS, ELEMNT)

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

CALL TGODAXIS(OPTNS, LOFTNS, OPTNS, XVAR, LOVAR, LIVAR,
LOLAB, NILAB, NILAB, ELEMNT)
CALL TGODEBLS(OPTNS, LOFTNS, ARRAY, NDIM, NDIM, TRANS,
DARRAY, LDIM, ELEMNT)

CALL TGODEPEN(OPTNS, IDENT)
The valid items in OPTNS are:

TERMINAL Specify the TERMINAL 4013.
MAGICICED=<value> The Magic Word that NILTEM will request.
The Magic Word may have from one to eight characters
and have a default value of XIXIIIX.
DDNAME=<value> The dname of the display unit when
IDSCOPF 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 TGODEPEN is called, your program will identify itself to
NILTEM as a sub-system and wait until you have made the
connection between your sub-system and your terminal.

CALL TGODECLOS(IDENT)
CALL TGODELCNT(IDENT)
CALL TGODESCLL(OPTNS, PROCLN, DEVCLN)
The default values for the programmer and device limits are:

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

The default values when FULSCR is used are:

\[
\begin{bmatrix}
-0.1566 & 1.1566 \\
0.0 & 1.0 \\
\end{bmatrix} \quad \begin{bmatrix}
0 & 1023 \\
0 & 779 \\
\end{bmatrix}
\]

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

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

CALL UGCTRL(OPTNS,ARRAY,ARRAY)

The only valid items in OPTNS are BEEP and TPCTL. TPCTL 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 subroutine.

CALL UGENTH(OPTNS)
The only item in OPTNS which will be recognized is KBED.

CALL UGDAETH(OPTNS)

CALL UGFPUT(OPTNS,Y,IDENT,STRING,NSTB)
OPTNS 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 UGGET(IDENT,STRING,NSTB,NSTB,ICUB)
The characters that are read are exactly those characters that the console operator has typed.

CALL UGDXYDY(LCDATA,HDATA,HDNLAB,HDNLAB,LOLAB,LOLAB,NLAB)

CALL UGVIEW(OPTNS,BEFP,VDIR,HDIR,SCRD,SCRE,TRANS)

CALL UGORTH(OPTNS,BEFP,VDIR,HDIR,SCRD,SCRE,TRANS)

CALL UGPROM(TRANS,PT3D,PT2D)

CALL UGCTRL(OPTNS,TEXT,TEXT,NSIZE,ARRAY,ARRAY,MPTS,BBITS)

CALL UGDEIV(ELNT1,ELNT2)

CALL UGPTOH(CFF,CPS,EC,NSTB)

CALL UGTOP(CFF,CPS,EC,NSTB)

CALL UGEOO1(LEVEL,DVNAME,INDEX)

CALL UGEOO2(OPTNS,INDATA,EXDATA)

CALL UGEOO3(DVNAME,COUNT,DVNAME)

In addition to the errors described with each subroutine, there are some other errors which may be detected. These errors are identified by a subroutine name of UGTR1302 and their description is:

N(4): (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 in 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 XSPACING 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 2990 keypunch in addition to the ten plotting symbols. Also the EBCDIC 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 UGK103A

