CGTM No. 170
February 1976

Revised:
December 1976
January 1978

THE SLAC UNIFIED GRAPHICS SYSTEM

PROGRAMMING MANUAL

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

TABLE OF CONTENTS

TABLE OF CONTENTS

SECTION 1: AN INTRODUCTION TO THE UNIFIED GRAPHICS SYSTEM

The SLAC Unified Graphics System is a collection of FORTRAN and PL/1 subroutines which may be used to program for the graphic devices which can be driven from the central computing facility at SLAC. Currently the central computing facility consists of an IBM 360/Model 91 and two IBM 370/Model 168 computers running under the ASP operating system.

This document contains a complete description of the Unified Graphics System from the users point of view. A description of the internal operation of the system is contained in [BEA-1].

Since this document is rather long, the following suggestions may be helpful to the person trying to read it for the first time.

1. Read the remainder of Section 1 carefully. These sections supply a basic introduction to the Unified Graphics System and include a number of examples.
2. In Section 2, read all of the introductory sections and the descriptions of subroutines UGEINT, UGEPNT, UGELIN, UGETXT, UGOPEN, UGCLOS, UGEPUT, and UGPICT. These eight subroutines will allow a programmer to use any non-interactive graphic device that is supported by the system.
3. If you are interested in interactive graphic devices, you should also read about subroutines UGCTBL, UGEATN, UGDATN, UGRATN, UGKPUT, and UGKGET.
4. Read the parts of Sections 4 and 5 which describe the graphic devices you expect to use.
5. Study the examples in Section 6. 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 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. 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.

The Unified Graphics System is an attempt to eliminate many of these difficulties for a large class of graphic applications. The 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. At the same time, this system allows the programmer to take advantage of most of the special features of a specific graphic 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.

The Unified Graphics System is a collection of very general purpose subroutines. In any specific graphic application it is usually convenient to build a level of subroutines which are tailored to the application and which call the subroutines in the Unified Graphics System. There are two sets of subroutines available which use the Unified Graphics System, and greatly simplify the task of producing graphs, scatter plots, and histograms. TOP DRAWER [CHA-1, CHA-2] is a very versatile package which can be used at a number of levels. At one level, TOP DRAWER provides an executable load module which reads data cards and produces graphs and histograms. At another level, a programmer may call the TOP DRAWER subroutines directly. A second group of subroutines is the DPAK and HPAK package [LOG-1]. The DPAK and HPAK subroutines can send their output to the printer as well as to any of the non-interactive devices supported by the Unified Graphics System. Anyone who is principally interested in plotting graphs, scatter plots, or histograms should investigate one of these packages to see if it is sufficient before trying to use the Unified Graphics System directly.

SECTION 1.2: A BRIEF DESCRIPTION OF THE SYSTEM

The remainder of Section 1 is an informal introduction to the more important subroutines in the Unified Graphics System. In these sections, the subroutines will be described by means of examples showing their function. Complete descriptions of each of the subroutines will be found in Section 2. The examples, in Section 1, will all be in FORTRAN. The corresponding statements in PL/1 are similar but not exactly the same; the calling sequences in FORTRAN and PL/1 are different in many cases, and the programmer should examine Section 2 carefully to see these differences.

One of the common features of almost all subroutines is their first argument which is a character string. This character string, the "options list", is used to pass optional information to the subroutine. The options list, in FORTRAN, must be terminated by an asterisk.

SECTION 1.2.1: CREATING PICTURES

Pictures are created in exactly the same manner for all of the graphic devices supported by this system. A programmer creates descriptions of pictures in the following manner:

1. An array is defined in the program.
2. The array is initialized by calling subroutine UGEINT.
3. Other subroutines, UGEPNT, UGELIN, UGETXT, etc., are called to pack the descriptions of points, lines, and text material into the array.

The information which has been packed into an array in this manner is called a "graphic element". A later section will describe a subroutine, UGEPUT, which can be used to transmit a graphic element to a graphic device. A complete picture may consist of many graphic elements. In essence, a graphic element is a device-independent description of a partial picture. Thus, the creation of a picture in this system is basically a two-step process. In the first step, the programmer packs a device-independent description of a partial picture into an array. The second step occurs when the information in this array is converted to device orders and transmitted to the device.

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. Subroutines are provided which allow the programmer to change these scaling and scissoring limits but it is usually better, especially for the beginning user, to use the default values. Other default values in the system assume that these values have not been changed. In general, the manipulation of these scaling and scissoring limits is a device-dependent operation and should be avoided; however, rectangular pictures may be generated in a device-independent manner by using subroutine UGRSCL.

An example of FORTRAN statements which define and initialize a graphic element are:

```
      INTEGER*4    ELEMNT(500)
      CALL UGEINT('CLEAR*',ELEMNT,500)
```

The first argument of UGEINT may be used to perform other operations which will not be described here. The third argument must specify the dimension of the array.

To add the description of a point at (0.1,0.2) to the graphic element, the programmer may write:

```
      CALL UGEPNT('BRIT*',0.1,0.2,ELEMNT)
```

When the element is finally transmitted to a graphic device, the point will be in the "bright" mode if the device has intensity level control; if the device does not have intensity level control, this intensity level item is ignored.

A simple way to specify lines is to give the end points, one at a time, to UGELIN along with a "blanking bit" which tells the device whether or not to draw when moving to the point. For

example, to draw a line segment from (0.3,0.4) to (0.5,0.6), the programmer should write:

```
      CALL UGELIN('*',0.3,0.4,0,ELEMNT)
      CALL UGELIN('*',0.5,0.6,1,ELEMNT)
```

The fourth argument is the blanking bit; a zero means a blank movement and a one means a drawn line. If, immediately after these two lines of code, the programmer writes:

```
      CALL UGELIN('*',0.7,0.8,1,ELEMNT)
```

then another line segment will be drawn from (0.5,0.6) to (0.7,0.8).

A programmer may add the description of character data to a graphic element by calling subroutine UGETXT. The specification of text data is more complex than that of lines and points because of the great diversity of graphic devices in this area. Some graphic devices do not have hardware character generators, while some others have character generators of great versatility. Consider the following statement:

```
      CALL UGETXT('SMAL*',0.1,0.2,'A',1,ELEMNT)
```

In this statement, the first argument says that the characters are to be drawn by the hardware character generator in the small size. If the graphic device does not have a character generator, the Unified Graphics System will select an appropriate SPACING value and generate the characters as described below. The second and third arguments give the X and Y coordinates of the center of the first of the characters which are to be displayed. The fourth and fifth arguments give the character string and its length. Now consider the statements:

```
      CALL UGETXT('SPACING=0.02*',0.3,0.4,'AB',2,ELEMNT)
      CALL UGETXT('XSPACING=0.02*',0.5,0.6,'ABC',3,ELEMNT)
```

In the first statement, the SPACING item says that the characters are to be produced by drawing the characters with short line segments (called "strokes"). The value, 0.02 in this case, is the spacing between the centers of consecutive characters. In the second statement, the XSPACING item gives a suggested character spacing. This system will use the hardware character generator if the device has one and if it can match the spacing value reasonably close; otherwise, the stroke generator is used. The first argument can also be used to specify other options like intensity levels and the orientation of the characters. The orientation item is ANGLE=<value> where the value is the angle, in degrees, that the characters make with the horizontal in a counter-clockwise direction. Multiple versions of the character stroke generator are available. The default version, the Basic stroke generator, produces all of the characters on the IBM 029 keypunch. Alternate versions include upper and lower case Roman and Greek letters, a large number of special characters, and a flexible subscripting and superscripting ability.

Now consider the following statements:

```
      INTEGER*4 ELEMNT(150)
      CALL UGEINT('CLEAR*',ELEMNT,150)
      DO 102 I=1,6
        DO 101 J=1,6
          X=0.05*FLOAT(I)+0.6
          Y=0.05*FLOAT(J)+0.6
          CALL UGEPNT('*',X,Y,ELEMNT)
```

```
101   CONTINUE
102 CONTINUE
    DO 103 I=1,11
      A=0.01745329*(9.0*FLOAT(I-1))
      X=0.5*COS(A)+0.1
      Y=0.5*SIN(A)+0.1
      CALL UGELIN('*',0.1,0.1,0,ELEMNT)
      CALL UGELIN('*',X,Y,1,ELEMNT)
103 CONTINUE
    CALL UGETXT('SPACING=0.1*',0.2,0.8,'TEST',4,ELEMNT)
    CALL UGETXT('SPACING=0.1,ANGLE=45*',0.7,0.2,
   X  'TILT',4,ELEMNT)
```

Figure 1.2.1 shows the picture which will be produced by these statements along with the layout of the default programmer coordinate system.

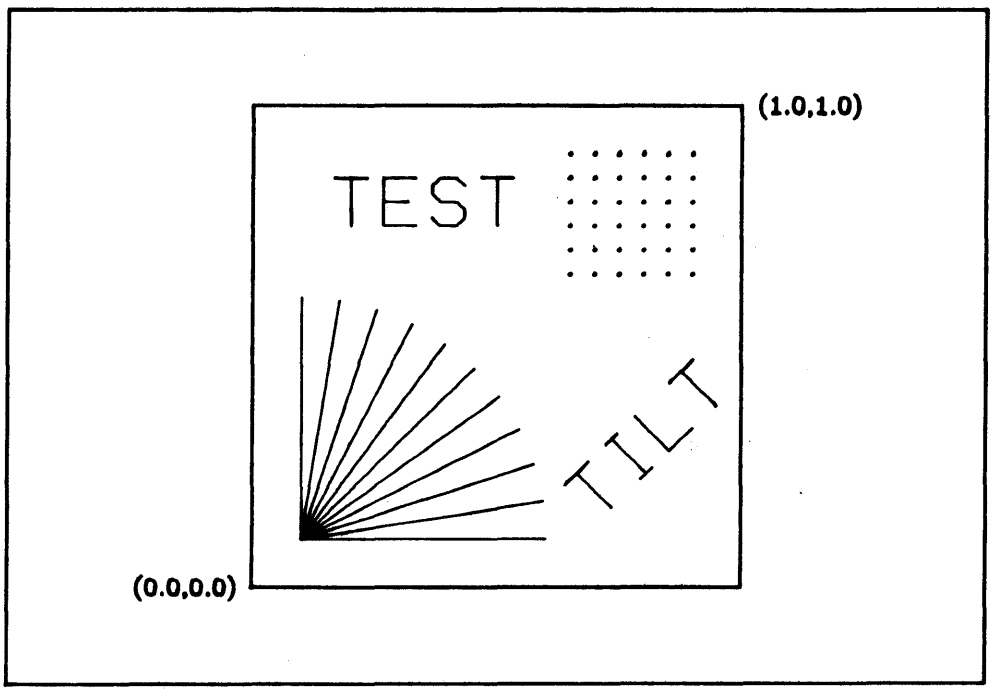


Figure 1.2.1: Examples of UGEPNT, UGELIN, and UGETXT.

In addition to subroutines UGEPNT and UGELIN which supply a single point or single line end point at a time, there are subroutines which supply a group of points or line end points at once. Suppose the arrays XARRAY and YARRAY contain the X and Y coordinates of points. Then the statement:

```
      CALL UGEPTS('*',XARRAY,YARRAY,NPTS,ELEMNT)
```

will add the NPTS points in the arrays to the element. The statement:

```
      CALL UGELNS('*',XARRAY,YARRAY,NPTS,1,1,ELEMNT)
```

will draw lines from the first point to the second point, from the second point to the third, etc. The fifth and sixth arguments in the example force line segment to be drawn between each consecutive pair of points. Other combinations of the fifth and sixth arguments are possible which can, for example, cause every third line to be blanked.

The Unified Graphics System supplies a number of additional subroutines for describing pictures. There is, for example, a subroutine called UGAXIS which generates an axis with tic marks and labels with a single call.

The principal advantage of having the programmer provide an array to hold the picture description data is that it provides an explicit receptacle for the device-independent form of the picture. For interactive devices, this scheme also serves to break a picture up into individual parts which may be identified and manipulated. On first reading, it may seem that this scheme has introduced one more problem for the programmer, namely assuring that the array does not overflow. However, the programmer can use the error processor to identify this problem and cause the picture data to be transmitted to a graphic device whenever the array fills up.

SECTION 1.2.2: NON-INTERACTIVE GRAPHIC DEVICES

Subroutine UGOPEN is used to initialize a graphic device. It usually should be called before any other subroutine in this package is called. It is when the program calls UGOPEN that the Unified Graphics System first knows what device the program intends to use. The call to UGOPEN, therefore, causes the device-dependent portion of the system to be brought into memory. For example, the statement:

```
      CALL UGOPEN('CALDRSM*',99)
```

specifies that the graphic device is the 10 inch CALCOMP Drum Plotter. In this case, the program will assume that the JCL contains a DD card with a DDNAME of UGDEVICE and the plotter commands will be written to this data set. The second argument in UGOPEN is the device identification; it is only important when a program is using more than one graphic device. The options list can also be used to supply a number of other parameters, some of them device-dependent, to the system.

Generally, one of the final things a program should do is to call subroutine UGCLOS. Calling this subroutine signals the system that the program intends to make no more use of the graphic device. A typical call is:

```
      CALL UGCLOS('*')
```

Calling UGCLOS allows the system to delete the device-dependent code from memory, and, for some devices, write out the final buffer full of graphic information.

Subroutine UGPICT is used to control the picture on a graphic device. The only operation which applies to non-interactive devices is the picture clearing operation:

```
      CALL UGPICT('CLEAR*',0)
```

This operation always signals that a new picture is being started; on a mechanical device like the CALCOMP Drum Plotters, the pen moves over to a fresh drawing area; on a device like an IBM 2250, the screen is cleared.

The subroutine which adds a graphic element to the current picture is UGEPUT. The subroutine call:

```
      CALL UGEPUT('*',0,ELEMNT)
```

will take the graphic data in the array ELEMNT, transform it to device orders and transmit it to the device. After subroutine UGEPUT has been called, the array ELEMNT may be re-used.

The second arguments in both UGPICT and UGEPUT have meaning only for interactive devices. For non-interactive devices, they should be zero.

SECTION 1.2.3: INTERACTIVE GRAPHIC DEVICES

Opening and closing an interactive device is the same as for a non-interactive device. For example, the statement:

```
      CALL UGOPEN('IBM2250,DDNAME=GDEV,MAXENUM=50*',99)
```

says that the graphic device to be used is an IBM 2250; the DD card which specifies which unit is to be used has a DDNAME of GDEV (instead of the default UGDEVICE); and the maximum number of elements which can be contained in any one picture is 50 (instead of the default 32). This limit on the number of elements in a picture applies only to refresh display devices.

Display device control is more complex on interactive devices, especially for refresh display devices. On such devices, each element may be manipulated individually. For example, the statement:

```
      CALL UGEPUT('DETC*',7,ELEMNT)
```

gives the element an identification of 7 and says that the element is to be detectable with the light pen. This numeric identification is the name by which the programmer will refer to the element when the element is to be manipulated. If a second element with the identification of 7 is transmitted and the device is a refresh display device, then the new element will replace the first. Elements transmitted with zero identifications are considered to be unidentified. A picture can contain many unidentified elements but they cannot be manipulated.

The use of subroutine UGPICT has been considerably extended for interactive devices also. For instance, the statement:

```
      CALL UGPICT('NDET,OMIT*',7)
```

will change element 7 so that it is no longer light pen detectable, and put it into the "omit" state. An element in the omit state will not appear on the display device. It can be put back into the "include" state by the statement:

```
      CALL UGPICT('INCL*',7)
```

This statement causes the element to reappear on the screen. Thus, this include-omit switching can be used to temporarily blank out an element and then restore it without re-generating it with subroutines UGEPNT, etc. and re-transmitting it with UGEPUT. Finally, the statement:

```
      CALL UGPICT('CLEAR*',7)
```

will delete element 7 from the picture.

The element manipulation that has been described here only applies to refresh display scopes; most storage scopes have an inherent hardware limitation which prevents individual elements from being deleted from a picture. On such storage scopes, as on non-interactive devices, the only valid operation is the clear screen operation. On storage devices, the element identification is usually ignored.

Another subroutine, UGCTRL, is provided for interactive devices to perform a number of special operations on the display device. Most of the operations are device-dependent; however, a common operation is:

```
      CALL UGCTRL('BEEP*',IARRAY,XARRAY)
```

which causes the audible alarm on the device to sound. IARRAY is an eight word INTEGER*4 array, and XARRAY is a two word REAL*4 array which is used for input and output of parameters for some of the other operations.

There are other 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. Some of the items which can generate attentions are the keyboard (on all of the interactive devices) or the light pen (on some devices). If an attention is "enabled", then the console operator can generate this attention and the program can determine that the attention has been generated. If an attention is "disabled", the actions of the console operator will not be communicated to the program. Initially, all attentions are disabled. Attentions are enabled and disabled with subroutines UGEATN and UGDATN. For example:

```
      CALL UGEATN('KBRD,LPEN*')
```

will enable the keyboard and the light pen. The statement:

```
      CALL UGDATN('LPEN*')
```

will disable the light pen.

Keyboard attentions are generated by a variety of keys: on the IBM 2250, the KBRD attention is generated by holding the key labeled ALT down and depressing the key labeled END; on the TEKTRONIX 4013, it is generated by the key labeled RETURN. On the IBM 2250, the LPEN attention is generated by pointing the light pen at a detectable element and depressing the foot pedal.

Subroutine UGRATN may be used to wait for an attention to be generated by the console operator. The statement:

```
      CALL UGRATN('*',-1.0,ATCODE,IARRAY,XARRAY)
```

will put the program into the wait state until an attention occurs. When an attention occurs, subroutine UGRATN will return with the last three arguments set to indicate what happened. ATCODE is an INTEGER*4 variable which will be set to 'KBRD' or 'LPEN' to indicate what has occurred. IARRAY is an INTEGER*4 array with a dimension of eight. The first word will be set to 1 or 2 corresponding to the above possibilities for ATCODE. For the KBRD attention, no other information is available. However, more information is available for a LPEN attention. The principal item of information is in IARRAY(2) and consists of the identification of the element selected by the light pen. XARRAY

is a REAL*4 array of dimension 2 which often contains additional information.

The minus one being used as the second argument in UGRATN in the above example indicates that the subroutine is to wait until an attention occurs. If a non-negative value is used as the second argument, it indicates that subroutine UGRATN is to wait only for the given number of seconds. Thus the code:

```
      REAL*4        TIME
      TIME=5.0
      CALL UGRATN('*',TIME,ATCODE,IARRAY,XARRAY);
```

will cause UGRATN to wait for five seconds or for an attention, whichever comes first. On exit from UGRATN, the value of TIME will be the amount of unexpired time in the five second interval. If no attention has occurred when UGRATN returns, then ATCODE will contain 'NONE' and IARRAY(1) will contain zero.

A "keyboard input buffer" is a special kind of text element which may be displayed on an interactive device 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.

A keyboard input buffer is added to the display by calling subroutine UGKPUT. A typical call is:

```
      CALL UGKPUT('SMAL*',0.1,0.2,5,'ABCD',4)
```

This statement adds a keyboard input buffer with an identification of 5 to the display. This buffer may be manipulated by UGPICT, just like any other element. In this example, the cursor will be positioned at the character at the point (0.1,0.2). On refresh devices like the IBM 2250, the characters ABCD will also appear on the screen and the console operator may change these characters or retype them completely. On this type of device, the programmer may put many input buffers on the screen at one time. On the IBM 2250, the JUMP key will move the cursor from one input buffer to another. Subroutine UGKPUT works a little differently on a storage scope like a TEKTRONIX 4013. On this device, the characters ABCD of the example would not be written on the screen and only one input buffer can be on the screen at one time.

A keyboard input buffer is read from the screen by subroutine UGKGET. Consider the example:

```
      INTEGER*4     STRING(4)
      CALL UGKGET('*',5,STRING,16,NSTR,ICUR)
```

This example will read the keyboard input buffer with an identification of 5 into the array STRING. The fourth argument gives the maximum number of characters which can be put in STRING. UGKGET puts the actual number of characters read into NSTR and sets ICUR to the index of the character which contained the cursor. On an IBM 2250, the entire buffer, as given by UGKPUT, is read back, while on the TEKTRONIX 4013, only those characters actually typed will be returned.

To summarize attention handling, simple interaction between the program and the console operator can be done as follows:

1. The program puts a picture on the screen, adds a keyboard input buffer, and assures that the keyboard is enabled.
2. The program waits for an attention. At this time, the console operator can enter text into the input buffer. When the text has been entered, the console operator generates a KBRD attention.
3. Upon receiving the attention, the program can read the keyboard input buffer and process the data it contains.

Interaction with a light pen would be done in an analogous manner.

This section gives a complete description of each of the subroutines provided in the 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 OPTIONS) which is used to specify information to the system which may be optional or device-dependent. This character string, the options list, may contain a number of items separated by commas. These items are of five types:

1. A simple flag consisting of a sequence of alphabetic and/or numeric characters.
2. A flag followed by an equal sign followed by an integer.
3. A flag followed by an equal sign followed by a floating point number (i.e., a number which may contain a decimal point).
4. A flag followed by an equal sign followed by a string of characters. This string of characters cannot include blanks or commas unless the string is enclosed in apostrophes, in which case the string cannot contain apostrophes.
5. A flag followed by an equal sign followed by a string of bits (the characters "0" or "1").

In FORTRAN, the entire options list must be terminated by an asterisk, and asterisks cannot appear anywhere else in an options list. Thus, an example of an options list in FORTRAN is:

```
'CAL16UM,LEADER=50,DDNAME=PLOTS,TITLE='''EXAMPLE TITLE'''*'
```

The equivalent options list in PL/1 would be exactly the same except that the terminal asterisk would not be there. The doubling of the apostrophes around the TITLE item is caused by the fact that the compiler will replace them with a single apostrophe. Blanks may occur in the options list at its beginning or end, and on either side of the commas separating items or the equal signs. 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 the options list 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 default 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 memory dump.
4. Print a message and terminate the program with a memory dump.

The error indicators, for non-terminal errors, 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, the index of the error, the level of the error, and a description of the error. Additional information on this subject will be found in the section on Error Processing. 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 given.

In the following descriptions of the subroutines, floating point or string arguments are always described as such; if nothing is said about the type of a parameter, the reader may assume that it is fixed point. In the FORTRAN version of this system, fixed point arguments are always INTEGER*4 unless there is an explicit statement to the contrary. Floating point arguments are always REAL*4. The arguments described as character strings may be integer or logical arrays of any precision. In the PL/1 version, fixed point arguments are always FIXED BINARY(15,0) unless stated otherwise, and floating point arguments are always FLOAT BINARY(21).

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 work integers. An example of a statement defining a graphic element in FORTRAN is:

```
      INTEGER*4    ELEMNT(500)
```

The equivalent statement in PL/1 is:

```
      DECLARE ELEMNT(500) FIXED BINARY(31,0);
```

As described below, the array ELEMNT may be initialized and have picture description data packed into it. After a graphic element has been initialized, the first word in it will always contain a count of the number of words in the array that are in use.

The first of the graphic element generation subroutines is UGEINT. This subroutine is used to initialize a graphic element before any picture description data is added to it. The next three subroutines, UGEPNT, UGELIN, and UGETXT, 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, UGEPTS and UGELNS, are similar to UGEPNT and UGELIN except that each call to these two subroutines will add an array of points or line segments to the graphic element. Finally, subroutines UGAXIS, UGXHCH, UG3DHS, and UGCNTR perform more complex operations; they add the description of a horizontal or vertical axis, the description of a cross-hatched region, the description of a mesh surface with hidden lines removed, or the description of a contour plot to a graphic element.

The OPTIONS argument in the element generation subroutines usually specifies picture description parameters. The key words used for these parameters are:

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

VDIM,DIMM,BRIT,VBRT Intensity level options. Any of four different intensity levels may be given. The key words stand for "very dim", "dim", "bright", and "very bright".

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

WINK,STDY Wink mode options. Some interactive devices have a feature whereby part of the display may flash on and off. The key words stand for "wink" and "steady".

When default values must be supplied by the system, these default values will be: SOLD, BRIT, COL1, and STDY. This system will utilize as many of these parameters as possible when the graphic element is transmitted to the display device. Items like the color codes can only be used if the display device has some hardware facility to produce multi-colored pictures. Other items like line structure is produced by internal software in the Unified Graphics System. An exception to the production of line structure is that it will not normally be produced on re-fresh display devices like the IBM 2250. The reason is that generating dots and dashes instead of a simple straight line on such a device would quickly fill the re-fresh buffer. Thus, a user of this system may make free use of any of these picture description parameters; when the graphic element is transmitted to the device, as many parameters as possible will be utilized within the limitations of the actual display device.

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. UGEINT initializes the first four words of the graphic element to contain bookkeeping information.

The calling sequences are:

```
(FORTRAN)  CALL UGEINT(OPTIONS,ELEMENT,NELEM)
```

(PL/1) CALL UGEINT(OPTIONS,ELEMENT);

The parameters in the calling sequences are:

OPTIONS 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 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 an external 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 Error Processing for an example of a case when this operation is necessary.

ELEMENT The graphic element which is to be initialized.

NELEM The length of the array ELEMENT (in words).

The errors detected by this subroutine are:

1(3): The length of the array ELEMENT is too small. It must be at least 4 words long. However, an array which is only 4 words long is not very useful because no data can be added to it.

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 sequences are:

(FORTRAN) CALL UGEPNT(OPTIONS,X,Y,ELEMENT)

(PL/1) CALL UGEPNT(OPTIONS,X,Y,ELEMENT);

The parameters in the calling sequences are:

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

VDIM,DIMM,BRIT,VBRT Intensity level options.

COL1,COL2,COL3,COL4 Color options.

WINK,STDY Wink mode options.

X The floating point X coordinate of the point.

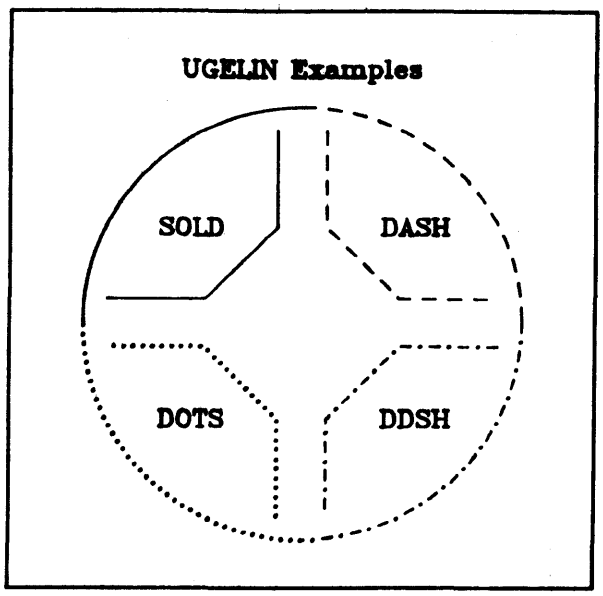
Y The floating point Y coordinate of the point.

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

The errors detected by this subroutine are:

9(1): Not enough room is available in the array ELEMENT 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. Line structure, as generated by this system, is relatively independent of point spacing on a curve. For example, the circular arcs in the examples were defined by a point every one degree. In spite of this close spacing, the dashes and dots are spaced more rationally along the arcs. In a dashed line, for example, the line always begins with a dash, and has dashes whose length is approximately one-third centimeter (the examples shown here were reduced to half size). A dash line may, therefore, not always end with a dash. The dots in a dotted line are spaced about one-fourth of a centimeter and a dot-dashed line also uses this spacing.

The calling sequences are:

```
(FORTRAN)  CALL UGELIN(OPTIONS,X,Y,BBIT,ELEMENT)
(PL/1)     CALL UGELIN(OPTIONS,X,Y,BBIT,ELEMENT);
```

The parameters in the calling sequences are:

- **OPTIONS** A character string which may contain any of the following items:
  - SOLD,DASH,DDSH,DOTS Line structure options.
  - VDIM,DIMM,BRIT,VBRT Intensity level options.
  - COL1,COL2,COL3,COL4 Color options.
  - WINK,STDY Wink mode options.
- **X** The floating point X coordinate of an end point of a line.
- **Y** The floating point Y coordinate of an end point of a line.
- **BBIT** The blanking bit. In FORTRAN, this parameter is an integer, while in PL/1 it is a bit string of length one. In either case, a zero indicates a blanked line and a one indicates a drawn line.
- **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 is available in the array ELEMENT to contain the new end point.

SECTION 2.1.4: SUBROUTINE UGETXT

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 [N/4]+3 words of information are added to the graphic element where N is the number of characters in the text and [X] is the greatest integer in X; otherwise [N/4]+7 words are added.

The calling sequences are:

```
(FORTRAN)  CALL UGETXT(OPTIONS,X,Y,TEXT,NTEXT,ELEMENT)
(PL/1)     CALL UGETXT(OPTIONS,X,Y,TEXT,ELEMENT);
```

The parameters in the calling sequences are:

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

- VSML,SMAL,LARG,VLRG Character size options. These key words stand for "very small", "small", "large", and "very large". The use of one of these items indicates that the text must be produced by the hardware character generator if the device has a hardware character generator. If the device does not have a character generator, the text will be produced by the stroke generator.
- XSPACING=<value> Suggested character spacing. The device character generator will be used if the between character spacing will approximately match the given value; otherwise the character stroke generator will be used.
- SPACING=<value> Character spacing. The character stroke generator must be used.
- ANGLE=<value> The angle that the characters make with the horizontal. The angle is measured, in degrees, in the counter-clockwise direction. The default value is an angle of 0.0 degrees.
- VDIM,DIMM,BRIT,VBRT Intensity level options.
- COL1,COL2,COL3,COL4 Color options.
- WINK,STDY Wink mode options.

If no character spacing of any kind is specified, then a default of XSPACING=0.015 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 The floating point X coordinate of the center of the first character.

Y The floating point Y coordinate of the center of the first character.

TEXT The string containing the characters to be added to the graphic element.

NTEXT The number of characters in TEXT.

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

The errors detected by this subroutine are:
1(2): The parameter TEXT contains more than 255 characters.
9(1): Not enough room is available in the array ELEMENT to contain the new text information.

Some additional comments about character size selection may be helpful. The options items SPACING and XSPACING specify the size of the characters relative to the current programmer scaling and scissoring limits. The options items VSML, SMAL, LARG, and VLRG specify the size independent of these current limits. Hardware character generators are, of course, independent of these limits. If the graphic device does not have a hardware character generator, the Unified Graphics System simulates one by using the current character stroke generator with the correspondence:

VSML SPACING=0.010
SMAL SPACING=0.015
LARG SPACING=0.020
VLRG SPACING=0.025

relative to the <u>default</u> programmer scaling and scissoring limits.

The size of the characters produced by the character stroke generators is determined by the scaling and scissoring limits in the X direction only; the values in the Y direction are not used. The character is then drawn so that it appears in its normal width to height ratio. Thus, different scaling and scissoring limits in the X and Y directions does not result in distorted characters and rotating a character does not change its size or shape. If distorted characters are required, the programmer will have to use subroutine UGCTOL.

There is a problem which can occur if the programmer tries to put characters generated by the device character generator too near the edge of the screen. The problem is that the user of the Unified Graphics System specifies the position of a character by giving the coordinates of its <u>center</u> while some actual devices use other parts of the character to locate its position. The result is that characters can often be plotted partially outside the area available to lines and points, but this is not consistent from device to device. Because of the differences in the hardware character generators, it is suggested that the programmer not try to plot hardware generated characters within one-half character height of the top or bottom of the screen or within one-half character width of the sides of the screen. This will keep the characters completely within the line and point plotting area and no problems should occur.

SECTION 2.1.5: SUBROUTINE UGEPTS

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 OPTIONS argument will be

scanned only once.

The calling sequences are:

```
(FORTRAN)  CALL UGEPTS(OPTIONS,XARRAY,YARRAY,NPTS,ELEMENT)
(PL/1)     CALL UGEPTS(OPTIONS,XARRAY,YARRAY,NPTS,ELEMENT);
```

The parameters in the calling sequences are:

OPTIONS A character string which may contain any of the following items:
- VDIM,DIMM,BRIT,VBRT Intensity level options.
- COL1,COL2,COL3,COL4 Color options.
- WINK,STDY Wink mode options.

XARRAY A floating point array containing the X coordinates of the points.

YARRAY A floating point array containing the Y coordinates of the points.

NPTS The number of points in XARRAY and YARRAY.

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

The errors detected by this subroutine are:

9(1): Not enough room is available in the array ELEMENT to contain the new points. The graphic element is unchanged.

SECTION 2.1.6: SUBROUTINE UGELNS

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 UGELIN, but one call to this subroutine is more efficient because the OPTIONS argument will be scanned only once.

The calling sequences are:

```
(FORTRAN)  CALL UGELNS(OPTIONS,XARRAY,YARRAY,NPTS,BBITS,
                       NBBITS,ELEMENT)
(PL/1)     CALL UGELNS(OPTIONS,XARRAY,YARRAY,NPTS,BBITS,
                       ELEMENT);
```

The parameters in the calling sequences are:

OPTIONS A character string which may contain any of the following items:
- SOLD,DASH,DDSH,DOTS Line structure options.
- VDIM,DIMM,BRIT,VBRT Intensity level options.
- COL1,COL2,COL3,COL4 Color options.
- WINK,STDY Wink mode options.

XARRAY A floating point array containing the X coordinates of the end points of the line segments.

YARRAY A floating point array containing the Y coordinates of the end points of the line segments.

NPTS The number of end points in XARRAY and YARRAY.

BBITS The blanking bits. In FORTRAN, this parameter is an integer array which may contain one bit per word or 32 bits per word. In PL/1, this parameter is a bit string of arbitrary length. The bits are selected

cyclically from BBITS. A single blanking bit of one results in a continuous curve joining the points; two blanking bits of one-zero results in a line being drawn between the first and second points, the third and fourth points, etc. Much more complex sequences can be constructed.

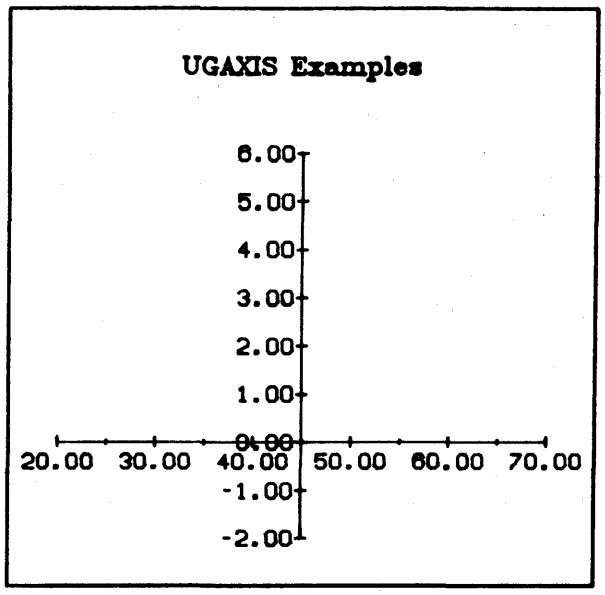
NBBITS The number of blanking bits in BBITS. If NBBITS is positive, the bits must be given one to a word; if NBBITS is negative, its absolute value is used for the count and the bits must be packed 32 to a word. When un-packed bits are used, the first blanking bit is the low order bit of BBITS(1); when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

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

The errors detected by this subroutine are:

9(1): Not enough room is available in the array ELEMENT to contain the new line segments. The graphic 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 UGETXT. If either of these subroutines cause the graphic element to overflow, then the graphic element is restored to the state that it was in when UGAXIS was called and UGAXIS then signals an error. The reader should refer to subroutine UGDXDY in the section on Miscellaneous Subroutines for help in assigning values to the parameters which result in "round numbers" being used for the labels. The FORTRAN version of this subroutine converts the labels from floating point to character string by using subroutine UGCNVF.