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(UGK103A)
```

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 XSPACING option in UGETEIT and their height is the same as that value). This alternate module generates characters of differing widths and heights, thus the upper case letter "H" 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 subroutines UGPTOX and UGITOP.

<table>
<thead>
<tr>
<th>Hex. Code</th>
<th>Character Description</th>
<th>Hex. Code</th>
<th>Character Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 aS</td>
<td>Null</td>
<td>06 qG</td>
<td>L.C. Theta</td>
</tr>
<tr>
<td>01 AG</td>
<td>L.C. Alpha</td>
<td>09 qG</td>
<td>L.C. Iota</td>
</tr>
<tr>
<td>02 BG</td>
<td>L.C. Beta</td>
<td>0a IS</td>
<td>Integral Sign</td>
</tr>
<tr>
<td>03 GG</td>
<td>L.C. Gamma</td>
<td>0B xS</td>
<td>Times Sign</td>
</tr>
<tr>
<td>04 DG</td>
<td>L.C. Delta</td>
<td>0c :S</td>
<td>Division Sign</td>
</tr>
<tr>
<td>05 EG</td>
<td>L.C. Epsilon</td>
<td>0d ES</td>
<td>Approximately Equal</td>
</tr>
<tr>
<td>06 ZG</td>
<td>L.C. Zeta</td>
<td>0E PS</td>
<td>Partial Derivative</td>
</tr>
<tr>
<td>07 HG</td>
<td>L.C.Eta</td>
<td>0F DS</td>
<td>Del</td>
</tr>
<tr>
<td>10 ?S</td>
<td>Interbang</td>
<td>18 RG</td>
<td>L.C. Rho</td>
</tr>
<tr>
<td>11 KG</td>
<td>L.C. Kappa</td>
<td>19 SG</td>
<td>L.C. Sigma</td>
</tr>
<tr>
<td>12 LG</td>
<td>L.C. Lambda</td>
<td>1a UA</td>
<td>Up Arrow</td>
</tr>
<tr>
<td>13 MG</td>
<td>L.C. Mu</td>
<td>1e DA</td>
<td>Down Arrow</td>
</tr>
<tr>
<td>14 MG</td>
<td>L.C. Mu</td>
<td>1c LA</td>
<td>Left Arrow</td>
</tr>
<tr>
<td>15 XG</td>
<td>L.C. Xi</td>
<td>1d BA</td>
<td>Right Arrow</td>
</tr>
<tr>
<td>16 OG</td>
<td>L.C. Omicron</td>
<td>1f BA</td>
<td>Left/Right Arrow</td>
</tr>
<tr>
<td>17 PG</td>
<td>L.C. Pi</td>
<td>1f Unassigned</td>
<td></td>
</tr>
<tr>
<td>20 Mx</td>
<td>Membership Symbol</td>
<td>28 yg</td>
<td>L.C. Psi</td>
</tr>
<tr>
<td>21 Mx</td>
<td>Membership Negation</td>
<td>29 wg</td>
<td>L.C. Omega</td>
</tr>
<tr>
<td>22 Ex</td>
<td>Existential Quant.</td>
<td>2a IX</td>
<td>Intersection</td>
</tr>
<tr>
<td>23 AX</td>
<td>Universal Quant.</td>
<td>2b UX</td>
<td>Union</td>
</tr>
<tr>
<td>24 TG</td>
<td>L.C. Tau</td>
<td>2c &lt;x</td>
<td>Contained in</td>
</tr>
<tr>
<td>25 UG</td>
<td>L.C. Upsilon</td>
<td>2d &gt;x</td>
<td>Contains</td>
</tr>
<tr>
<td>26 PG</td>
<td>L.C. Phi</td>
<td>2e LX</td>
<td>Contained in/Equal</td>
</tr>
<tr>
<td>27 CG</td>
<td>L.C. Chi</td>
<td>2f RX</td>
<td>Contains/Equal</td>
</tr>
<tr>
<td>30 0C</td>
<td>Enter Superscr Mode</td>
<td>38 8C</td>
<td>Save Position-3</td>
</tr>
<tr>
<td>31 1C</td>
<td>Leave Superscr Mode</td>
<td>39 9C</td>
<td>Restore Pos-3</td>
</tr>
<tr>
<td>32 2C</td>
<td>Enter Subscr Mode</td>
<td>3a Unassigned</td>
<td></td>
</tr>
<tr>
<td>33 3C</td>
<td>Leave Subscr Mode</td>
<td>3b Unassigned</td>
<td></td>
</tr>
<tr>
<td>34 4C</td>
<td>Save Position-1</td>
<td>3c Unassigned</td>
<td></td>
</tr>
<tr>
<td>35 5C</td>
<td>Restore Pos-1</td>
<td>3d Unassigned</td>
<td></td>
</tr>
<tr>
<td>36 6C</td>
<td>Save Position-2</td>
<td>3e Ec</td>
<td>Increase Size</td>
</tr>
<tr>
<td>37 7C</td>
<td>Restore Pos-2</td>
<td>3f Pf</td>
<td>Decrease Size</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>40</td>
<td>\u200b</td>
<td>Blank</td>
<td>48</td>
</tr>
<tr>
<td>41</td>
<td>AH</td>
<td>U.C. Alpha</td>
<td>49</td>
</tr>
<tr>
<td>42</td>
<td>BH</td>
<td>U.C. Beta</td>
<td>4A</td>
</tr>
<tr>
<td>43</td>
<td>CH</td>
<td>U.C. Gamma</td>
<td>4B</td>
</tr>
<tr>
<td>44</td>
<td>DH</td>
<td>U.C. Delta</td>
<td>4C</td>
</tr>
<tr>
<td>45</td>
<td>EH</td>
<td>U.C. Epsilon</td>
<td>4D</td>
</tr>
<tr>