The calling sequences are:

```
(FORTRAN)  CALL UGAXIS(OPTIONS,LOPTIONS,TOPTIONS,INDVAR,LOVAR,
                       HIVAR,LOLAB,HILAB,NLAB,ELEMENT)
(PL/1)     CALL UGAXIS(OPTIONS,LOPTIONS,TOPTIONS,INDVAR,LOVAR,
                       HIVAR,LOLAB,HILAB,NLAB,ELEMENT);
```

The parameters in the calling sequences are:

OPTIONS A character string which may contain any of the

following items:

VERT Indicates that a vertical axis is required. The default setting is a horizontal axis.

MINMV 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.

NOLABL 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. For the FORTRAN version of this subroutine, LWID must satisfy 1≤LWID≤12.

LDEC=<value> The number of places to the right of the decimal point in the labels. The default value is 2. If LDEC has the value zero, then no decimal point will appear in the label. For the FORTRAN version of this subroutine, LDEC must satisfy 0≤LDEC≤10.

LXOFF=<value> The offset, in the X direction, of the first of the LWID characters in the label from the axis-tic mark intersection. The default value is -0.060 for a horizontal axis and -0.120 for a vertical axis.

LYOFF=<value> The offset, in the Y direction, of the first of the LWID characters in the label from the axis-tic mark intersection. The default value is -0.020 for a horizontal axis and 0.000 for a vertical axis.

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

HITM=<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.

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

LOPTIONS An options list that will be passed to NGELIN.

TOPTIONS An options list that will be passed to UGETXT.

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

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

HIVAR For a horizontal axis, this is the floating point X

coordinate of the high end of the axis in the programmer coordinate system; for a vertical axis, this is the floating point Y coordinate.

LOLAB A floating point value giving the label to be placed at the low end of the axis.

HILAB A floating point value giving the label to be placed at the high end of the axis.

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

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

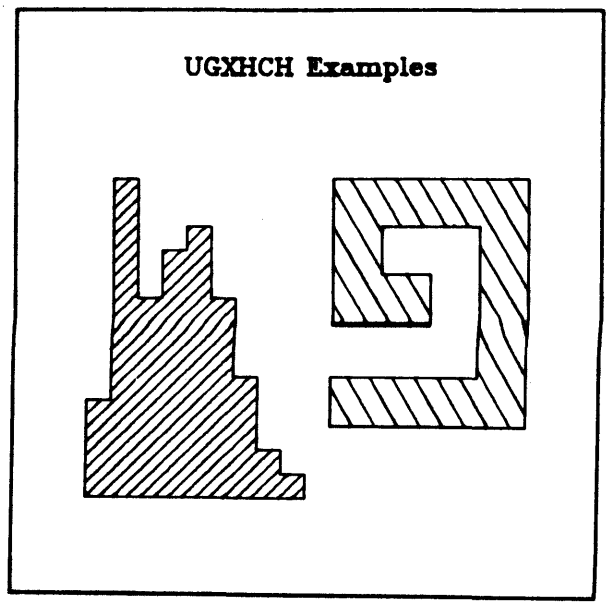
The errors detected by this subroutine are:

9(1): Not enough room is available in the array ELEMENT to contain the description of the axis. The graphic element is unchanged.

It is important to remember that the default values of the options items which specify the size and position of the labels assume that the default programmer coordinate system is in effect. If the current programmer coordinate system is very different from the default system, the programmer must change these values. For example, suppose the programmer is working with a device with a fixed plotting area and has changed the scaling and scissoring limits in the X direction to 0.0 to 10.0 (instead of the default 0.0 to 1.0). Thus, the length of the picture in the X direction, in its own coordinate system, is 10.0 times larger. To obtain the same axis in this coordinate system as one would get in the default system requires that the programmer set the values to LXOFF, LYOFF, LOTH, and HITH to 10.0 times their default values and increase the character size (by means of the TOPTIONS parameter) to 10.0 times the default value.

SECTION 2.1.8: SUBROUTINE UGXHCH

This subroutine may be used to add data describing a cross-hatched region to a graphic element. The region to be cross-hatched is specified by giving a sequence of points which describe a simple closed curve in the programmer reference system. The principal, intended purpose of this subroutine is to provide a simple means of cross-hatching histograms, but many other possibilities can be imagined. In referring to the examples on the right, the user should be aware that the outline of the cross-hatched region is not part of the information added to the graphic element by this subroutine. However, this outline can be generated by a single



call to subroutine UGELNS. This subroutine adds the actual data to the graphic element by calling subroutine UGELIN. If UGELIN causes the graphic element to overflow, then the graphic element is restored to the state that it was in when UGXHCH was called and UGXHCH then signals an error.

The calling sequences are:

```
(FORTRAN)  CALL UGXHCH(OPTIONS,LOPTIONS,XARRAY,YARRAY,NPTS,
                       WKAREA,LDIM,ELEMENT)
(PL/1)     CALL UGXHCH(OPTIONS,LOPTIONS,XARRAY,YARRAY,NPTS,
                       WKAREA,ELEMENT);
```

The parameters in the calling sequence are:

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

- SPACING=<value> The spacing between the lines of cross-hatching. The default value is 0.02.
- ANGLE=<value> The angle at which the lines of cross-hatching are drawn. The default value is 45.0.
- X=<value> The X coordinate of a point which will have a line of cross-hatching pass through it. The default value is XARRAY(1).
- Y=<value> The Y coordinate of a point which will have a line of cross-hatching pass through it. The default value is YARRAY(1).
- TOLER=<value> An internal tolerance which may have to be adjusted if the programmer reference system is very different from the standard system. The default value is 0.0001.

LOPTIONS An options list that will be passed to UGELIN.

XARRAY An array containing the X coordinates of the end points of line segments which bound the region to be cross-hatched.

YARRAY An array containing the Y coordinates of the end points of the line segments.

NPTS The number of end points in XARRAY and YARRAY.

WKAREA A floating point array which will be used as a work area. A dimension at least as large as the maximum number of intersections between a line of cross-hatching and the region boundary is required.

LDIM The dimension of WKAREA.

ELEMENT The graphic element which will have the description of the cross-hatching added to it.

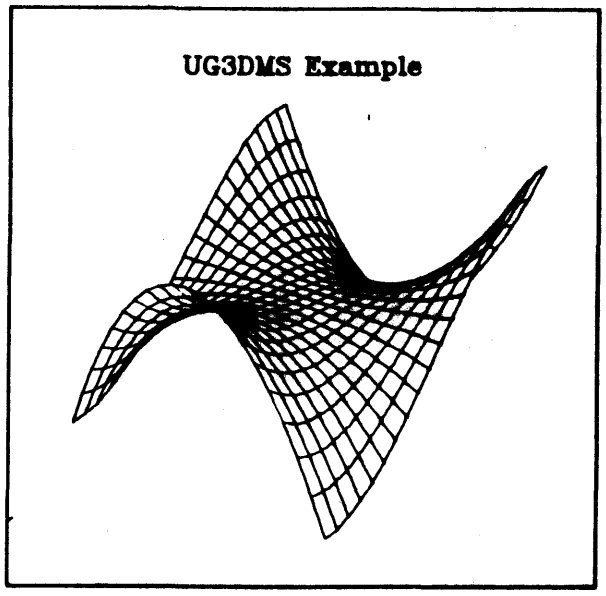
The errors detected by this subroutine are:

1(3): Either the first and last points in the region definition are not the same, or there are fewer than three points given.

2(3): The array WKAREA is not large enough.

9(1): Not enough room is available in the array ELEMENT to contain the description of the cross-hatching. The graphic element is unchanged.

SECTION 2.1.9: SUBROUTINE UG3DMS



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 causes the graphic element to overflow, then the graphic element is restored to the state that it was in when UG3DMS was called and UG3DMS 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.

Algorithms of this nature have been described in [KUB-1, WIL-1, WRI-1, BAR-1, WAT-1]. The algorithm used in UG3DMS is based on the information in [WRI-1, BAR-1]. Additional examples of this type of computer generated picture will be found in [PRU-1, PRU-2].

The calling sequences are:

```
(FORTRAN)  CALL UG3DMS(OPTIONS,LOPTIONS,ARRAY,MDIM,NDIM,TRANS,
                       WKAREA,LDIM,ELEMENT)
(PL/1)     CALL UG3DMS(OPTIONS,LOPTIONS,ARRAY,TRANS,WKAREA,
                       ELEMENT);
```

The parameters in the calling sequences are:

- OPTIONS A character string which may contain any of the following items:
  - LOWER Indicates that the under side of the surface is to be generated. Normally the upper side of the surface is generated. Two calls to this subroutine are necessary to generate the full view of a surface.
  - NOCOMM 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.
  - TOLER=<value> An internal tolerance which may have to be adjusted if the X, Y, and Z coordinates take on large values. The default value is 0.00005.
- LOPTIONS An options list that will be passed to UGELIN.
- ARRAY A floating point array which contains the X, Y, and Z

coordinates of the points on the surface. The format of ARRAY is:

```
--  X1   X2  ... XN
Y1  Z11  Z21 ... ZN1
Y2  Z12  Z22 ... ZN2
...          ...
YN  Z1N  Z2N ... ZNN
```

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YN) must be monotonically increasing. The X's and Y's do not have to be equally spaced, but better pictures usually result if this is the case.

HDIM — The first dimension of ARRAY; that is N+1.

NDIM — The second dimension of ARRAY; that is N+1.

TRANS — A floating point array of dimension 31 containing the transformation as produced by subroutine UGPROJ or UGORTH. There are a few restrictions on the views which may be used here. First, the eye point must not be directly above the surface but must be off to one side; and second, the transformation must be defined with a zero vector for HDIR.

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

LDIM — The dimension of WKAREA.

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

The errors detected by this subroutine are:

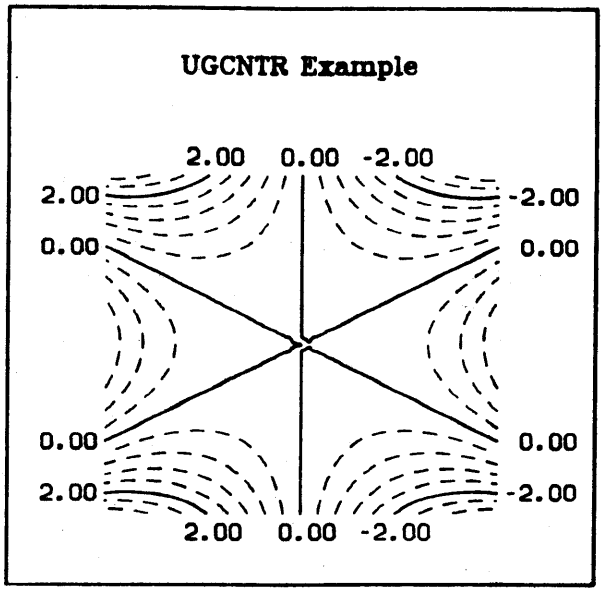
1(3): The dimension of the rectangular array ARRAY must be at least 3 by 3 to define a valid surface.

2(3): The array WKAREA is not large enough.

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

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 when using a projective transformation. It can often be overcome by moving to a distant eye position or using an orthogonal transformation.

SECTION 2.1.10: SUBROUTINE UGCNTR



This subroutine may be used to add the description of a contour plot of a surface to a graphic element. The surface is given by the Z coordinates of points above a rectangular grid in the X-Y plane. The surface is approximated by these rectangular patches. The X and Y coordinates must be given in the programmer coordinate system of the graphic device while the Z coordinate is in a coordinate system of the user's choice. The contour lines consist of "primary" and "secondary" contours. The primary contours will normally be labeled where they cross the boundary of the surface. This subroutine adds the actual data to the graphic element by calling subroutines UGELIN and UGETXT. If either of these subroutines causes the graphic element to overflow, then the graphic element is restored to the state that it was in when UGCNTR was called, and UGCNTR signals an error. The reader should refer to subroutine UGDXDY in the section on Miscellaneous Subroutines for help in assigning values to the parameters which result in "round numbers" being used for the labels. The FORTRAN version of this subroutine converts the labels from floating point to character string by using subroutine UGCNVF.

A discussion of an algorithm similar to the one used here will be found in [COT-1].

The calling sequences are:

```
(FORTRAN)  CALL UGCNTR(OPTIONS,POPTIONS,SOPTIONS,TOPTIONS,
                       ARRAY,NDIM,NDIM,CNTRLO,CNTRHI,NCNTR,
                       WKAREA,LDIM,ELEMENT)
(PL/1)     CALL UGCNTR(OPTIONS,POPTIONS,SOPTIONS,TOPTIONS,
                       ARRAY,CNTRLO,CNTRHI,NCNTR,
                       WKAREA,ELEMENT);
```

The parameters in the calling sequence are:

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

- NOLABL This item will suppress the labeling of the primary contours where they cross the boundary.
- LWID=<value> The field width of the labels. The default value is 8. For the FORTRAN version of this subroutine, LWID must satisfy 1≤LWID≤12.
- LDEC=<value> The number of places to the right of the decimal point in the labels. The default value is 2. If LDEC has the value zero, then no decimal point will appear in

the label. For the FORTRAN version of this subroutine, LDEC must satisfy 0≤LDEC≤10.

XOFFR=<value> The offset, in the X direction, of the first of the LWID characters in the label from the point where a primary contour intersects the right hand boundary. The default value is -0.015.

XOFFL=<value> The offset, in the X direction, of the first of the LWID characters in the label from the point where a primary contour intersects the left hand boundary. The default value is -0.120.

XOFFV=<value> The offset, in the X direction, of the first of the LWID characters in the label from the point where a primary contour intersects the top or bottom boundary. The default value is -0.060.

YOFFV=<value> The offset, in the Y direction, of the first of the LWID characters in the label from the point where a primary contour intersects the top or bottom boundary. The default value is 0.020.

NSCL=<value> The number of secondary contour lines between the primary contour lines. The default value is 0.

TOLER=<value> An internal tolerance which may have to be adjusted if the X, Y, and Z coordinates take on large values. The default value is 0.0001.

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

POPTIONS An options list that will be passed to UGELIN when drawing the primary contours.

SOPTIONS An options list that will be passed to UGELIN when drawing the secondary contours.

TOPTIONS An options list that will be passed to UGETXT.

ARRAY A floating point array which contains the X, Y, and Z coordinates of the points on the surface. The format of ARRAY is:

```
--  X1   X2  ... XN
Y1  Z11  Z21 ... ZN1
Y2  Z12  Z22 ... ZN2
...          ...
YM  Z1M  Z2M ... ZNM
```

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YM) must be monotonically increasing but do not have to be equally spaced.

MDIM The first dimension of ARRAY; that is, M+1.

NDIM The second dimension of ARRAY; that is, N+1.

CNTRLO A floating point value giving the Z value of the lowest primary contour.

CNTRHI A floating point value giving the Z value of the highest primary contour.

NCNTR The number of primary contours.

WKAREA A full word, fixed point array which will be used as a work area. The amount of space that is needed in this

array depends on the dimension of the array ARRAY. A dimension of NDIM*NDIM/15 will be more than sufficient in most cases.

LDIM The dimension of WKAREA.

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

The errors detected by this subroutine are:

1(3): The dimension of the rectangular array ARRAY must be at least 3 by 3 to define a valid surface.

2(3): The array WKAREA is not large enough.

3(4): There is something substantially wrong with the definition of the surface.

9(1): Not enough room is available in the array ELEMENT to contain the description of the contour plot. The graphic element is unchanged.

The solution to this problem is also an approximation in that a unique surface is not defined by the rectangular array. A sufficient requirement to produce good contour plots is to have a fine enough mesh that no more than one contour line goes through each rectangular patch in the neighborhood of a saddle point of the surface. The example at the beginning of this section illustrates a problem that most contour plot algorithms have. The center of the contour plot is a three-way saddle point, the so-called "monkey saddle", and all of the zero level lines should come together at a point.

The information given with subroutine UGAXLS about labels and programmer coordinate systems other than the default also applies here. A non-default programmer coordinate system will usually mean that XOFFR, XOFFL, XOFFY, and YOFFV will have to be changed in addition to changing the character size with TOPTIONS.

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. Subroutines UGDINF and UGRINF may be used to obtain information about a display device at execution time. Finally, subroutine UGODEV can determine the identifications of all graphic devices which are currently open.

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. Normally a program should open a graphic device exactly

once and this should be done at the beginning of the program.

The calling sequences are:

```
(FORTRAN)  CALL UGOPEN(OPTIONS,IDENT)
(PL/1)     CALL UGOPEN(OPTIONS,IDENT);
```

The parameters in the calling sequences are:

OPTIONS A character string containing the graphic device type and any device-dependent information which may be required. The device types which select one of the non-interactive graphic devices are:

- CALDRSM The CALCOMP Drum Plotter (Model 1136) with the small size (10 inch) paper.
- CALDRLG The CALCOMP Drum Plotter (Model 1136) with the large size (33 inch) paper.
- CAL16MU The CALCOMP Microfilm Recorder (Model 1675) with 16mm unsprocketted film.
- CALFICH The CALCOMP Microfilm Recorder (Model 1675) with Microfiche film.
- PDS4013 Display Files for the TEKTRONIX 4013 are saved in a partitioned data set.
- SEQ4013 Display Files for the TEKTRONIX 4013 are saved in a sequential data set.
- PDS300Q Display Files for the GEN-COM 300-Q are saved in a partitioned data set.
- SEQ300Q Display Files for the GEN-COM 300-Q are saved in a sequential data set.
- VEP12FF The VERSATEC Electrostatic Plotter (Model 1200A) with fan-fold paper.
- VEP12CR The VERSATEC Electrostatic Plotter (Model 1200A) with continuous roll paper.
- PDSPDEV A pseudo-device which allows graphic elements in their device-independent form to be written to or read from a partitioned data set.
- SEQPDEV A pseudo-device which allows pictures to be saved in a device-independent form in a sequential data set.

The device types which select one of the interactive graphic devices are:

- IBM2250 The IBM 2250 Display Console.
- TEK4013 Interactive use of the TEKTRONIX 4013 Display Terminal under the MILTEN Terminal Manager.
- SLACXTV The SLAC Experimental Display Console.

The device types which select some non-standard or unusual graphic devices are:

- VEP1100 The VERSATEC Electrostatic Plotter (Model 1100A) located at the 40" Bubble Chamber.
- VEP1200 The VERSATEC Electrostatic Plotter (Model 1200A) located at LASS.
- TEM4013 Display files are created which the user may transmit to a TEKTRONIX 4013.
- SLACXSS Display files are created which the user may transmit to a SLAC Slave Scope.

The options list may also contain a number of other

items. Some of the more common items are:

DDNAME=<value> On most devices this item can be used to change the DD name of the necessary JCL card from the default value of UGDEVICE to the given value.

FULSCR This item makes the full plotting area available on devices with rectangular screens.

TITLE=<value> The output on some non-interactive devices is identified by a title consisting of the job name and his number. This item can be used to supply an alternate title which may consist of a character string of one to 24 characters. The character stroke generator that will be used for the titles is the one in effect when UGOPEN is called.

PICTID=<value> This item and PICTSQ are used on devices which save their pictures in a partitioned data set to construct the member name. This item supplies the first four characters of the member name. The default value is PICT.

PICTSQ=<value> This item is used with PICTSQ to complete the member name. It supplies an integer which is converted to three digits and concatenated to the PICTID value. The value is incremented after each picture and has a default value of one.

MAXENUM=<value> On interactive devices which allow element manipulation, this item gives the maximum number of elements which may be present in a single display. The default value is 32.

There are many additional items which may appear in the options list but most of these are very device-dependent. They are described in the sections devoted to the specific device.

IDENT A numeric value which is used to identify the graphic device being opened. If a program uses only one graphic device, then the value of this parameter is not important; it may be any non-zero value. However, if multiple graphic devices are being used, each must have a distinct identification.

The errors detected by this subroutine are:

1(3): Invalid or duplicate identification. The IDENT parameter must have a non-zero value. If you are trying to open more than one graphic device, then each device must have a unique identification.

2(3): Too many graphic devices. Only eight can be open at once.

3(3): A valid graphic device was not specified. Check your OPTIONS parameter.

11(3): The graphic device cannot be opened. Check your DD cards.

12(3): Not enough display buffer space is available. You

will probably have to reduce your MAXDFIL options item.

This subroutine can also ABEND with an 806 System Completion Code. This is usually caused by the library with the run time modules not being available; check your JCL.

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. Either the active device or all devices may be closed. Closing is an important operation for most graphic devices. Among the things that may happen are: (1) the last picture is completed, (2) the final buffer of information is written out, and (3) memory occupied by some of the internal control blocks is released. Normally a program should close a graphic device exactly once and this should be done at the end of the program.

The calling sequences are:

```
(FORTRAN)  CALL UGCLOS(OPTIONS)
(PL/1)     CALL UGCLOS(OPTIONS);
```

The parameter in the calling sequences is:

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

ALL This item indicates that all of the open graphic devices are to be closed. Normally only the active device is closed.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

2(4): A Unified Graphics control block (the MCA) has been destroyed by your program.

SECTION 2.2.3: SUBROUTINE UGSLCT

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 being used by the program.

The calling sequences are:

```
(FORTRAN)  CALL UGSLCT(OPTIONS,IDENT)
(PL/1)     CALL UGSLCT(OPTIONS,IDENT);
```

The parameters in the calling sequences are:

OPTIONS This parameter is present for consistency with the other calling sequences; no items will be recognized.

IDENT The identification of the device which is to become the active graphic device.

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

SECTION 2.2.4: SUBROUTINE UGDINF

This subroutine may be used to determine the active device type, its identification, and many of its properties and capabilities. With each call to this subroutine, the programmer can ask a single question (or a small number of questions if the answers do not conflict) about the active graphic device.

The calling sequences are:

```
(FORTRAN)  CALL UGDINF(OPTIONS,STRING,IVALUE,XVALUE)
(PL/1)     CALL UGDINF(OPTIONS,STRING,IVALUE,XVALUE);
```

THE PARAMETERS IN THE CALLING SEQUENCE ARE:

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

- DEVTYP This item causes STRING to be set to the device type as listed under UGOPEN.
- DEVID The parameter IVALUE will be set to the device identification.
- QVSML,QSMAL,QLARG,QVLRG Tests if the given character size is supported by the device and sets STRING to 'YES ' if it is and 'NO ' if it is not supported.
- QSOLD,QDASH,QDDSH,QDOTS Tests if the given line structure is supported on the device and sets STRING.
- QVDIM,QDIMM,QBRIT,QVBRT Tests if the given intensity is supported by the device and sets STRING.
- QCOL1,QCOL2,QCOL3,QCOL4 Tests if the given color is supported by the device and sets STRING.
- QSTDY,QWINK Tests if the given mode is supported by the device and sets STRING.
- QPXUP,QPXRIGHT,QPXDOWN,QPXLEFT Tests if the picture on the device can be extended without limit in the given direction and sets STRING. A drum plotter, for instance, has a picture which can be extended without limit on the right.
- QPTHARD,QPTREFR,QPTSTOR Tests the picture type, that is, the type of drawing surface. QPTHARD will return a yes answer if the device is a non-interactive device, QPTREFR returns a yes if the device has a refresh CRT, and QPTSTOR returns a yes if the device has a storage CRT.
- QKBRD,QLPEN,QSPTR Tests if the device has a keyboard, light pen, or screen pointer and sets STRING.
- ZVSML,ZSMAL,ZLARG,ZVLRG Sets XVALUE to the

actual character spacing that will be used when the given character size is requested. The result is based on the default scaling and scissoring limits.

ZSCRN Sets XVALUE to the screen size of the device in centimeters. The size is the length of the side of the square area based on the default scaling and scissoring limits.

STRING A character string of 8 characters. In addition to the YES-NO answers described above, this parameter will be set to 'NODEVICE' if no graphic device is active.

IVALUE A fixed point parameter that will be set as described above.

XVALUE A floating point parameter that will be set as described above.

No error messages are produced by this subroutine.

SECTION 2.2.5: SUBROUTINE UGRINF

This subroutine allows the programmer to obtain device-dependent run-time information about the active graphic device. An array of integer values associated with each device may be obtained or reset. The meaning of these values is given in the sections covering the individual graphic devices.

The calling sequences are:

```
(FORTRAN)  CALL UGRINF(OPTIONS,IARRAY)
(PL/1)     CALL UGRINF(OPTIONS,IARRAY);
```

The parameters in the calling sequences are:

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

GET,PUT Flags to indicate if the information is to be retrieved or set to the given values.

IARRAY A fixed point array of dimension 8 which will either be set to the current device information or replace the current device information.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

SECTION 2.2.6: SUBROUTINE UGODEV

This subroutine will supply a count of the number of graphic devices which are currently open and a list of the identifications of the open devices.

The calling sequences are:

```
(FORTRAN)  CALL UGODEV(OPTIONS,NDEV,IARRAY)
(PL/1)     CALL UGODEV(OPTIONS,NDEV,IARRAY);
```

The parameters in the calling sequences are:

OPTIONS	This parameter is present for consistency with the other calling sequences; no items will be recognized.
NDEV	A fixed point parameter that will be set to the number of graphic devices which are currently open.
IARRAY	A fixed point array of dimension 8 which will have the identifications of the open devices inserted into it.

No error messages are produced by this subroutine.

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 UGEPUT will transmit a graphic element to a display device. Subroutine UGPICT gives the programmer control over the picture. Subroutine UGCTRL allows the programmer to control the display device itself. Finally, subroutine UGEGET 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 "omit" state (as opposed to the "include" state).

SECTION 2.3.1: SUBROUTINE UGEPUT

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. It is when UGEPUT is called that the system

makes use of the current scaling and scissoring limits and the current character stroke generator.

The calling sequences are:

```
(FORTRAN)  CALL UGEPUT(OPTIONS,IDENT,ELEMENT)
(PL/1)     CALL UGEPUT(OPTIONS,IDENT,ELEMENT);
```

The parameters in the calling sequences are:

OPTIONS A character string which may contain any of the following items:
- DETC Make the element light pen detectable.
- OMIT Leave the element in the omit state.
- NOOPT Do not optimize the orders for the device. Normally the system trys to generate the minimum number of display orders to produce the picture.
- ON,OFF Leave the display in the "on" or "off" state.

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 non-zero identification will cause the second to replace the first.

ELEMENT The graphic element which is to be transmitted.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

11(2): The graphic element is too large for the device-dependent element buffer. You will probably have to increase the size of the MAXESIZ options item in the call to UGOPEN.

12(2): There is not enough space in the display file for this graphic element. You will either have to increase the size of the MAXDFIL options item in the call to UGOPEN, or you will have to simplify the pictures you are trying to display.

13(2): The maximum graphic element count has been exceeded. You will probably have to increase the size of the MAXENUM options item in the call to UGOPEN.

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

This subroutine sometimes generates underflows. This is caused by a user programming error; a subroutine such as UGELIN has been called with a fixed point value for an X or Y coordinate instead of a floating point value.

SECTION 2.3.2: SUBROUTINE UGPICT

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 sequences are:

```
(FORTRAN)  CALL UGPICT(OPTIONS,IDENT)
(PL/1)     CALL UGPICT(OPTIONS,IDENT);
```

The parameters in the calling sequences are:

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

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

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

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

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

The most common use of this subroutine is to signal the beginning of a new picture. The call:

```
      CALL UGPICT('CLEAR*',0)
```

in FORTRAN, or:

```
      CALL UGPICT('CLEAR',0);
```

in PL/1 performs this operation. On a device like the CALCOMP Drum Plotter, these calls cause the pen to move over to a clean piece of paper; on an IBM 2250-like device, the screen is cleared by re-initializing the display buffer. Although this call should be made before each picture, it is not actually necessary to do it before the first picture; UGOPEN always leaves the graphic device with a clean drawing area. Calling UGPICT in this manner before the first picture will not generate a blank picture.

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 sequences are:

```
(FORTRAN)  CALL UGCTRL(OPTIONS,IARRAY,XARRAY)
(PL/1)     CALL UGCTRL(OPTIONS,IODATA);
```

The parameters in the calling sequences are:

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

BEEP Sound the audible alarm on the display device.

RDSPTR Read the current position of the screen pointer. The X and Y coordinates of the current position of the screen pointer in the programmer coordinate system will be put into XARRAY(1) and XARRAY(2); the coordinates in the device coordinate system are put into IARRAY(1) and IARRAY(2). If the screen pointer is not on the screen, then IARRAY(1) and IARRAY(2) are set to minus one.

IARRAY A fixed point input and output array of dimension 8.

XARRAY A floating point input and output array of dimension 2.

IODATA An input and output structure whose declaration is:

```
DECLARE 1 IODATA,
          2 IARRAY(8) FIXED BINARY(15,0),
          2 XARRAY(2) FLOAT BINARY(21);
```

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

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

SECTION 2.3.4: SUBROUTINE UGEGET

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

The calling sequences are:

```
(FORTRAN)  CALL UGEGET(OPTIONS,ELEMENT,NELEM)
(PL/1)     CALL UGEGET(OPTIONS,ELEMENT);
```

The parameters in the calling sequences are:

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

NAME=<value> When this item is given, the element that will be read is the first element in the partitioned data set member with the given name. If this item is not given, the element that will be read is the next element in the partitioned data set member that was previously accessed.

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

NELEM The dimension of ELEMENT.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

11(1): End of Picture. No more graphic elements are available in the partitioned data set member.

12(3): No partitioned data set member with the given name is available.

13(3): The graphic element that you are trying to read is larger than the array ELEMENT.

99(2): This subroutine is not supported for the active graphic 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 UGEATN may be used to enable attentions from a control unit (such as the keyboard or light pen) on a display device. Subroutine UGDATN may be used to disable attentions from a control unit. When an attention is generated by an enabled control unit, it is saved on an attention queue; an attention from a disabled unit is not saved. Finally, subroutine UGRATN may be used to check to see if an attention has occurred and optionally wait until one does occur.

Attentions are enabled or disabled by specifying a four-character attention code in the options list of 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 a character string, ATCODE, an integer array, IARRAY, and a floating point array, XARRAY.

The following list gives (1) the valid attention codes, (2) a numeric code for the attention, and (3) a description of the attention:

NONE 0 This is not an actual attention code but is a possible response from UGRATN when no attention is available to report and an indefinite wait is not requested. This response is also possible when a program is using multiple interactive devices, is waiting on one device, and an attention is generated on another device.

KBRD 1 An attention generated by a special key on the keyboard which is reserved for this use. The actual key that is used to generate this attention varies from device to device.

LPEN 2 An attention associated with a light pen device. Devices of this type may be used to point at parts of the display. The identification of the element being pointed at is normally part of the information made available when this attention is reported. For some devices, the X and Y

coordinates of the item pointed at is also available.

SPTR 3 An attention associated with a screen pointer device. Devices of this type may be used to point at an arbitrary position on the screen. The coordinates of the spot being pointed at is normally part of the information made available when this attention is reported.

The programmer may assume that all interactive devices contain a keyboard and can generate the KBRD attention. However, this keyboard is about the only thing that two interactive devices are sure to have in common. To find out exactly what control units are available on a specific device, the user should refer to the sections devoted to the specific device.

The arrays IARRAY and XARRAY in the calling sequence of UGRATN are used to supply additional information about the attention. The elements of IARRAY are reserved for the following information:

IARRAY(1) The numeric code for the attention as described above.
IARRAY(2) The identification of an element.
IARRAY(3) A byte offset in an element. This information is always device-dependent.
IARRAY(4) An X coordinate in the device coordinate system. This information is device-dependent.
IARRAY(5) A Y coordinate in the device coordinate system.
IARRAY(6) A button or key number.
IARRAY(7) Not assigned.
IARRAY(8) Not assigned.

The elements of XARRAY are reserved for the following information:

XARRAY(1) An X coordinate in the programmer coordinate system. The scaling and scissoring limits that are used to compute this value are the ones in effect when the attention is reported and <u>not</u> the ones in effect when it was generated.
XARRAY(2) A Y coordinate in the programmer coordinate system.

To find out exactly which items of information are made available on a specific device, the user is again referred to the sections devoted to the specific device.

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 UGEATN 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 UGEATN

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 sequences are:
(FORTRAN) CALL UGEATN(OPTIONS)
(PL/1) CALL UGEATN(OPTIONS);

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

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

SECTION 2.4.2: SUBROUTINE UGDATN

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

The calling sequences are:
(FORTRAN) CALL UGDATN(OPTIONS)
(PL/1) CALL UGDATN(OPTIONS);

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

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

SECTION 2.4.3: SUBROUTINE UGRATN

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) UGRATN 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. If the attention queue is not empty, the programmer may (1) obtain only the oldest queued attention without disturbing any other queued attentions, or, (2) ask for only the latest attention and discard all of the others.

The calling sequences are:
(FORTRAN) CALL UGRATN(OPTIONS,TIME,ATCODE,IARRAY,XARRAY)
(PL/1) CALL UGRATN(OPTIONS,TIME,ATTND);

The parameters in the calling sequences are:
OPTIONS A character string which may contain any of the

following items:
LAST This item indicates that only the last queued attention is wanted and any other attentions on the queue should be discarded.

TIME A floating point variable where 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. The FORTRAN programmer should not use positive literal values in the calling sequence because that will cause the constant to be changed in the program.

ATCODE A character string of length 4 which will be set to the attention type.

IARRAY A fixed point array of dimension 8 which will be set to contain attention data.

XARRAY A floating point array of dimension 2 which will be set to contain attention data.

ATTND A structure whose declaration is:

```
DECLARE 1 ATTND,
          2 ATCODE CHARACTER(4),
          2 IARRAY(8) FIXED BINARY(15,0),
          2 XARRAY(2) FLOAT BINARY(21);
```

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

99(3): This subroutine is not supported for the active graphic device.

SECTION 2.5: KEYBOARD INPUT BUFFER CONTROL

A "Keyboard Input Buffer" is a special kind of graphic element consisting of text 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 UGKPUT 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 add a keyboard input buffer to the display file.

The calling sequences are:

```
(FORTRAN)  CALL UGKPUT(OPTIONS,X,Y,IDENT,STRING,NSTRING)
(PL/1)     CALL UGKPUT(OPTIONS,X,Y,IDENT,STRING);
```

The parameters in the calling sequences are:

OPTIONS	A character string which may contain any of the following items:
	VSML,SMAL,LARG,VLRG Character size.
	VDIM,DIMM,BRIT,VBRT Intensity levels.
	COL1,COL2,COL3,COL4 Color options.
X	The floating point X coordinate of the center of the first character.
Y	The floating point Y coordinate of the center of the first character.
IDENT	The identification of the keyboard input buffer.
STRING	A character string which contains the initial values for the keyboard input buffer. This may be a string of blanks.
NSTRING	The number of characters in STRING.

The errors detected by this subroutine are:

1(3):	No graphic device is active at present.
11(2):	The parameter STRING contains more than 255 characters.
12(2):	There is not enough space in the display file for this keyboard input buffer. You will either have to increase the size of the MAXDFIL options item in the call to UGOPEN, or you will have to simplify the pictures you are trying to display.
13(2):	The maximum graphic element count has been exceeded. You will probably have to increase the size of the MAXENUM options item in the call to UGOPEN.
99(2):	This subroutine is not supported for the active graphic device.

On refresh display devices, the keyboard input buffer will be initialized to the characters in STRING; for storage CRT's, the characters will not be transmitted because this would force the console operator to type over the transmitted characters.

SECTION 2.5.2: SUBROUTINE UGKGET

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

The calling sequences are:

```
(FORTRAN)  CALL UGKGET(OPTIONS,IDENT,STRING,MSTRING,NSTRING,
                       ICUR);
(PL/1)     CALL UGKGET(OPTIONS,IDENT,STRING,ICUR);
```

The parameters in the calling sequences are:

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

LCASE This indicates that the character entered at the console are not to be translated to upper case characters.