<td>46</td>
<td>EH</td>
<td>U.C. Zeta</td>
<td>4E</td>
</tr>
<tr>
<td>47</td>
<td>EH</td>
<td>U.C. Eta</td>
<td>4F</td>
</tr>
<tr>
<td>50</td>
<td>$n</td>
<td>Ampersand</td>
<td>58</td>
</tr>
<tr>
<td>51</td>
<td>KN</td>
<td>U.C. Kappa</td>
<td>59</td>
</tr>
<tr>
<td>52</td>
<td>LH</td>
<td>U.C. Lambda</td>
<td>5A</td>
</tr>
<tr>
<td>53</td>
<td>HH</td>
<td>U.C. Nu</td>
<td>5B</td>
</tr>
<tr>
<td>54</td>
<td>HH</td>
<td>U.C. Nu</td>
<td>5C</td>
</tr>
<tr>
<td>55</td>
<td>XH</td>
<td>U.C. Xi</td>
<td>5D</td>
</tr>
<tr>
<td>56</td>
<td>OH</td>
<td>U.C. Omicron</td>
<td>5E</td>
</tr>
<tr>
<td>57</td>
<td>PH</td>
<td>U.C. Pi</td>
<td>5F</td>
</tr>
<tr>
<td>60</td>
<td>\n</td>
<td>Minus Sign</td>
<td>68</td>
</tr>
<tr>
<td>61</td>
<td>/n</td>
<td>Slash Mark</td>
<td>69</td>
</tr>
<tr>
<td>62</td>
<td>+S</td>
<td>Group Plus</td>
<td>6A</td>
</tr>
<tr>
<td>63</td>
<td>*S</td>
<td>Group Multiply</td>
<td>6B</td>
</tr>
<tr>
<td>64</td>
<td>TH</td>
<td>U.C. Tau</td>
<td>6C</td>
</tr>
<tr>
<td>65</td>
<td>UN</td>
<td>U.C. Upsilon</td>
<td>6D</td>
</tr>
<tr>
<td>66</td>
<td>PN</td>
<td>U.C. Phi</td>
<td>6E</td>
</tr>
<tr>
<td>67</td>
<td>CN</td>
<td>U.C. Chi</td>
<td>6F</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>Unassigned</td>
<td>78</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td>Unassigned</td>
<td>79</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>Unassigned</td>
<td>7A</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td>Unassigned</td>
<td>7B</td>
</tr>
<tr>
<td>74</td>
<td></td>
<td>Unassigned</td>
<td>7C</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td>Unassigned</td>
<td>7D</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td>Unassigned</td>
<td>7E</td>
</tr>
<tr>
<td>77</td>
<td></td>
<td>Unassigned</td>
<td>7F</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>Unassigned</td>
<td>88</td>
</tr>
<tr>
<td>81</td>
<td>AL</td>
<td>L.C. A</td>
<td>89</td>
</tr>
<tr>
<td>82</td>
<td>BL</td>
<td>L.C. B</td>
<td>8A</td>
</tr>
<tr>
<td>83</td>
<td>CL</td>
<td>L.C. C</td>
<td>8B</td>
</tr>
<tr>
<td>84</td>
<td>DL</td>
<td>L.C. D</td>
<td>8C</td>
</tr>
<tr>
<td>85</td>
<td>EL</td>
<td>L.C. E</td>
<td>8D</td>
</tr>
<tr>
<td>86</td>
<td>FL</td>
<td>L.C. F</td>
<td>8E</td>
</tr>
<tr>
<td>87</td>
<td>GL</td>
<td>L.C. G</td>
<td>8F</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td>Unassigned</td>
<td>98</td>
</tr>
<tr>
<td>91</td>
<td>JL</td>
<td>L.C. J</td>
<td>99</td>
</tr>
<tr>
<td>92</td>
<td>KL</td>
<td>L.C. K</td>
<td>9A</td>
</tr>
<tr>
<td>93</td>
<td>LL</td>
<td>L.C. L</td>
<td>9B</td>
</tr>
<tr>
<td>94</td>
<td>HL</td>
<td>L.C. M</td>
<td>9C</td>
</tr>
<tr>
<td>95</td>
<td>NL</td>
<td>L.C. N</td>
<td>9D</td>
</tr>
<tr>
<td>96</td>
<td>OL</td>
<td>L.C. O</td>
<td>9E</td>
</tr>
<tr>
<td>97</td>
<td>PL</td>
<td>L.C. P</td>
<td>9F</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>AE</td>
</tr>
<tr>
<td>A4</td>
<td>WL</td>
<td>L.C. W</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>VL</td>
<td>L.C. V</td>
<td>AE</td>
</tr>
<tr>
<td>A7</td>
<td>VL</td>
<td>L.C. V</td>
<td>AP</td>
</tr>
<tr>
<td>B0</td>
<td>OP</td>
<td>Plot Sym V Cross</td>
<td>B8</td>
</tr>
<tr>
<td>B1</td>
<td>1P</td>
<td>Plot Sym D Cross</td>
<td>B9</td>
</tr>
<tr>
<td>B2</td>
<td>2P</td>
<td>Plot Sym Diamond</td>
<td>BA</td>
</tr>
<tr>
<td>B3</td>
<td>3P</td>
<td>Plot Sym Square</td>
<td>BB</td>
</tr>
<tr>
<td>B4</td>
<td>4P</td>
<td>Plot Sym P Diamond</td>
<td>BC</td>
</tr>
<tr>
<td>B5</td>
<td>5P</td>
<td>Plot Sym P Square</td>
<td>BE</td>
</tr>
<tr>
<td>B6</td>
<td>6P</td>
<td>Plot Sym P V Cross</td>
<td>BF</td>
</tr>
<tr>
<td>B7</td>
<td>7P</td>