IDENT The identification of the keyboard input buffer to be read.

STRING A character string which will have the keyboard input buffer placed in it. In PL/1 this character string must have the VARYING attribute.

MSTRING The number of characters available in STRING for storing the buffer.

NSTRING The number of characters actually stored in STRING.

ICUR If the keyboard input buffer being read contains the cursor, then this is the index of the character containing the cursor; otherwise it is zero.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

11(2): There is no keyboard input buffer with the given identification.

12(2): There is not enough room in the parameter STRING to contain the keyboard input buffer.

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

SECTION 2.6: SCALING AND SCISSORING

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 the rectangular area defined by the device coordinate system. 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.

Subroutine UGSCAL allows a programmer to manipulate the scaling and scissoring limits in a completely general way, while UGRSCL provides a device-independent means of generating rectangular pictures. The programmer and device coordinate system limits are specified by two 2 by 2 matrices with the following format;

```
┌              ┐        ┌              ┐
│ PXLO   PXHI  │        │ DXLO   DXHI  │
│ PYLO   PYHI  │        │ DYLO   DYHI  │
└              ┘        └              ┘
```

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.

In the horizontal direction, the number of device units per programmer unit is (DXHI-DXLO)/(PXHI-PXLO). In the vertical direction this ratio is (DYHI-DYLO)/(PYHI-PYLO). Under normal circumstances, the ratio of these two values should be the same as its default value. If this ratio is changed, the operation of drawing characters at an angle by the character stroke generator is ambiguous and the results may be different than what the programmer expected. Subroutine UGRSCL always preserves this ratio. The preservation of this ratio simply means that the number of programmer units per centimeter in the X direction is the same as the number of units per centimeter in the Y direction.

SECTION 2.6.1: SUBROUTINE UGSCAL

This subroutine may be used to retrieve or change the scaling and scissoring parameters of the active graphic device. The limits may also be reset to the values that they were given by UGOPEN. The default values for a device are given in the sections devoted to specific devices. The user should remember that the scaling and scissoring limits are not actually used until subroutine UGEPUT or UGETRN is called.

The calling sequences are:

```
(FORTRAN)  CALL UGSCAL(OPTIONS,PROGLN,DEVCLN)
(PL/1)     CALL UGSCAL(OPTIONS,PROGLN,DEVCLN);
```

The parameters in the calling sequences are:

OPTIONS A character string which may contain any of the following items:
- GET,PUT,RESET Flags to indicate if the limits are to be retrieved, changed, or reset to the values they had when UGOPEN was called.
- PROGL A flag to indicate that the programmer limits are to be retrieved or changed.
- DEVCL A flag to indicate that the device limits are to be retrieved or changed.

PROGLN A floating point array of dimension (2,2) for the programmer scaling limits.

DEVCLN A full word, integer array of dimension (2,2) for the device scaling limits. Note that in PL/1, this argument must be FIXED BINARY(31,0).

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

2(3): The new scaling and scissoring limits are not valid. The low X and Y limits must be less than the high X and Y limits.

When the arrays PROGLN or DEVCLN are initialized with FORTRAN data statements, the correct order is:

```
REAL*4        PROGLN(2,2)
DATA          PROGLN/PXLO,PYLO,PXHI,PYHI/
```

In PL/1 the correct order is:

```
      DECLARE PROGLM(2,2) INITIAL(PXLO,PXHI,PYLO,PYHI);
```

Setting the arrays at executation time is done with statements similar to:

```
      PROGLM(1,1)=PXLO
      PROGLM(1,2)=PXHI
      PROGLM(2,1)=PYLO
      PROGLM(2,2)=PYHI
```

SECTION 2.6.2: SUBROUTINE UGRSCL

This subroutine provides a device-independent way of generating rectangular pictures. To use this subroutine, the graphic device should be opened with the FULSCR option to make the full plotting area available. Then this subroutine may be called to establish scaling and scissoring limits for a rectangular picture. The subroutine may be called repeatedly to change the scaling and scissoring limits for each picture. Normally this subroutine works by calling UGSCAL to obtain the initial scaling and scissoring limits, calling UGDINF to see if the picture can be extended in any direction, and then using this data to produce the largest picture possible with the given dimensions. However, if the CURLIM options item is given, this subroutine will use the current scaling and scissoring limits instead of the initial values.

The calling sequences are:

```
  (FORTRAN)  CALL UGRSCL(OPTIONS,PROGLM)
  (PL/1)     CALL UGRSCL(OPTIONS,PROGLM);
```

The parameters in the calling sequences are:

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

CURLIM This flag indicates that the current scaling and scissoring limits are to be used instead of the initial values.

PROGLM A floating point array of dimension (2,2) containing the required programmer scaling limits. The only restrictions are that the upper limits in X and Y must be greater than the lower limits and that a unit distance in the X direction is the same as a unit distance in the Y direction.

No error messages are produced by this subroutine but UGSCAL or UGDINF may themselves issue error messages.

A few examples may clarify the way subroutines UGSCAL and UGRSCL may be used. For a first example, suppose it is desired to generate a rectangular picture 50% wider than it is high. Also, suppose that the programmer coordinate system is to run from 0.0 to 15.0 in the X direction and 0.0 to 10.0 in the Y direction. This is a simple application of UGRSCL and the FORTRAN code is the following:

```
      REAL*4        PLIM(2,2)
      PLIM(1,1)=0.0
```

```
      PLIM(1,2)=15.0
      PLIM(2,1)=0.0
      PLIM(2,2)=10.0
      CALL UGRSCL('*',PLIM)
```

On a device like the 10 inch CALCOMP Drum Plotter, the drawing area would be 15 inches along the paper by 10 inches across the paper. On a device like the IBM 2250, a rectangular area the full width of the screen would constitute the drawing area. The drawing area would be centered vertically and would not extend to the top or bottom of the screen.

As a second example, suppose the programmer must produce a picture of the same size, but with a programmer coordinate system running from 0.0 to 100.0 in the X direction. If the 15.0 is changed to 100.0 in the preceding example, the result will not be correct. On the CALCOMP, the picture would stretch 100 inches along the paper and on the IBM 2250, the picture would be only about an inch high. The problem is that the programmer units per inch in the X direction is different from that in the Y direction. The way to achieve the desired result is to proceed in two steps. First, UGRSCL is called to set up the programmer and device limits for a picture with the correct physical dimensions, and then the programmer limits are changed with UGSCAL. The FORTRAN code is:

```
      REAL*4          PLIM(2,2)
      INTEGER*4       DUMY(2,2)
      PLIM(1,1)=0.0
      PLIM(1,2)=15.0
      PLIM(2,1)=0.0
      PLIM(2,2)=10.0
      CALL UGRSCL('*',PLIM)
      PLIM(1,2)=100.0
      CALL UGSCAL('PUT,PROGL*',PLIM,DUMY)
```

Now consider a more complicated problem. Suppose the programmer is working with a graphic device with a square screen like the IBM 2250 and wants to divide the screen into two parts; the upper three-fourths for pictures and the bottom one-fourth for messages. The upper part of the screen is to have a programmer coordinate system with X running from -1.0 to 1.0, and Y running from 2.0 to 10.0. It is also important that the code will run on any similar graphic device. One way to do this is to use UGSCAL to read the device coordinate system, modify these values, and write them back with the new programmer coordinate system. The FORTRAN code to do this is:

```
      REAL*4          PLIM(2,2), DUMY(2,2)
      INTEGER*4       DLIM(2,2)
      CALL UGSCAL('GET,DEVCL*',DUMY,DLIM)
      PLIM(1,1)=-1.0
      PLIM(1,2)=1.0
      PLIM(2,1)=2.0
      PLIM(2,2)=10.0
      DLIM(2,1)=DLIM(2,1)+0.25*(DLIM(2,2)-DLIM(2,1))
      CALL UGSCAL('PUT,DEVCL,PROGL*',PLIM,DLIM)
```

These few examples should give some idea of how the scaling and scissoring limits may be manipulated in a device-independent

manner. One thing which should be avoided is to build the device coordinate system into a program. The third example illustrates a situation where the temptation to simply define DLIM by:

```
DLIM(1,1)=0
DLIM(1,2)=4095
DLIM(2,1)=1023
DLIM(2,2)=4095
```

and call UGSCAL to set the device coordinate system to these values is very strong. The problem, of course, is that a program that does this will only run correctly on an IBM 2250 or a device with an identical raster unit configuration. The example, however, shows a much more acceptable way of doing the same thing. Building the device coordinate system into a program can create a time bomb which will cause trouble later when it becomes desirable to use a different graphic device.

SECTION 2.6.3: SUBROUTINE UGSCIS

This subroutine may be used to turn scissoring on and off for the active device. Small amounts of computer time can be saved if scissoring of the picture can be turned off. However, if scissoring is turned off when it is actually needed, the results are unpredictable. The use of this subroutine is, therefore, not recommended, especially for programs which are not fully checked out. The user should also remember that the scissoring module is not actually used until subroutine UGEPUT or UGETRM is called.

The calling sequences are:

```
(FORTRAN)  CALL UGSCIS(OPTIONS)
(PL/1)     CALL UGSCIS(OPTIONS);
```

The parameter in the calling sequences is:

OPTIONS A character string which may contain any of the following items:
ON,OFF Turn scissoring on or off.

The errors detected by this subroutine are:

1(3): No graphic device is active at present.

SECTION 2.7: CHARACTER STROKE GENERATING

The Unified Graphics System contains a Character Stroke Generating Sub-system which is used to convert text data in a graphic element into short line segments (called "strokes") 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 options item 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 options item, but either the device has no character generator or it has one which cannot satisfy all of the requirements.

There is actually more than one stroke generator available in the system. The Basic generator, which is the one that gets used if the programmer does not request an alternate one with subroutine UGCSET, can process all of the characters on an IBM 029 keypunch, ten additional plotting symbols, and produces lower case letters as upper case. The characters produced by the Basic stroke generator are shown in Figure 2.7.1.

ABCDEFGHIJKLMNØPQRSTUVWXYZ
0123456789 +×◇□✧⌑⌖⌘⌗○
¢.<(+|&!$*);¬-/,%_>?:#@'=″

Figure 2.7.1: The Basic Character Set.

An Extended stroke generator is available which can produce all of the characters shown in Figure 2.7.2. These characters include upper and lower case Roman and Greek letters and a large group of special characters. In addition, some special characters are provided which cause the stroke generator to enter superscript or subscript mode. Superscripts or subscripts may themselves have superscripts or subscripts. This stroke generator differs from the Basic generator in one other important respect. The Basic stroke generator produces characters which all have the same width (their width is two-thirds of the value given by the SPACING or XSPACING option in UGETXT and their height is the same as that value). The Extended generator produces characters of differing widths and heights; thus the upper case letter "M" is twice as wide as the "I", and most lower case letters are three-fourths as wide as most upper case letters. This results in a more pleasing appearance, but also causes some problems; if, for example, 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 control character has been introduced which causes the stroke generator to save its current position and state. Another control character in a later part of the string can cause the earlier state of the stroke generator to be restored. There are three independent save-restore control character pairs available. The scope of these save-restore pairs is a single string supplied by UGETXT; that is, you cannot save a position in one call to UGETXT and try to use it in a later call. If you try to use a position without saving it in an earlier part of the string, the position you will obtain is that of the beginning of the string.

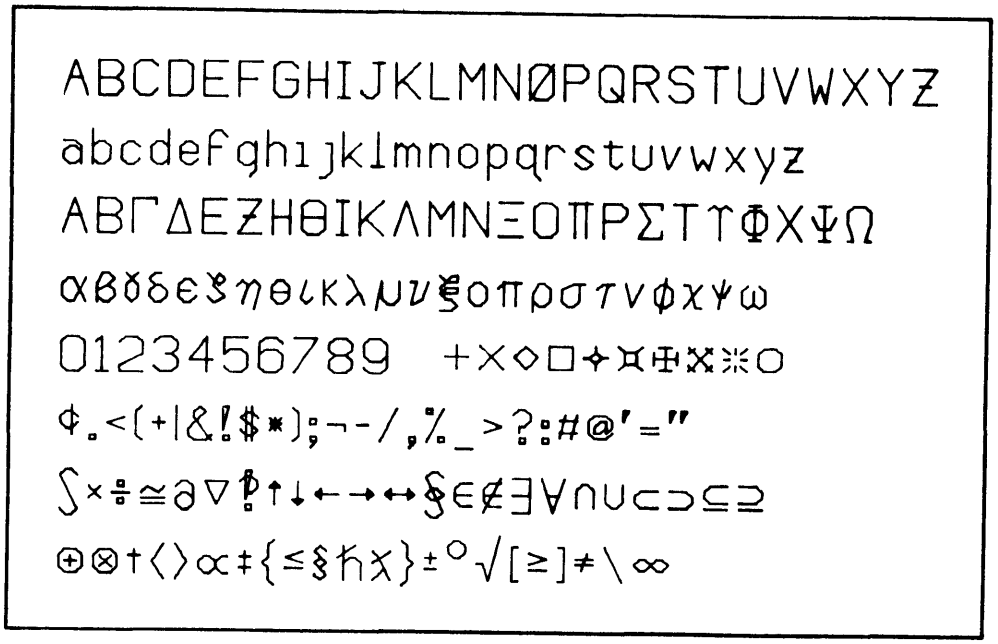


Figure 2.7.2: The Extended Character Set.

A third character generator, known as the Duplex generator, produces the characters shown in Figure 2.7.3. The character set is called the Duplex set because certain characters are partially traced over twice to make them more attractive. The majority of these characters were designed by A. V. Hershey and are described in [HER-1]. The Duplex character generator produces exactly the same character set as the Extended character generator; however, there are slight differences in the spacing and sizes of certain characters. While the Duplex character generator does produce very pleasing characters, the user should remember that there is a price to pay in execution time, memory usage, and speed of plotting; some of the characters are composed of more than 50 line segments.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

αβγδεζηθικλμνξοπρστυφχψω

0123456789 +×◇□✧⌑⌘⌧⌗○

¢.<(+|&!$*);¬−/,%_>?:#@'=''

∫×÷≅∂∇⍐↑↓←→↔∮∈∉∃∀∩∪⊂⊃⊆⊇

⊕⊗†⟨⟩∝‡{≦§ℏ⍱}±°√[≧]≠\∞

Figure 2.7.3: The Duplex Character Set.

The following table gives the hexadecimal equivalents of each of the characters in the extended character sets. The meaning and use of the character pairs will be described in the section on subroutine UGCHAR. For the moment, the reader should notice that there is a pattern to these characters; lower case Roman letters have "L" as the second character; upper and lower case Greek have "H" and "G" as the second character, respectively; control functions have a "C"; special move functions a "U" or "V"; plotting symbols a "P"; special characters an "A", "S", or "X"; and finally, the standard characters all have a second character of "n" which stands for a blank.

Hex. Code	Char Pair	Character Description	Hex. Code	Char Pair	Character Description
00	nS	Null	08	QG	L. C. Theta
01	AG	L. C. Alpha	09	IG	L. C. Iota
02	BG	L. C. Beta	0A	IS	Integral Sign
03	GG	L. C. Gamma	0B	XS	Times Sign
04	DG	L. C. Delta	0C	:S	Division Sign
05	EG	L. C. Epsilon	0D	ES	Approximately Equal
06	ZG	L. C. Zeta	0E	PS	Partial Derivative
07	HG	L. C. Eta	0F	DS	Del

Hex. Code	Char Pair	Character Description	Hex. Code	Char Pair	Character Description
10	?S	Interbang	18	RG	L. C. Rho
11	KG	L. C. Kappa	19	SG	L. C. Sigma
12	LG	L. C. Lambda	1A	UA	Up Arrow
13	MG	L. C. Mu	1B	DA	Down Arrow
14	NG	L. C. Nu	1C	LA	Left Arrow
15	XG	L. C. Xi	1D	RA	Right Arrow
16	OG	L. C. Omicron	1E	BA	Left/Right Arrow
17	PG	L. C. Pi	1F	JS	Line Integral
20	MX	Membership Symbol	28	YG	L. C. Psi
21	NX	Membership Negation	29	WG	L. C. Omega
22	EX	Existential Quant.	2A	IX	Intersection
23	AX	Universal Quant.	2B	UX	Union
24	TG	L. C. Tau	2C	<X	Contained in
25	UG	L. C. Upsilon	2D	>X	Contains
26	FG	L. C. Phi	2E	LX	Contained in/Equal
27	CG	L. C. Chi	2F	RX	Contains/Equal
30	0C	Enter Superscr. Mode	38	8C	Save Position-3
31	1C	Leave Superscr. Mode	39	9C	Restore Position-3
32	2C	Enter Subscr. Mode	3A		Unassigned
33	3C	Leave Subscr. Mode	3B		Unassigned
34	4C	Save Position-1	3C		Unassigned
35	5C	Restore Position-1	3D		Unassigned
36	6C	Save Position-2	3E	EC	Increase Size
37	7C	Restore Position-2	3F	FC	Decrease Size
40	¤¤	Blank	48	QH	U. C. Theta
41	AH	U. C. Alpha	49	IH	U. C. Iota
42	BH	U. C. Beta	4A	¢¤	Cent Sign
43	GH	U. C. Gamma	4B	.¤	Period
44	DH	U. C. Delta	4C	<¤	Less Than
45	EH	U. C. Epsilon	4D	(¤	Left Parenthesis
46	ZH	U. C. Zeta	4E	+¤	Plus Sign
47	HH	U. C. Eta	4F	|¤	Vertical Line
50	&¤	Ampersand	58	RH	U. C. Rho
51	KH	U. C. Kappa	59	SH	U. C. Sigma
52	LH	U. C. Lambda	5A	!¤	Exclamation Mark
53	MH	U. C. Mu	5B	$¤	Dollar Sign
54	NH	U. C. Nu	5C	*¤	Asterisk
55	XH	U. C. Xi	5D	)¤	Right Parenthesis
56	OH	U. C. Omicron	5E	;¤	Semi-Colon
57	PH	U. C. Pi	5F	¬¤	Not Sign
60	-¤	Minus Sign	68	YH	U. C. Psi
61	/¤	Slash Mark	69	WH	U. C. Omega
62	+S	Group Plus	6A	GS	Dagger
63	*S	Group Multiply	6B	,¤	Comma
64	TH	U. C. Tau	6C	%¤	Percent
65	UH	U. C. Upsilon	6D	_¤	Underline
66	FH	U. C. Phi	6E	>¤	Greater Than
67	CH	U. C. Chi	6F	?¤	Question Mark