<td>Plot Sym P D Cross</td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td></td>
<td>Unassigned</td>
<td>C8</td>
</tr>
<tr>
<td>C1</td>
<td>An</td>
<td>U.C. A</td>
<td>C9</td>
</tr>
<tr>
<td>C2</td>
<td>Bn</td>
<td>U.C. B</td>
<td>CA</td>
</tr>
<tr>
<td>C3</td>
<td>Cn</td>
<td>U.C. C</td>
<td>CB</td>
</tr>
<tr>
<td>C4</td>
<td>Dn</td>
<td>U.C. D</td>
<td>CC</td>
</tr>
<tr>
<td>C5</td>
<td>En</td>
<td>U.C. E</td>
<td>CD</td>
</tr>
<tr>
<td>C6</td>
<td>Fn</td>
<td>U.C. F</td>
<td>CE</td>
</tr>
<tr>
<td>C7</td>
<td>Gm</td>
<td>U.C. G</td>
<td>CF</td>
</tr>
<tr>
<td>D0</td>
<td>OU</td>
<td>Backwards Blank</td>
<td>D8</td>
</tr>
<tr>
<td>D1</td>
<td>Jn</td>
<td>U.C. J</td>
<td>D9</td>
</tr>
<tr>
<td>D2</td>
<td>Kn</td>
<td>U.C. K</td>
<td>DA</td>
</tr>
<tr>
<td>D3</td>
<td>Ln</td>
<td>U.C. L</td>
<td>DB</td>
</tr>
<tr>
<td>D4</td>
<td>Mn</td>
<td>U.C. M</td>
<td>DC</td>
</tr>
<tr>
<td>D5</td>
<td>Nn</td>
<td>U.C. N</td>
<td>DD</td>
</tr>
<tr>
<td>D6</td>
<td>Om</td>
<td>U.C. O</td>
<td>DE</td>
</tr>
<tr>
<td>D7</td>
<td>Pm</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>Sn</td>
<td>U.C. S</td>
<td>EA</td>
</tr>
<tr>
<td>E3</td>
<td>Tn</td>
<td>U.C. T</td>
<td>EB</td>
</tr>
<tr>
<td>E4</td>
<td>Un</td>
<td>U.C. U</td>
<td>EC</td>
</tr>
<tr>
<td>E5</td>
<td>Vn</td>
<td>U.C. V</td>
<td>ED</td>
</tr>
<tr>
<td>E6</td>
<td>Wn</td>
<td>U.C. W</td>
<td>EE</td>
</tr>
<tr>
<td>E7</td>
<td>Xn</td>
<td>U.C. X</td>
<td>EF</td>
</tr>
<tr>
<td>F0</td>
<td>0m</td>
<td>Numeral 0</td>
<td>F8</td>
</tr>
<tr>
<td>F1</td>
<td>1m</td>
<td>Numeral 1</td>
<td>F9</td>
</tr>
<tr>
<td>F2</td>
<td>2m</td>
<td>Numeral 2</td>
<td>FA</td>
</tr>
<tr>
<td>F3</td>
<td>3m</td>
<td>Numeral 3</td>
<td>FB</td>
</tr>
<tr>
<td>F4</td>
<td>4m</td>
<td>Numeral 4</td>
<td>FC</td>
</tr>
<tr>
<td>F5</td>
<td>5m</td>
<td>Numeral 5</td>
<td>FD</td>
</tr>
<tr>
<td>F6</td>
<td>6m</td>
<td>Numeral 6</td>
<td>FE</td>
</tr>
<tr>
<td>F7</td>
<td>7m</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.

<table>
<thead>
<tr>
<th>ABCDEFGHIJKLMNOPQRSTUVWXYZ</th>
<th>abcdefghijklmnopqrstuvwxyz</th>
</tr>
</thead>
<tbody>
<tr>
<td>ΑΒΓΔΕΖΗΘΙΚΑΜΝΞΟΠΡΣΤΥΦΩ</td>
<td>αβγδεζηθικαμνξοπρςτυφω</td>
</tr>
<tr>
<td>0123456789 +×÷×××0</td>
<td>0.(-</td>
</tr>
<tr>
<td>$+××××××××××××××××&lt;××××</td>
<td>{=f(x)×0[±]0}</td>
</tr>
</tbody>
</table>

Figure C.1.1: The Extended Character Set.

SECTION C.2: SUBROUTINES UGETOX 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 subroutine UGETXT and then transmitted to a display device with subroutine UGETPUT. The subroutines UGETOX 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. Subroutine UGETOX uses the primary and secondary pairs to generate the extended string and subroutine UGITOP does the inverse operation. For an example using UGETOX 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 UGETOX 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 "n" 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 "N", control functions by a "C", plotting symbols by a "P", and special characters by an "A", "S", or "X". Finally, the standard characters are all designated by a blank secondary character.

The calling sequences are:

CALL UGETOX (CFF,CPS,EC,HST)
CALL UGITOP (CFF,CPS,EC,HST)
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.
HSTR   The number of characters in the strings (INTEGER*4).

No error messages are produced by these subroutines.

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.](image1)

\[
(n-1)! = \Gamma(n) = \sum_{t=0}^{\infty} e^{-t}t^{n-1} dt
\]

![Figure C.2.2: A More Elaborate Example.](image2)

SECTION C.3: MODULE UGX103Z

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:

```
INCLUDE SYSLIB(UGX1032)
```

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 subroutine 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 subroutine 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 subroutine 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 subroutine has been called. The error indicators are declared by the statements:

```fortran
COMMON /UGERR/UGNAME(2),UGIND
INTEGER*4 UGNAME,UGIND
```

When a Minor Error or Error has occurred, UGNAME will contain the subroutine name and UGIND will contain the index. If no error has occurred, UGNAME 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 subroutine called UGIXERR. If the user includes a subroutine with this name, it will be called whenever any error is detected. The skeleton for the subroutine is:

```fortran
SUBROUTINE UGIXERR(LEVEL,NAME,INDEX)
INTEGER*4 LEVEL,NAME(2),INDEX
...
END
```

This subroutine can do almost anything to try to recover from the error, including calling other subroutines in this system. However, it is the programmer's responsibility to see that no FORTRAN subroutine is entered recursively. In particular, this means that if subroutine UGIXERR calls other subroutines in this system, these subroutines must not themselves generate errors because that would mean a recursive call on UGIXERR. If an attempt is made to call UGIXERR recursively, the message for the second error will be printed followed by a message with a subroutine name of UGIXERR whose description is:

1(4): An attempt was made to use UGIXERR recursively.

Subroutine UGIXERR 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 subroutine 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 subroutine UGEINT with the CLEAN 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. Put the graphic element in a COMMON block.
2. Prepare a UGIXE subroutine 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 UGIXE should transmit the graphic element to the CALCOMP by calling subroutine UGIPUT and then clear the graphic element by calling subroutine UGEINT with the CONTINUE option. The CONTINUE option is needed here because of the way subroutine UGELIN works. It draws from the last point to the given point. The CONTINUE option causes the last point to be saved. Subroutine UGIXE 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: SUBROUTINE UGIX001

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

The calling sequence is:

CALL UGIX001(LEVEL, DVNAME, INDEX)

The parameters in the calling sequence are:

LEVEL The level number of the error (INTEGER*2). A value of zero will cause the error flags to be cleared.
DVNAME A dope vector for the name of the subroutine identifying the error (INTEGER*4, DIMENSION=2). A dope vector may be constructed by using the subroutine named UGIX203.
INDEX The index of the error (INTEGER*2).
SECTION D.2: SUBROUTINE UG1203

The purpose of this subroutine is to provide a means of constructing the dope vectors which must be passed to UG1001. Dope vectors are passed to UG1001 because this assures compatibility with the PL/1 version of this system.

The calling sequence is:
CALL UG1203(NAME,COUNT,DVNAME)

The parameters in the calling sequence are:

NAME A character string. When a dope vector is being constructed for UG1001, this parameter should contain the name of the subroutine identifying the error.
COUNT The number of characters in name (INTEGER*4). When a dope vector is being constructed for UG1001, the parameter should have a value of eight.
DVNAME The dope vector which will be constructed (INTEGER*4, DIMENSION=2).

SECTION D.3: MODULE UG1001A

Under some circumstances, the programmer may find it desirable to relax the constraint against recursive use of UG1ERR. If the programmer creates the load module with the LINK-EDITOR and includes the card:

INCLUDE SYSLIB(UG1001A)

then an alternate error processing sub-system will be loaded which permits recursive use of UG1ERR. This means that the copy of UG1ERR in the load module must be coded in Assembler Language and must be capable of being used recursively.
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 subroutine UGSCALE. These matrices have the following form:

\[
\begin{bmatrix}
  PXLO & PXHI \\
  PYLO & PYHI \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  DXLO & DXHI \\
  DYLO & DYHI \\
\end{bmatrix}
\]

The programmer limits are given in floating point form and the device limits are in fixed point form. To scale the X coordinate, PXLO maps into DXLO, PXHI maps into DXHI, 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 CALCOMP Drum Plotter and the programmer wishes to specify the coordinates on the paper in terms of inches. The proper matrices are:

\[
\begin{bmatrix}
  0.0 & 25.0 \\
  0.0 & 10.0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  0 & 2500 \\
  0 & 1000 \\
\end{bmatrix}
\]

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 CALCOMP 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:
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:

\[
\begin{bmatrix}
0.0 & 25.0 \\
0.0 & 10.0
\end{bmatrix}
\begin{bmatrix}
0 & 4095 \\
1229 & 2867
\end{bmatrix}
\]

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-IDICON.

2. No check is made in subroutine 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 (DXI-DXLO)/(PXH-PXLO). In the vertical direction this ratio is (DYL-DYLO)/(FYH-FYLO). 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 UGX112Z

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 lead module with the LINK-EDITOR and includes the card:

`INCLUDE SYSLIB(UGI112Z)

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

SECTION F.1: SUBROUTINE UGI002

This subroutine 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 UGI002 (OPTNS, INDATA, EINDATA)

The parameters in the calling sequence are:

OPTNS  The options list which will be scanned. A special character, usually an asterisk, terminates the string.
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, commas, or the terminal character).
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. This halfword may also be used to supply an alternate string terminator. If the high order byte is zero, the default asterisk is used; if the high order byte is not zero, then this byte specifies the terminating character.
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.

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

The following example may help explain this subroutine. Suppose we wish to scan for items of the form 'CAL/10D', or 'DDNUM=<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:

```
INTEGER*2   EXDATA(5),FLAG,DDNM(4)
```

```
EQUVALENCE (FLAG,EXDATA(1)),(DDNM(1),EXDATA(2))
```

Now that the byte offsets in the output array are known, we can specify the input array as follows:

```
INTEGER*2   INDATA(16)/2,
```

```
1,7,0,5,'CA','L/','10','D ',
4,6,2,0,'DD','NA','NR'/
```

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

```
FLAG=0
DO 100 I=1,4
100 DDNM(I)=0
```

```
CALL UG3002(OPTNS,INDATA,EXDATA)
```

If OPTNS contained 'CAL/10D' then FLAG will be changed to 5; if OPTNS contained 'DDNUM=<string>' then DDNM 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 subroutines 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:

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

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

Notice that if a graphic element is to be saved in a data set, ELEMNT(I) gives the number of words which must be written. The value in ELEMNT(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 subroutine UGEINT is called with the CLEAN option, it resets the graphic element to the following:

ELEMNT(1): 3
ELEMNT(2): 4
ELEMNT(3): 0
ELEMNT(4): ELTSIZ-1

When the RESET option is used, UGEINT sets:

ELEMNT(ELEMNT(1)+1): ELTSIZ-1

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

ELEMNT(1): 1
ELEMNT(2): 4
ELEMNT(3): 4
ELEMNT(4)...ELEMNT(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 zero.
ELEMNT(I+1): ELTSIZ-1

The graphic element generators work by moving the words with the maximum length down, and inserting a block of data into the element. ELEMNT(1), and if necessary ELEMNT(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:

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

ELEMT(I+1): The intensity, color, and wink bits. Bits 4 through 7 are reserved for VBRT, BRIT, DIMH, and VDIM; bits 8 through 11 are for COL4, COL3, COL2, and COL1; bit 15 is reserved for VIMK.

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

A line data block consists of the following:

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

ELEMT(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:

ELEMT(I): The first character contains 'C'; the second contains 'I' (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.

ELEMT(I+1): The intensity, color, and wink bits.

ELEMT(I+2): The rotation angle in floating point form.

ELEMT(I+3): Either the character spacing for the stroke generator in floating point form or the character sizes VLRG, LABG, SHAL, and VSHL coded in bits 28 through 31.

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

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

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

ELEMT(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:

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

ELEMT(I+1): The intensity, color, wink, and line structure bits.

ELEMT(I+2): The rotation angle in floating point form.

ELEMT(I+3): The scaling value in floating point form.
Following this may be an arbitrary number of sub-blocks consisting of the following:

**ELEMNT(I):** X coordinate of the relative origin of the element-subroutine in floating point form.

**ELEMNT(I+1):** Y coordinate of the relative origin of the element-subroutine in floating point form.

**ELEMNT(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:

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

**ELEMNT(I+1):** The intensity, color, wink, and line structure bits.

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

**ELEMNT(I):** The start of the information, four bytes to the word, with the first byte containing the number of bytes in the sub-block.
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Active Device, 3, 18</td>
</tr>
<tr>
<td></td>
<td>ALIAS=&lt;value&gt; (options item), 22, 53, 55</td>
</tr>
<tr>
<td></td>
<td>ANGLE=&lt;value&gt; (options item), 8, 15, 35, 64</td>
</tr>
<tr>
<td>B</td>
<td>Barkow, J., 14</td>
</tr>
<tr>
<td></td>
<td>BEEP (options item), 22, 59, 69</td>
</tr>
<tr>
<td>C</td>
<td>CALCOMB Drum Plotter, 1-2, 38-40, 46-48, 70, 80</td>
</tr>
<tr>
<td></td>
<td>CALCOMB Microfilm Plotter, 48-51</td>
</tr>
<tr>
<td></td>
<td>CAL/FCN (options item), 17, 49</td>
</tr>
<tr>
<td></td>
<td>CAL/10D (options item), 17, 47</td>
</tr>
<tr>
<td></td>
<td>CAL/92D (options item), 17, 47</td>
</tr>
<tr>
<td></td>
<td>CAL/60M (options item), 17, 49</td>
</tr>
<tr>
<td></td>
<td>Character Size Options, 3, 5, 8, 20, 40-49, 51-52, 57-58, 59, 61, 64, 65, 67-68</td>
</tr>
<tr>
<td></td>
<td>Character Stroke Generator, 3, 8, 35, 70-75</td>
</tr>
<tr>
<td></td>
<td>CLEAR (options item), 6, 21, 47, 50, 53, 55, 59, 65, 69, 78, 85</td>
</tr>
<tr>
<td></td>
<td>Color Code Options, 5, 7, 8, 9, 10, 15, 16, 20, 30</td>
</tr>
<tr>
<td></td>
<td>CONTINUE (options item), 6, 78, 85</td>
</tr>
<tr>
<td>D</td>
<td>DDNAME=&lt;value&gt; (options item), 17, 47, 49, 52, 55, 59, 64, 68</td>
</tr>
<tr>
<td></td>
<td>Default Values, 5, 47, 50, 53, 55, 59, 65, 68</td>
</tr>
<tr>
<td></td>
<td>DTC (options item), 20, 21, 59, 65</td>
</tr>
<tr>
<td></td>
<td>DEVCL (options item), 19</td>
</tr>
<tr>
<td></td>
<td>DISKOPS (options item), 17, 24, 55</td>
</tr>
<tr>
<td></td>
<td>Dixon, W. J., 31</td>
</tr>
<tr>
<td></td>
<td>DUNIT=&lt;value&gt; (options item), 12</td>
</tr>
<tr>
<td>E</td>
<td>Element, see Graphic Element</td>
</tr>
<tr>
<td></td>
<td>Element-Subroutine, 15, 21, 62</td>
</tr>
<tr>
<td></td>
<td>Error Processor, 4, 38, 77-79</td>
</tr>
<tr>
<td></td>
<td>ESOFM (options item), 23, 65</td>
</tr>
<tr>
<td></td>
<td>ESUN (options item), 23, 65</td>
</tr>
<tr>
<td></td>
<td>ESTNC (options item), 23, 65</td>
</tr>
<tr>
<td></td>
<td>ESWB (options item), 21, 65</td>
</tr>
<tr>
<td></td>
<td>EYED=&lt;value&gt; (options item), 32</td>
</tr>
<tr>
<td>F</td>
<td>FRAM (attention), 25, 59, 62-63, 65</td>
</tr>
<tr>
<td></td>
<td>FRAM (options item), 22, 65</td>
</tr>
<tr>
<td></td>
<td>Franek, B., 14</td>
</tr>
<tr>
<td></td>
<td>FULSCR (options item), 48-50, 51-53, 67-68</td>
</tr>
<tr>
<td>G</td>
<td>GET (options item), 19</td>
</tr>
<tr>
<td></td>
<td>Giorno, T. C., 31</td>
</tr>
<tr>
<td></td>
<td>GIF/IDI (options item), 17, 64</td>
</tr>
<tr>
<td></td>
<td>GIF-IDI/ION Display Console, 2, 60-66, 81</td>
</tr>
<tr>
<td></td>
<td>Giulieri, S. A., 14</td>
</tr>
<tr>
<td></td>
<td>Graphic Element, 2, 5-16, 20, 85-87</td>
</tr>
<tr>
<td>H</td>
<td>Hidden Line Problem, 13</td>
</tr>
<tr>
<td></td>
<td>HITH=&lt;value&gt; (options item), 12</td>
</tr>
<tr>
<td>I</td>
<td>IBM 2250 Display Console, 1-2, 40-45, 57-60, 80-81</td>
</tr>
<tr>
<td></td>
<td>IBM2250 (options item), 17, 58</td>
</tr>
<tr>
<td></td>
<td>IDIOION Display Console, see</td>
</tr>
<tr>
<td></td>
<td>GIF-IDI/ION Display Console</td>
</tr>
<tr>
<td></td>
<td>IDSCEP (options item), 17, 67</td>
</tr>
<tr>
<td></td>
<td>INCL (options item), 21, 59, 65</td>
</tr>
<tr>
<td></td>
<td>INPUT (options item), 24, 54, 55</td>
</tr>
<tr>
<td></td>
<td>Intensity Level Options, 5, 7, 8, 9, 10, 15, 16, 28, 30, 49, 52, 64, 66, 68</td>
</tr>
</tbody>
</table>
K
KBD (attention), 25, 59, 65, 69
Keyboard Input Buffer, 27-29, 58, 61, 67
Kronmal, R. A., 31
Kubert, B., 14

L
LANGSCH (options item), 48-50
LCASE (options item), 28, 69
LDEC=<value> (options item), 11
LEADER=<value> (options item), 49
Leukn, C. R., 31
Light Pen Input Buffer, 29-30, 61-62
Line Structure, 5, 8, 10, 15, 16, 30, 64, 66
LINK-EDITOR, 37, 70, 74, 79
LITES=<value> (options item), 22, 65
LOTH=<value> (options item), 12
LOWER (options item), 13
LPEN (attention), 25, 59, 65
LPSB (attention), 26, 65
LPSH (attention), 26, 65
LUXD=<value> (options item), 11