Hex. Code	Char Pair	Character Description	Hex. Code	Char Pair	Character Description
70		Unassigned	78	LS	Left Angle Bracket
71		Unassigned	79	RS	Right Angle bracket
72		Unassigned	7A	:n	Colon
73		Unassigned	7B	#n	Pound Sign
74		Unassigned	7C	@n	At Sign
75		Unassigned	7D	'n	Apostrophe
76		Unassigned	7E	=n	Equals Sign
77		Unassigned	7F	"n	Quote Marks
80	QS	Proportional to	88	HL	L. C. H
81	AL	L. C. A	89	IL	L. C. I
82	BL	L. C. B	8A	FS	Double Dagger
83	CL	L. C. C	8B	AS	Left Brace
84	DL	L. C. D	8C	<S	Less or Equal
85	EL	L. C. E	8D	NS	New Paragraph
86	FL	L. C. F	8E	HS	H-Bar
87	GL	L. C. G	8F	WS	Lambda-Bar
90		Unassigned	98	QL	L. C. Q
91	JL	L. C. J	99	RL	L. C. R
92	KL	L. C. K	9A		Unassigned
93	LL	L. C. L	9B	ZS	Right Brace
94	ML	L. C. M	9C		Unassigned
95	NL	L. C. N	9D		Unassigned
96	OL	L. C. O	9E	SS	Plus or Minus
97	PL	L. C. P	9F		Unassigned
A0		Unassigned	A8	YL	L. C. Y
A1	OS	Degrees	A9	ZL	L. C. Z
A2	SL	L. C. S	AA	2S	Square Root
A3	TL	L. C. T	AB	US	Under Score
A4	UL	L. C. U	AC	VS	Over Score
A5	VL	L. C. V	AD	(S	Left Bracket
A6	WL	L. C. W	AE	>S	Greater or Equal
A7	XL	L. C. X	AF		Unassigned
B0	0P	Plot Sym. V. Cross	B8	8P	Plot Sym. Star
B1	1P	Plot Sym. D. Cross	B9	9P	Plot Sym. Octagon
B2	2P	Plot Sym. Diamond	BA		Unassigned
B3	3P	Plot Sym. Square	BB		Unassigned
B4	4P	Plot Sym. F. Diamond	BC		Unassigned
B5	5P	Plot Sym. F. Square	BD	)S	Right Bracket
B6	6P	Plot Sym. F. V. Cross	BE	=S	Not Equal
B7	7P	Plot Sym. F. D. Cross	BF		Unassigned
C0		Unassigned	C8	Hn	U. C. H
C1	An	U. C. A	C9	In	U. C. I
C2	Bn	U. C. B	CA	1V	Half Up
C3	Cn	U. C. C	CB	2V	Half Down
C4	Dn	U. C. D	CC	3V	Third Up
C5	En	U. C. E	CD	4V	Third Down
C6	Fn	U. C. F	CE	5V	Sixth Up
C7	Gn	U. C. G	CF	6V	Sixth Down

Hex. Code	Char Pair	Character Description	Hex. Code	Char Pair	Character Description
D0	0U	Backwards Blank	D8	Q¤	U. C. Q
D1	J¤	U. C. J	D9	R¤	U. C. R
D2	K¤	U. C. K	DA	1U	Extra Half Blank
D3	L¤	U. C. L	DB	2U	Half Back Blank
D4	M¤	U. C. M	DC	3U	Extra Third Blank
D5	N¤	U. C. N	DD	4U	Third Back Blank
D6	O¤	U. C. O	DE	5U	Extra Sixth Blank
D7	P¤	U. C. P	DF	6U	Sixth Back Blank
E0	/S	Backwards Slash	E8	Y¤	U. C. Y
E1		Unassigned	E9	Z¤	U. C. Z
E2	S¤	U. C. S	EA		Unassigned
E3	T¤	U. C. T	EB		Unassigned
E4	U¤	U. C. U	EC		Unassigned
E5	V¤	U. C. V	ED		Unassigned
E6	W¤	U. C. W	EE		Unassigned
E7	X¤	U. C. X	EF		Unassigned
F0	0¤	Numeral 0	F8	8¤	Numeral 8
F1	1¤	Numeral 1	F9	9¤	Numeral 9
F2	2¤	Numeral 2	FA		Unassigned
F3	3¤	Numeral 3	FB		Unassigned
F4	4¤	Numeral 4	FC		Unassigned
F5	5¤	Numeral 5	FD		Unassigned
F6	6¤	Numeral 6	FE		Unassigned
F7	7¤	Numeral 7	FF	0S	Infinity

There are two special characters shown in the above table which do not appear in Figures 2.7.2 and 2.7.3. These characters are Under Score (Hex AB) and Over Score (Hex AC). The purpose of these characters is to allow the programmer to draw lines under or over a line of text. Two consecutive Under Score characters, for example, will join together into a single line (this is not true of the Underline character). Thus the programmer, with some difficulty, can generate such things as fractions. The Over Score will also join properly with the Square Root character to form a full radical sign.

SECTION 2.7.1: SUBROUTINE UGCSET

The basic purpose of this subroutine is to select a character stroke generator for later use. The user should remember that the character stroke generator is not actually used until subroutine UGEPUT, UGETRN, or UGCTOL is called.

The calling sequences are:

```
(FORTRAN)  CALL UGCSET(OPTIONS)
(PL/1)     CALL UGCSET(OPTIONS);
```

The parameter in the calling sequences is:

OPTIONS A character string which may contain any of the

following items:

BASIC Selects the Basic character stroke generator.

EXTENDED Selects the Extended character stroke generator.

DUPLEX Selects the Duplex character stroke generator.

DELETE This item causes work space for the character stroke generator to be freed and the stroke generating code itself to be deleted from memory. This item is not normally needed but it can be used to advantage when memory space is scarce.

LOAD This causes work space for the character stroke generator to be allocated and the stroke generating code itself to be brought into memory. Like DELETE, this item is not normally needed.

No error messages are produced by this subroutine.

SECTION 2.7.2: SUBROUTINE UGCHAR

This subroutine may be used to transform between character pairs and the extended characters. The subroutine provides a means of generating the characters for the non-EBCDIC characters without having to deal with their hexadecimal representations.

The calling sequences are:

```
(FORTRAN)  CALL UGCHAR(OPTIONS,CPRI,CSEC,CEXT,NCHAR)
(PL/1)     CALL UGCHAR(OPTIONS,CPRI,CSEC,CEXT);
```

The parameters in the calling sequences are:

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

PTOX This item indicates that a character pairs to extended character transformation is required.

XTOP This item indicates that an extended character to character pairs transformation is required.

CPRI A character string for the primary characters.

CSEC A character string for the secondary characters.

CEXT A character string for the extended characters.

NCHAR The number of characters in the character strings.

No error messages are produced by this subroutine.

A simple example may clarify the use of this subroutine. Suppose that it is required to have the character for a lower case Greek Delta in the third position of a character string. The preceding table shows that the character pair for a lower case Greek Delta is "DG". Therefore, if subroutine UGCHAR is called with the PTOX option, a "D" in the third position of CPRI, and a "G" in the third position of CSEC, then the character for a lower case Greek

Delta will be inserted as the third character of CEXT. Figures 2.7.4 and 2.7.5 give additional examples of the types of displays which can be generated using the character pair scheme with the Extended and Duplex stroke generators.

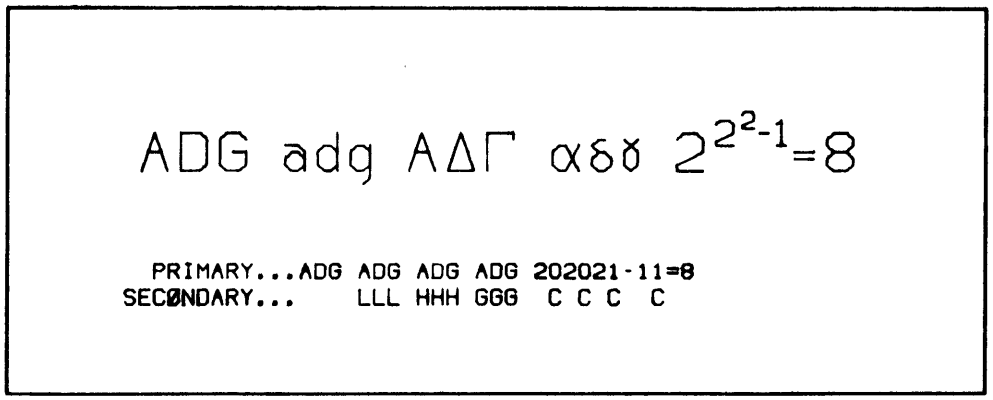


Figure 2.7.4: Some Simple Examples.

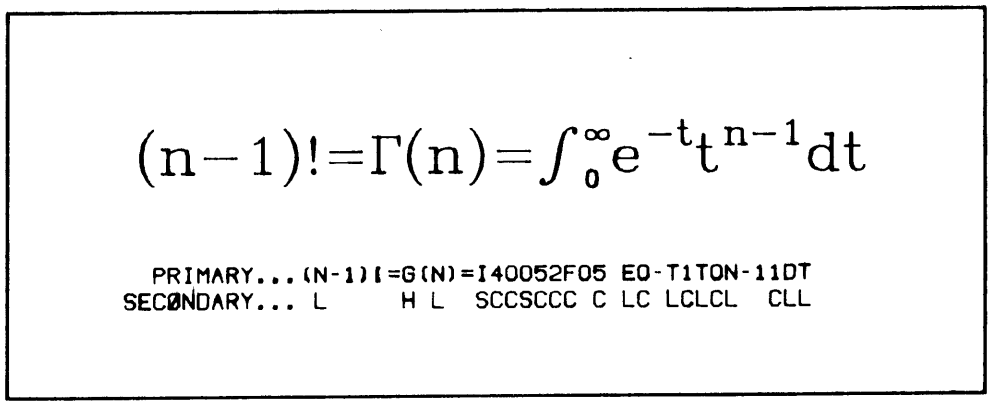


Figure 2.7.5: A More Elaborate Example.

SECTION 2.7.3: 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 the same as that required by subroutine UGELNS. Before passing these line segments on to UGELNS, the programmer may transform them in any manner. One simple thing a programmer could do is create italic lettering. Another use of this subroutine could be the creation of perspective views of a letter in 3-Space.

The calling sequences are:

```
(FORTRAN)   CALL UGCTOL(OPTIONS,TEXT,NTEXT,NSIZE,XARRAY,YARRAY,
                        NPTS,BBITS)
(PL/1)      CALL UGCTOL(OPTIONS,TEXT,XARRAY,YARRAY,NPTS,BBITS);
```

The parameters in the calling sequences are:

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

- X=<value> The X coordinate of the center of the first character. The default value is 0.0.
- Y=<value> The Y coordinate of the center of the first character. The default value is 0.0.
- SPACING=<value> The character spacing value which will be used to develop the strokes. The default value is 0.015.
- 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.

NSIZE The number of entries available in XARRAY, YARRAY, and BBITS. The blanking bits are stored in packed form.

XARRAY A floating point array where the X coordinates of the end points of the line segments will be stored.

YARRAY A floating point array where the Y coordinates of the end points of the line segments will be stored.

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

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

The errors detected by this subroutine are:

1(2): The parameter TEXT contains more than 255 characters.

2(2): There is not enough space available in the parameters XARRAY, YARRAY, and BBITS to contain the generated line segment data.

SECTION 2.8: ERROR PROCESSING

The Unified Graphics System contains a flexible scheme for reporting errors to the programmer. When a subroutine detects an error, it supplies three pieces of information to the error processor. This information consists of (1) the severity level of the error, (2) the name of the subroutine detecting the error, and (3) the index of the error. The description of each subroutine in this document includes a list of all of the error conditions, the index of the error, and its severity level (in parentheses).

There are four severity levels for the errors. These levels and the default actions of the error processor are:

1. Minor Errors: The severity level, subroutine name, and index are saved in some error indicators.
2. Errors: The error information is saved in the error indicators and an error message is printed. The error message contains the name of the subroutine detecting the error, the index of the error, the level of the error, and a description of the error.

3. Severe Errors: The error message is printed and the program is terminated without a memory dump.
4. Terminal Errors: The error message is printed and the program is terminated with a memory dump.

At execution time, the programmer can check for Minor Errors and Errors and try to recover from them. This can be done by checking the error indicators after a subroutine has been called to see if anything has been put into them. In FORTRAN, the error indicators are declared by:

```
      COMMON        /UGERRD/UGELVL,UGENAM(2),UGEIND
      INTEGER*4     UGELVL,UGENAM,UGEIND
```

For PL/1 the declaration of the error indicators is:

```
      DECLARE 1 UGERRD STATIC EXTERNAL,
                2 UGELVL FIXED BINARY(15,0),
                2 UGENAM CHARACTER(8),
                2 UGEIND FIXED BINARY(15,0);
```

When a Minor Error or Error has occurred, UGELVL will contain the severity level, UGENAM will contain the subroutine name (padded on the right with blanks) and UGEIND will contain the index. If no error has occurred, UGELVL and UGEIND will contain zero and UGENAM will contain blanks.

A second and more versatile way of processing errors is by means of a user written error processing subroutine named UGXERR. If a user includes a subroutine with this name, it will be called whenever an error is detected. The skeleton for this subroutine in FORTRAN is:

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

The skeleton for this subroutine in PL/1 is:

```
      UGXERR: PROCEDURE(LEVEL,NAME,INDEX);
      DECLARE (LEVEL,INDEX) FIXED BINARY(15,0);
      DECLARE NAME CHARACTER(8);
        ...     ...
      END UGXERR;
```

This error processing subroutine can do almost anything to try to recover from the error, including calling other subroutines in this system. However, a problem can arise if one of these subroutines calls the error processor because this implies recursive use of the error processor. In FORTRAN, this recursive use of the error processor is not permitted unless UGXERR is an Assembler Language subroutine with certain special properties. If an invalid attempt is made to call UGXERR recursively, the message for the first error will be printed followed by a message with a subroutine name of UGXERR whose description is:

1(4): An attempt has been made to use the subroutine UGXERR recursively.

An example of how to write a recursive error processing subroutine in FORTRAN will be found in the Unified Graphics System Internal Operation and Maintenance Manual [BEA-1]. In PL/1, the programmer may declare subroutine UGXERR to have the RECURSIVE attribute.

When the users version of UGXERR has successfully processed an error, it should signal that no further error processing is

needed by setting LEVEL to zero.

Consider the following non-trivial example of the use on an error processing subroutine. Suppose a program has been written which generates large complicated pictures for a non-interactive device, and suppose it is written as follows:

1. Before each plot, the graphic element is cleared by calling procedure UGEINT with the CLEAR option.
2. The graphic element generators are used to create the graphic element.
3. The graphic element is transmitted to the device.

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

1. Make the graphic element an external variable.
2. Prepare a UGXERR 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, UGXERR should transmit the graphic element to the device by calling UGEPUT and then clear the graphic element by calling UGEINT with the CONTINUE option. The CONTINUE option is needed here because of the way UGELIN works. It draws from the last point to the given point. The CONTINUE option causes the last point to be saved. UGXERR 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 2.8.1: SUBROUTINE UGRERR

This subroutine gives the programmer the ability to make use of the error processor. Any errors reported in this way will be handled in the same manner as described above.

The calling sequences are:

```
(FORTRAN)  CALL UGRERR(LEVEL,SNAME,INDEX)
(PL/1)     CALL UGRERR(LEVEL,SNAME,INDEX);
```

The parameters in the calling sequences are:

LEVEL	The level number of the error. A value of zero will cause the error flags to be cleared.
SNAME	A character string of 8 characters giving the name of the subroutine identifying the error.
INDEX	The index of the error.

SECTION 2.9: OPTIONS SCANNING

Most 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 2.9.1: SUBROUTINE UGOPTN

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. This subroutine is limited in the types of strings that can be processed but these limitations are dictated by the fact that this subroutine is called repeatedly and must be efficient.

The calling sequences are:

```
(FORTRAN)  CALL UGOPTN(OPTIONS,INDATA,EXDATA)
(PL/1)     CALL UGOPTN(OPTIONS,INDATA,EXDATA);
```

The parameters in the calling sequences are:

OPTIONS The options list which will be scanned. The options list may contain as many as 16 items to be scanned.

INDATA An input structure which specifies the items to be scanned for and the format of the output structure. The input consists of a mixture of half word integers and character strings. At the highest level, the format of INDATA is:

1. The number of descriptors in INDATA.
2. A descriptor for each of the items to be scanned for.

Each descriptor has the following format:

1. The descriptor type (1 through 5).
2. The number of bytes in the flag.
3. The byte offset in the output structure where the result of finding a match will be stored.
4. A type-dependent value.
5. The flag (padded on the right to an even number of bytes).

The five types of items that can be searched for and the meaning of the type-dependent value are:

1. A simple flag. The type-dependent value is moved into the two bytes in the output structure if the byte offset is positive. If the byte offset is not positive, the absolute value of the given byte offset is used as the actual offset and the type-dependent value is OR-ed into the two bytes.
2. A flag followed by an equal sign followed by an integer. The type-dependent value is not used. The integer value is stored in the output structure as a two byte fixed point

number.