LXOFF=<value> (options item), 11
LYOFF=<value> (options item), 11

M
MAGICNWD=<value> (options item), 68
MAIDFIL=<value> (options item), 59
MAXEMN=<value> (options item), 58, 64
MAXESIZ=<value> (options item), 58, 64
MILTEX Terminal Manager, 66-68
MINESIZ=<value> (options item), 21, 64, 65
MINHV (options item), 11
Motion Picture Camera, 23, 62-63

N
NAME=<value> (options item), 24
NECP (options item), 20, 65
NDE (options item), 21, 59, 65
NOCONM (options item), 13
NOLABEL (options item), 11
NONE, 25
NSHR=<value> (options item), 12

O
OFF (options item), 20, 22, 26, 59, 65
OMIT (options item), 20, 21, 59, 65
ON (options item), 20, 22, 59, 65
OPTM argument, 4, 5, 15-19
Orthogonal Transformation, 13-14, 33-34
Overlay Structures, 37

P
PFKB (attention), 25, 65
PFKM (attention), 25, 65
PFLE (attention), 26, 65
PFLH (attention), 26, 65
PICTID=<value> (options item), 52, 55
PICTSQD=<value> (options item), 52, 55
PROGL (options item), 19
Programmer Coordinate System, 2, 80-81
Projective Transformation, 13-14, 32-33
Prueitt, H. L., 14
PUT (options item), 19

R
RDCSOP (options item), 17
RESET (options item), 6, 24
Round Numbers, 12, 31
RTTC (options item), 20, 65

S
SCALE=<value> (options item), 15, 64
Scaling and Scissoring, 2-3, 18-19, 59, 60-61
SHOW (options item), 23, 63, 65
SHOWH (options item), 23, 63, 65
SPACING=<value> (options item), 0, 35, 47, 70
Szabo, J., 14

T

TEKTROMIX 4013 Display
Terminal, 51-54, 66-69
T44013 (options item), 17, 68
TITLE=<value> (options item), 47, 49
TPCTL (options item), 23, 69
TPDNU (options item), 23, 65
TPDRS (options item), 23, 65
TPGET (options item), 22, 65
TPIAC (options item), 23, 65
TPFUT (options item), 22, 65
TPRNV (options item), 22, 65
Tracking Pattern, 22-23, 61-62

U

UGAXIS, 5, 11-12, 31, 47, 49, 52, 55, 58, 64, 68
UCALD02, 48
UCALD02, 51
UCLOS, 16, 18, 47, 50, 53, 55, 59, 64, 68
ULECTL, 35, 48, 51, 53, 56, 60, 66, 69
UCCTL, 20, 22-24, 59, 62-63, 65, 69
UCEATH, 25, 26, 59, 65, 69
UGBPS02, 56
UGIXY, 12, 31-32, 48, 50, 53, 55, 59, 66, 69
UGCATH, 25-26, 59, 65, 69
UGDIV, 35-36, 48, 51, 53, 56, 60, 66, 69
UGFUT, 5, 16, 49, 64
UGGET, 20, 24, 36, 54, 55
UGHIT, 5, 6-7, 24, 47, 49, 52, 58, 64, 68, 85
UGINV, 5, 15, 64
UGKEH, 5, 7-8, 11-12, 13, 47, 49, 52, 54, 58, 64, 78
UGLNS, 5, 10, 30, 35, 47, 49, 52, 54, 58, 64, 68
UGPMT, 5, 7, 47, 49, 52, 54, 58, 64, 68
UGPTS, 5, 9, 47, 49, 52, 54, 58, 64, 68
UGPMT, 20-21, 47, 50, 53, 55, 59, 64, 65, 68, 74
UGHIT, 5, 8-9, 11-12, 47, 49, 52, 54, 56, 64, 68, 70, 74
UGCIP02, 66
UGID5002, 60
UGIEFO, 16, 19, 47, 50, 53, 55, 59, 65, 68
UGHGET, 27, 28-29, 59, 66, 69
UGHPMT, 27-28, 59, 65, 69
UGIEFT, 29, 30, 66
UGHLPMT, 29-30, 66
UGOPHE, 16-18, 47, 49, 52, 55, 58, 64, 68
UGORIN, 14, 33-34, 48, 51, 53, 56, 60, 66, 69
UGPCT, 20, 21-22, 47, 50, 53, 55, 59, 65, 68
UGPFOJ, 32, 33, 34, 48, 51, 53, 56, 60, 66, 69
UGTOL, 48, 51, 53, 56, 60, 66, 69, 74-75
UGPLE, 25, 27, 59, 65, 69
UGSCAL, 16, 18-19, 47, 50, 53, 54, 55, 59, 64, 68, 80
UGSLCT, 16, 18, 47, 50, 53, 55, 59, 64, 68
UGX1302, 69
UGVIEW, 14, 32-33, 34, 48, 50, 53, 55, 59, 66, 69
UGX1502, 53
UGXER, 77-79
UGXTOP, 48, 51, 53, 56, 60, 66, 69, 74-75
UGX000, 37
UGX001, 48, 51, 53, 56, 60, 66, 69, 78
UGX001A, 79
UGX002, 48, 51, 53, 56, 60, 66, 69, 83-84
UGX03A, 70
UGX03B, 75
UGX122, 81
UGX303, 48, 51, 53, 56, 60, 66, 69, 79
UG3DH3, 5, 13-15, 47, 49, 52, 55, 58, 64, 68
Williamson, H., 14
Umbaugh, C. Options, 5, 7, 8, 9, 10, 15, 16, 64
Wright, T., 14
SYLIRU Text Editor, 51-52, 66-67
SYLO13 (options item), 17, 52

x
X=<value> (options item), 35
XHI=<value> (options item), 32, 34
XLO=<value> (options item), 32, 34
XSPACING=<value> (options item), 8-9, 47, 70
XSYSTEM=<value> (options item), 17

Y
Y=<value> (options item), 35
YHI=<value> (options item), 32, 34
YLO=<value> (options item), 32, 34

3-D Viewer, 63-64