3. A flag followed by an equal sign followed by a floating point number. The type-dependent value is not used. The floating point value is stored in the output structure as a four byte floating point number.
4. A flag followed by an equal sign followed by a string of characters. The type-dependent value gives the number of bytes in the output structure which are available to hold the character string. The given character string is padded on the right with blanks if necessary.
5. A flag followed by an equal sign followed by a string of bits. The type-dependent value gives the number of bytes in the output structure which are available to hold the bit string. The given bit string is padded on the right with zeros if necessary.

EXDATA An output structure which will have the processed information stored in it.

No error messages are produced by this subroutine. However, the programmer should remember that errors in an options item will simply cause that item to be ignored. Examples of things which can cause an options item to be ignored are (1) an alphabetic character in a numeric field of a type 2 or 3 item, (2) a length of a string in a type 4 or 5 item which is longer than provided for in the output structure, or (3) more than 16 items in the character string being scanned. In addition, misspelled items will not be recognized and will be ignored. Forgetting the terminal asterisk in FORTRAN can produce very unpredictable results.

The following FORTRAN example may help to explain this subroutine. Suppose we wish to search for items of the form FLAG or INT=<integer>. First, let us specify the output structure. In this structure, we need a pair of two byte integers for the output. Therefore:

```
      INTEGER*2     EXDATA(2),EXFLAG,EXINT
      EQUIVALENCE   (EXFLAG,EXDATA(1)),(EXINT,EXDATA(2))
```

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

```
      INTEGER*2     INDATA(13)/2,
     X                          1,4,0,5,'FL','AG,
     X                          2,3,2,0,'IN','T '/
```

With these structues defined, we may now write code to initialize the output structure with its default values and call the options scanning subroutine as follows:

```
      EXFLAG=0
      EXINT=0
      CALL UGOPTN(OPTNS,INDATA,EXDATA)
```

If the character string OPTNS contained FLAG, then EXFLAG will be changed to 5; if it contained INT=<integer>, then EXINT will be changed to the given value.

The corresponding structures in PL/1 would be:

```
DECLARE 1 EXDATA,
          2 EXFLAG FIXED BINARY(15,0),
          2 EXINT FIXED BINARY(15,0);
DECLARE 1 INDATA STATIC,
          2 NEXT FIXED BINARY(15,0) INITIAL(2),
          2 NN1A(4) FIXED BINARY(15,0) INITIAL(1,4,0,5),
          2 NN1B CHARACTER(4) INITIAL('FLAG'),
          2 NN2A(4) FIXED BINARY(15,0) INITIAL(2,3,2,0),
          2 NN2B CHARACTER(4) INITIAL('INT ');
```

SECTION 2.10: MISCELLANEOUS SUBROUTINES

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

SECTION 2.10.1: SUBROUTINE UGDXDY

This subroutine is an aid in using the subroutines UGAXIS and UGCNTR which were 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".

Algorithms of this nature have been described in [GIA-1, DIX-1, LEW-1]. The algorithm used in UGDXDY is based on the information in [DIX-1].

The calling sequences are:

```
(FORTRAN)  CALL UGDXDY(OPTIONS,LODATA,HIDATA,MINLAB,MAXLAB,
                       LOLAB,HILAB,NLAB)
(PL/1)     CALL UGDXDY(OPTIONS,LODATA,HIDATA,MINLAB,MAXLAB,
                       LOLAB,HILAB,NLAB);
```

The parameters in the calling sequences are:

OPTIONS	This parameter is present for consistency with the other calling sequences; no items will be recognized.
LODATA	A floating point value giving the low extent of the data.
HIDATA	A floating point value giving the high extent of the data.

MINLAB	The minimum acceptable number of labeled tic marks.
MAXLAB	The maximum acceptable number of labeled tic marks.
LOLAB	A computed floating point value which will be LODATA reduced to a "round number".
HILAB	A computed floating point value which will be HIDATA increased to a "round number".
NLAB	A computed value which will make all of the labels "round numbers".

No error messages are produced by this subroutine.

SECTION 2.10.2: SUBROUTINE UGPROJ

This subroutine may be used to define a projective transformation from three-dimensional space into two-dimensional space. The transformation may be used with subroutine UG3TO2 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 2.10.1. First a reference point, REFP, and a view direction, VDIR, are given. REFP and VDIR are given in the three-dimensional coordinate system of the object being viewed. 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 contains a square with a side length of SCRZ and its horizontal axis is defined by the vector HDIR. Finally an eye point, E, is defined by moving a distance EYED from REFP in the direction of HDIR. REFP is a point in the three-dimensional coordinate system; VDIR and HDIR are direction numbers of vectors in that system; and SCRD, SCRZ, and EYED are also measured in that system. 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 purpose of the EYED parameter is the generation of stereo pairs; a right eye transformation has a positive value for EYED while a left eye transformation has a minus value. If only a single view is being produced, that is, if EYED is zero, then only the ratio of SCRD to SCRZ is critical; multiplying both by the same constant will not change the transformation. No scissoring of data is done when the points are projected; projected points may lie outside the square area on the projection screen.

The calling sequences are:

```
(FORTRAN)  CALL UGPROJ(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,
                       TRANS)
(PL/1)     CALL UGPROJ(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,
                       TRANS);
```

The parameters in the calling sequences are:

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

EYED=<value> One-half the eye separation for making stereo pairs. 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 A floating point array of dimension 3 giving the projective reference point.

VDIR A floating point array of dimension 3 giving the view direction.

HDIR A floating point array of dimension 3 giving the horizontal direction of the projection screen. If this vector is given as (0.0,0.0,0.0), then a vector parallel to the X-Y plane will be supplied.

SCRD A floating point value giving the distance from REFP to the screen.

SCRZ A floating point value giving the size of the screen.

TRANS A floating point array of dimension 31 which will be set to the projective transformation. This array will contain a 3 by 4 projection matrix as well as the given parameters.

No error messages are produced by this subroutine.

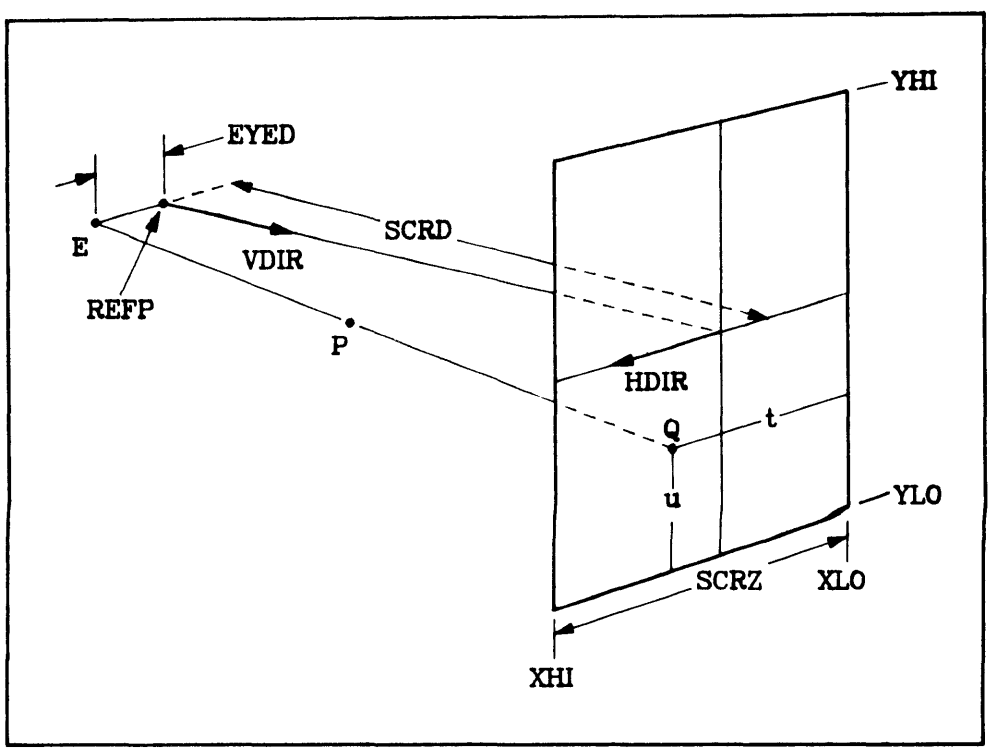


Figure 2.10.1: The Definition of a Projective Transformation.

SECTION 2.10.3: SUBROUTINE UGORTH

This subroutine may be used to define an orthogonal transformation from three-dimensional space into two-dimensional space. The transformation may be used with subroutine UG3TO2 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 UGPROJ. 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 may be utilized by other subroutines.

The calling sequences are:

```
(FORTRAN)  CALL UGORTH(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,
                       TRANS)
(PL/1)     CALL UGORTH(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,
                       TRANS);
```

The parameters in the calling sequences are:

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

- XLO=<value> The X coordinate at the left hand side of the screen. The default value is 0.0.
- XHI=<value> The X coordinate at the right hand side of the screen. The default value is 1.0.
- YLO=<value> The Y coordinate at the bottom of the screen. The default value is 0.0.
- YHI=<value> The Y coordinate at the top of the screen. The default value is 1.0.

REFP A floating point array of dimension 3 giving the orthogonal reference point.

VDIR A floating point array of dimension 3 giving the view direction.

HDIR A floating point array of dimension 3 giving the horizontal direction of the projection screen. If this vector is given as (0.0,0.0,0.0), then a vector parallel to the X-Y plane will be supplied.

SCRD A floating point value giving the distance from REFP to the screen.

SCRZ A floating point value giving the size of the screen.

TRANS A floating point array of dimension 31 which will be set to the orthogonal transformation. This array will contain a 3 by 4 projection matrix as well as the given parameters.

No error messages are produced by this subroutine.

SECTION 2.10.4: SUBROUTINE UG3TO2

This subroutine uses a transformation defined by subroutine UGPROJ or UGORTH to project a three-dimensional point into two dimensions.

The calling sequences are:

```
(FORTRAN)  CALL UG3TO2(TRANS,PT3D,PT2D)
(PL/1)     CALL UG3TO2(TRANS,PT3D,PT2D);
```

The parameters in the calling sequences are:

TRANS	A floating point array of dimension 31 containing the transformation.
PT3D	A floating point array of dimension 3 giving the coordinates of the three-dimensional point.
PT2D	A floating point array of dimension 2 which will be set to the projected point.

No error messages are produced by this subroutine.

SECTION 2.10.5: SUBROUTINE UGCNVF

This subroutine may be used to convert a floating point number to a character string. The conversion is similar to the conversion done by a Fw.d format item on a FORTRAN write. The character string may then, for example, be passed to subroutine UGETXT. This subroutine is only available in FORTRAN; the "PUT STRING" statement of PL/1 performs the equivalent operation in that language.

The calling sequence is:

```
(FORTRAN)  CALL UGCNVF(NUMBER,FWID,FDEC,STRING)
```

The parameters in the calling sequence are:

NUMBER	The floating point value which is to be converted to a character string.
FWID	A fixed point value giving the field width of the resulting string. If the given value is less than one, a one will be used; if it is greater than 12, then 12 will be used.
FDEC	A fixed point value giving the number of places to the right of the decimal point. If FDEC has the value zero, then no decimal point will appear in the string. If the given value is less than zero, a zero will be used; if it is greater than 10, then 10 will be used.
STRING	The resulting character string. The string must be at least FWID characters long.

No error messages are produced by the subroutine. If, for any reason, the number cannot be converted, the field will be filled with asterisks.

The Unified Graphics System may be used in conjunction with FORTRAN (either the G or H Compilers) or PL/1 (either the Optimizing Compiler or the F-Level Compiler).

There are two program libraries of which the user must be aware. The first contains modules which must be made available to the LINK-EDITOR or LOADER by concatenating it with the SYSLIB data set. The LINK-EDITOR or LOADER will select the necessary subroutines from this library and incorporate them into the load module. The second library must be made available at execution time by making it part of the JOBLIB or STEPLIB data sets. The modules in this library are loaded dynamically and contain the device-dependent code, the character stroke generators and other utility routines. The names of the required data sets are given in the following sections.

The programmer may use the LINK-EDITOR to create overlay structures. Any of the modules in this package may be overlaid with one exception. This exception is that the module named UGX000 must not be overlaid while any device is open or while any other use is being made of the subroutines in this system. In the FORTRAN version, the programmer should call UGCLOS with the ALL option immediately before UGX000 is overlaid. Calling UGCLOS in this manner will assure that UGX000 is in a state where it can be overlaid.

SECTION 3.1: USING THE FORTRAN VERSION

The library which must be made available to the LINK-EDITOR or LOADER is WYL.CG.RCB.UGFTNLIB. The library which must be made available at execution time is WYL.CG.RCB.UGRUNLIB.

SECTION 3.2: USING THE PL/1 OPTIMIZING COMPILER VERSION

The library which must be made available to the LINK-EDITOR or LOADER is WYL.CG.RCB.UGPLOLIB. The library which must be made available at execution time is WYL.CG.RCB.UGRUNLIB.

There is a third library which is made available for the convenience of the PL/1 programmer. This data set is WLY.CG.RCB.UGPL1DCL. It is a source library and contains entry declaration statements for all of the procedures in this system. It is suggested that this library be concatenated with the SYSLIB

data sets in the PL1 step. If this is done, and the INCLUDE or MACRO option is specified, then the PL/1 statement:

```
  INCLUDE UGOPEN,UGCLOS;
```

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

SECTION 3.3: USING THE PL/1 F-LEVEL COMPILER VERSION

The library which must be made available to the LINK-EDITOR or LOADER is WYL.CG.RCB.UGPLFLIB. The library which must be made available at execution time is WYL.CG.RCB.UGRUNLIB.

The calling sequences for the F-Level Compiler version are exactly the same as the ones for the Optimizing Compiler. Therefore, the library WYL.CG.RCB.UGPL1DCL may also be used.

This section gives a detailed description of each of the non-interactive graphic devices that are supported by this system. For each device there is a description of the hardware, including its features and limitations. If there is anything unusual about the manner in which a subroutine relates to a device, it will be mentioned in these sections.

SECTION 4.1: THE CALCOMP DRUM PLOTTER (MODEL 1136)

The CALCOMP Drum Plotter, Model 1136, is an off-line electro-mechanical pen plotting device. The device is located in Bacina Hall at the Campus Facility of the Stanford Center for Information Processing. Programs generating pictures for this device will write their plotting orders onto a tape which must be physically carried to the device. The Model 1136 Plotter has three pens which may be selected under program control. The paper which can be used on this device comes in rolls of two widths. The narrow paper has a plotting width, that is, the width reachable by all three pens, of 10 inches. With the large paper, the device has a plotting width of 33 inches. Pen movement occurs at 400 increments per inch. Both ball-point pens and liquid ink pens with various colored inks are available. The liquid ink pens come in a number of line widths.

This device is selected by the options item CALDRSH or CALDRLG in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME and TITLE. The device-dependent code uses characters 1-8, usually the job name, and characters 10-16, usually the bin number, from TITLE to identify the output. In addition, UGOPEN will recognize three additional items: PEN2=<value>, PEN3=<value> and INCHGRID. These items are used to tell the program how the pens will be set up and what paper will be on the plotter when the pictures are produced. Suppose, for example, the items PEN2=VBRT and PEN3=COL1COL4 are given. In this case, the second pen will be used whenever a picture item has the VBRT attribute and the third pen will be used whenever a picture item fails the PEN2 test and has the attributes of COL1 or COL4. All picture items which fail both of these tests will be drawn with the first pen. The character strings given by PEN2 and PEN3 may each contain up to four intensity level or color codes. The default values for PEN2 and PEN3 are null strings. The plotter is usually set up with plain paper; however, if paper ruled with a one-inch grid is used, then the INCHGRID item should be given. The principal difference when the INCHGRID item is given is that the plotter is not adjusted by driving the pen into the limit switches and then backing off; instead it is assumed that the plotter operator will adjust the plotter so that the

principal pen is over a grid marking at the start of the run. This set up procedure is not easy to do, especially when multiple pens are being used, and ruled paper should be avoided. The programmer and device scaling and scissoring limits for the small size paper are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 4000 \\ 0 & 4000 \end{bmatrix}$$

The limits for the large size paper are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 13200 \\ 0 & 13200 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

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

```
//GO.UGDEVICE  DD DSNAME=&&PLOT,DISP=(NEW,PASS),
//             VOLUME=SER=PLOT,UNIT=T9-1600,LABEL=(1,SL),
//             DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
```

The name of the basic device-dependent module for this device is UGCALD2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

SECTION 4.2: THE CALCOMP MICROFILM PLOTTER (MODEL 1675)

The CALCOMP Microfilm Plotter, Model 1675, is an off-line device with a 5 inch flat-faced CRT. This CRT is photographed by a camera which can contain 16mm unsprocketted film or 105mm microfiche. Programs generating pictures for this device will eventually have their plotting orders written onto a tape which must be physically carried to the device. This unit can display points, lines, and a single character size at 30 intensity levels. All 30 intensity levels are not useful, even fewer are distinguishable, and only four levels are supported by this system. 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 left, the second picture is below the first, etc.

This device is selected by the options item CAL16MU or CALFICH in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME, FULSCR, and TITLE. For film output, the device-dependent code uses characters 1-8, usually the job name, and characters 10-16, usually the bin number, to form leader frames to identify the output; for microfiche, all 24 characters

are used to form a title on each fiche. In addition, UGOPEN will recognize the item LEADER=<value> which gives the number of frames of leader which will be put at the beginning and end of film output; the default value is 20. The characters available on this device are:

VSHL SPACING=0.00916 109 Characters/Line(132 with FULSCR)

The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 2130 & 6062 \\ 2130 & 6062 \end{bmatrix}$$

When FULSCR is used, the values are:

$$\begin{bmatrix} -0.10427 & 1.10427 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 1720 & 6472 \\ 2130 & 6062 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

A DD statement is required at execution time to define the output data set. If you wish to write the display orders on your own tape, a typical DD statement is:

```
//GO.UGDEVICE  DD DSNAME=&&PLOT,DISP=(NEW,PASS),
//             VOLUME=SER=uuuuuu,UNIT=T9-1600,LABEL=(1,SL),
//             DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480,DEN=3)
```

As an alternative to using your own tape, you may spool your graphic output onto the system by supplying a DD statement like:

```
//GO.UGDEVICE  DD SYSOUT=X,
//             DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

for 16mm unsprocketted film, or:

```
//GO.UGDEVICE  DD SYSOUT=Z,
//             DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

for Microfiche. At regular intervals, the operations staff will gather the spooled output from a number of jobs together on a single tape and run that tape.

The name of the basic device-dependent module for this device is UGCALH2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

SECTION 4.3: 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 device can display points, lines, and characters of a single size at a single intensity level. This section describes a method of using this device in an essentially non-interactive way in conjunction with the WYLBUR Text Editing System [FAJ-1, WYL-1]. To use the

TEKTRONIX 4013 in this manner, the programmer runs a job which uses this device in the manner described below. That job will write its pictures into a card image, partitioned or sequential data set. When a partitioned data set is used, each picture is a separate member. When a sequential data set is used, the pictures are separated by IEBUPDTE control cards; that is, a card:

```
./        ADD   NAME=<value>
```

precedes the data for each picture; if a picture has an alias, the picture data is followed by the card:

```
./        ALIAS NAME=<value>
```

and the last card in the data set is:

```
./        ENDUP
```

After the data set with pictures in it has been created, the user may sign on to WYLBUR from a TEKTRONIX 4013 terminal and issue a "USE" or "COPY" command to obtain a member of the partitioned data or a segment of a sequential file. When a sequential data set is being used, the user can search for all cards with "./" in columns 1 and 2; no data card will ever contain these characters in column 1 and 2. The WYLBUR command "LIST UNNUMBERED" will cause the screen to be cleared and the picture drawn. The audible alarm will sound when the picture is complete.

This device is selected by the options item PDS4013 (if a partitioned data set is being created), or SEQ4013 (if a sequential data set is being created) in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME, FULSCR, PICTID, and PICTSQ. In addition, UGOPEN will recognize the item BAUDRATE=<value>. BAUDRATE gives the speed at which data will be transmitted to the TEKTRONIX 4013. Its value may be any number from 300 to 9600 and its default value is 2400. Transmitting at a higher baud rate than specified can cause distorted or partial pictures. Points with the VDIM or DIMM option are plotted as single dots; all other points are plotted as four adjacent dots. The character sizes available on this device are:

```
LARG   SPACING=0.01797   56 Characters/Line(74 with FULSCR)
```

The programmer and device scaling and scissoring limits without FULSCR are:

```
┌          ┐   ┌          ┐
│ 0.0   1.0 │   │ 122   901 │
│ 0.0   1.0 │   │   0   779 │
└          ┘   └          ┘
```

When FULSCR is used, the values are:

```
┌                  ┐ ┌          ┐
│ -0.15661   1.15661 │ │ 0   1023 │
│  0.0       1.0     │ │ 0    779 │
└                  ┘ └          ┘
```

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device, IARRAY(5) will contain the number of card images which have been produced, and IARRAY(6) will contain the current value of the PICTSQ sequence number. Other entries in IARRAY will be zero. Changing IARRAY(6) with the PUT option in UGRINF will <u>not</u> change the names being assigned to the members of the partioned data set.

A DD statement is required at execution time to define the output data set. A typical DD statement when PDS4013 is used is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2,3),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

A typical DD statement when SEQ4013 is used is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

There is another way that output to a sequential data set may be used under WYLBUR. Instead of writing to an allocated data set, the JCL can be:

```
//GO.UGDEVICE  DD SYSOUT=B,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

and the job can be submitted with a "RUN HOLD OUTPUT" command. The "FETCH" command can then be used to obtain the punched card output for display at the terminal in the usual manner. The basic difference in running like this is that a distinct name does not have to be generated for the output data set. This can be an advantage when a person is trying to run a number of different jobs.

The output data set contains information in card image form with at most 72 characters per card. The first card of each picture contains a title identifying it as a display file for the TEKTRONIX 4013, a clear screen command, and some idle characters. Depending on the baud rate, additional cards containing idle characters may follow the first card. The picture data itself starts on a new card following any idle cards.

The name of the basic device-dependent module for this device is UGPDS42. In addition to the errors described with each subroutine, this module may detect errors whose description is:

- 1(4): Not enough directory blocks are available in the output partitioned data set. You will have to increase the third number in the SPACE parameter on your JCL card for the output data set.
- N(4): Where N is greater than 255: An extremely unusual error condition has been detected.

SECTION 4.4: DISPLAY FILES FOR THE GEN-COM 300-Q

This device is a typewriter terminal which is made by Gen-Com Systems, Inc. There are a few of the Model 300-Q terminals available at SLAC. This device is different from the usual typewriter terminal in that it can be positioned one-sixth of a character width horizontally, and one-eighth of a character height vertically. Thus, the period character can be used to draw reasonably good lines. Since the terminal must run at a 300 baud rate, complicated pictures can require a long time to plot.

This section describes a method of using this device in a non-interactive way in conjunction with the WYLBUR Text Editing System [PAJ-1, WYL-1]. To use the GEN-COM in this manner, the programmer runs a job which uses this device in the manner described below. That job will write its pictures into a card image, partitioned or sequential data set. When a partitioned data set is used, each picture is a separate member. When a sequential data set is used, the pictures are separated by IEBUPDTE control cards; that is, a card:

```
./        ADD   NAME=<value>
```

precedes the data for each picture; if a picture has an alias, the picture data is followed by the card:

```
./        ALIAS NAME=<value>
```

and the last card in the data set is:

```
./        ENDUP
```

After the data set with pictures in it has been created, the user may sign on to WYLBUR from a GEN-COM terminal and issue a "USE" or "COPY" command to obtain a member of the partitioned data or a segment of a sequential file. When a sequential data set is being used, the user can search for all cards with "./" in columns 1 and 2; no data card will ever contain these characters in column 1 and 2. The WYLBUR command "LIST UNNUMBERED NONL" will cause the picture to be drawn. The paper should be positioned so that the LIST command is typed at the fifth line above the perforations on the paper. The paper should consist of the standard 11 inch by 14 inch computer paper. The audible alarm will sound when the picture is complete.

This device is selected by the options item PDS300Q (if a partitioned data set is being created), or SEQ300Q (if a sequential data set is being created) in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME, FULSCR, PICTID, and PICTSQ. In addition, UGOPEN will recognize the item ELITE, which specifies that elite spacing (12 characters per inch) is to be used when printing characters instead of the usual pica spacing (10 characters per inch). The character sizes available on this device are:

VSML(PICA)	SPACING=0.01002	100 Characters/Line (132 with FULSCR)
VSML(ELITE)	SPACING=0.00835	120 Characters/Line (158 with FULSCR)

The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 93 & 692 \\ 0 & 479 \end{bmatrix}$$

When FULSCR is used, the values are:

$$\begin{bmatrix} -0.15609 & 1.15609 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 786 \\ 0 & 479 \end{bmatrix}$$

When UGRINP is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device, IARRAY(5) will contain the number of card images which have been produced, and IARRAY(6) will contain the current value of the PICTSQ sequence number. Other

entries in IARRAY will be zero. Changing IARRAY(6) with the PUT option in UGRINF will not change the names being assigned to the members of the partioned data set.

A DD statement is required at execution time to define the output data set. A typical DD statement when PDS300Q is used is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2,3),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

A typical DD statement when SEQ300Q is used is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

There is another way that output to a sequential data set may be used under WYLBUR. Instead of writing to an allocated data set, the JCL can be:

```
//GO.UGDEVICE  DD SYSOUT=B,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
```

and the job can be submitted with a "RUN HOLD OUTPUT" command. The "FETCH" command can then be used to obtain the punched card output for display at the terminal in the usual manner. The basic difference in running like this is that a distinct name does not have to be generated for the output data set. This can be an advantage when a person is trying to run a number of different jobs.

The output data set contains information in card image form with at most 72 characters per card. The first card of each picture contains a title identifying it as a display file for the GEN-COM 300-Q, and some initialization orders. The orders for the picture itself starts on the second card.

The name of the basic device-dependent module for this device is UGPDS32. In addition to the errors described with each subroutine, this module may detect errors whose description is:

1(4): Not enough directory blocks are available in the output partitioned data set. You will have to increase the third number in the SPACE parameter on your JCL card for the output data set.

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

SECTION 4.5: THE VERSATEC ELECTROSTATIC PLOTTER (MODEL 1200A)

The VERSATEC Electrostatic Printer/Plotter, Model 1200A, is connected directly to the central computing facility through a special controller. This controller can accept graphic data which gives the X and Y coordinates of end points of line segments and will convert this data to the raster data required by the plotting unit. This raster data consists of a series of

lines of dots which are printed across the width of the paper. Each line of dots is called a "scan line". The Model 1200A plotter has 200 dots per inch along a scan line and 200 scan lines per inch. The character generator on the unit can only be used in the print mode and cannot be used in the plot mode. The printer/plotter normally has 8.5 by 11 inch fan-fold paper mounted on it but it can also use continuous roll paper.

This device is selected by the options item VEP12FF or VEP12CR in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME and FULSCR. In addition, UGOPEN will recognize four additional items; EXTSORT, MAXMSA=<value>, LINEMX=<value>, and GENIL. EXTSORT indicates that an external sort is to be performed on the data instead of the default internal sort. The external sort requires much less memory at run time but does require more complicated JCL. MAXMSA has a default value of 4096 and is only used during an internal sort. MAXMSA gives the size of the blocks of memory where the line segments, which compose the picture, will be saved. The line multiplicity, LINEMX, has a default value of 32 and is the number of scan lines which will be developed concurrently in the controller. Increasing MAXMSA, when complicated pictures are being generated, can reduce execution time. LINEMX should not normally be changed; however, for very complicated pictures, the memory in the controller may be exceeded. In this case, LINEMX will have to be reduced. GENIL causes the Unified Graphics System to utilize the intensity level options. Normally, lines are drawn as a single string of dots and points with the VDIM or DIMM option are plotted as single dots while all other points are plotted as four adjacent dots. When GENIL is used, VDIM lines are drawn by a single line of dots, DIMM lines are composed of two adjacent lines of dots, BRIT lines are three adjacent dots, and VBRT lines are five adjacent dots. Points are drawn in a similar manner. The thicker lines produce plots that have a much better appearance and can be reproduced better. However, the GENIL option should be used only when finished plots are required. On the average, this option will cause three times as much output to be written to the SYSOUT=P data set. When internal sorts are being performed, an average of three times as much memory will be required to contain the picture; for external sorts, the temporary data set will contain an average of three times as much data. The programmer and device scaling and scissoring limits with VEP12FF and without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 268 & 1843 \\ 0 & 1575 \end{bmatrix}$$

When VEP12FF and FULSCR are used but GENIL is not used, the values are:

$$\begin{bmatrix} -0.17016 & 1.17016 \\ 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} 0 & 2111 \\ 0 & 1575 \end{bmatrix}$$

When VEP12FF, FULSCR and GENIL are all used, the values are:

```
[ -0.16889   1.16889 ]  [ 2   2109 ]
[  0.0       1.0     ]  [ 0   1575 ]
```

When VEP12CR is used, the values are:

```
[ 0.0   1.0 ]           [  0   2000 ]
[ 0.0   1.0 ]           [ 55   2055 ]
```

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

These scaling and scissoring limits shown here cause the largest possible area to be used on fan-fold paper. For continuous roll paper, the plotting area is limited to 10 inches instead of the full 10.56 inches. In this way, the plots from this device will be exactly the same size as the plots from the CALCOMP Drum Plotter when using the small size paper.

When an internal sort is performed, the only DD statement that is required at execution time is:

```
//GO.UGDEVICE  DD SYSOUT=P,
//             DCB=(RECFM=FBA,LRECL=134,BLKSIZE=1340
```

If the pictures are to be drawn on continuous roll paper, you will also have to include a card like:

```
//*FORMAT      PR,DDNAME=GO.UGDEVICE,FORMS=ROLL
```

When an external sort is performed, the output data set is a temporary data set and a typical DD statement is:

```
//GO.UGDEVICE  DD DSNAME=&&PLOT1,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,5),
//             DCB=(RECFM=FB,LRECL=12,BLKSIZE=1440)
```

The job step which generates this temporary data set must be followed by job steps which sort the data set and then re-format it. The JCL statements to do this are:

```
//UGSORT  EXEC SORT
//SORT.SORTIN  DD DSNAME=&&PLOT1,DISP=(OLD,DELETE)
//SORT.SORTOUT DD DSNAME=&&PLOT2,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(CYL,5),
//             DCB=(RECFM=FB,LRECL=12,BLKSIZE=1440)
//SORT.SYSIN   DD *
         SORT  FIELDS=(1,2,FI,A,3,2,FI,A)
//UGFMAT  EXEC PGM=UGPPVEP
//STEPLIB      DD DSNAME=WYL.CG.RCB.UGRUNLIB,DISP=SHR
//SYSIN        DD DSNAME=&&PLOT2,DISP=(OLD,DELETE)
//SYSOUT       DD SYSOUT=P,
//             DCB=(RECFM=FBA,LRECL=134,BLKSIZE=1340)
//SYSPRINT     DD SYSOUT=A
```

For continuous roll paper, you will also need a card like:

```
//*FORMAT      PR,DDNAME=UGFMAT.SYSOUT,FORMS=ROLL
```

The names of the basic device-dependent modules for this device are UGVP122 (for internal sorts) and UGVP123 (for external sorts). In addition to the errors described with each subroutine, these modules may detect errors whose description is:

1(4): You have made changes to the scaling and scissoring

limits which are erroneous and invalid.

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

SECTION 4.6: A PARTITIONED DATA SET PSEUDO-DEVICE

This section does not describe a display device but instead describes a simple method by which pictures may be saved in a partitioned data set in a device-independent form. 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 is different. The pictures are saved at a very early stage; the information written into the partitioned data set is essentially that contained in the graphic elements. Thus, the information written to the data set has not been scissored, has not had the necessary characters converted to strokes, etc. A method is provided to read the pictures from the partitioned data set.

This pseudo-device is selected by the options item PDSPDEV in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME, PICTID, and PICTSQ. In addition, UGOPEN will recognize the item INPUT. This latter item indicates that the pictures are to be read from the partitioned data set by subroutine UGEGET. The programmer and device scaling and scissoring limits are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1000 \\ 0 & 1000 \end{bmatrix}$$

When UGRINP is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device, and IARRAY(6) will contain the current value of the PICTSQ sequence number. Other entries in IARRAY will be zero. Changing IARRAY(6) with the PUT option in UGRINP will <u>not</u> change the names being assigned to the members of the partioned data set. When this pseudo-device is opened with the INPUT option, the subroutines UGEPUT, UGPICT, and UGCTRL may not be used; only UGEGET will perform a useful service.

A DD statement is required at execution time to define the input or output data set. A typical DD statement is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2,3),RLSE),
//             DCB=(RECFM=U,BLKSIZE=4000)
```

The BLKSIZE 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.

When pictures are being written out, each element is a separate record and each picture is a separate member. In addition, the

programmer scaling and scissoring limits are saved in the data set when the first element of each picture is written. Therefore, under most circumstances, the programmer should create the pictures so that all elements depend on the same programmer limits and these limits should be set by UGSCAL or UGRSCL before the first element of each picture is saved in the data set. When pictures are being read, these programmer scaling and scissoring limits will be read with the first element of each picture.

The name of the basic device-dependent module for this device is UGPDSD2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

1(4): Not enough directory blocks are available in the output partitioned data set. You will have to increase the third number in the SPACE parameter on your JCL card for the output data set.

2(4): An uncorrectable error has been detected while trying to read a partitioned data set member.

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

SECTION 4.7: A SEQUENTIAL DATA SET PSEUDO-DEVICE

This section describes a second pseudo-device. For this pseudo-device, the pictures are written into a sequential data set after all of the transformations, such as scaling and scissoring, character stroke generation, etc., have been performed. The output data set is a standard data set which may be read by other FORTRAN or PL/1 programs.

One possible use of this pseudo-device is with graphic devices that are not supported by the existing Unified Graphics System. An existing program could generate data for this pseudo-device, and a very simple program could be written to read the data set and call vendor supplied subroutines to produce the picture. Another use could be the generation of pictures on the IBM 360/370 computers by writing the data from this pseudo-device onto a tape and then producing the pictures on a graphic device connected to a remote computer. The tape could be transported to the remote computer where a very simple program could read the tape and produce the picture using vendor supplied subroutines.

This pseudo-device is selected by the options item SEQPDEV in subroutine UGOPEN. The only standard item which will be processed by UGOPEN is DDNAME. In addition, UGOPEN will recognize a large number of items which may be used to cause this pseudo-device to simulate a wide range of graphic devices. The items PXLO=<value>, PXHI=<value>, PYLO=<value>, and PYHI=<value> may be used to change the initial programmer coordinate system from the normal default values shown below. The items DXLO=<value>, DXHI=<value>, DYLO=<value>, and DYHI=<value> may change the initial device coordinate system. The item ZSCRN=<value> may be used to set the screen size to something

other than the default value of 25.0. The items PXUP, PXRIGHT, PXDOWN, or PXLEFT may be used to indicate that the picture is extendable in some direction. The items WVSHL=<value>, XOVSHL=<value>, and YOVSHL=<value> control the assumptions made by the pseudo-device about the VSHL character size. If WVSHL is given, the pseudo-device assumes that a VSHL hardware character generator is available and WVSHL gives the width of the characters in raster units; if WVSHL is not given, the pseudo-device assumes that a character generator capable of drawing VSHL characters is not available. When WVSHL is given, XOVSHL and YOVSHL, which have default values of 0, give the distance from the hardware origin of the character to the center of the character, in rasters, in the X and Y directions, respectively. The items WSHAL=<value>, XOSHAL=<value>, and YOSHAL=<value> serve the same function for the SHAL hardware character generator; WLARG=<value>, XOLARG=<value>, and YOLARG=<value> serve for the LARG character size; and WVLRG=<value>, XOVLRG=<value>, and YOVLRG=<value> serve for VLRG. Normally, the hardware character generator is assumed to only be capable of processing characters at an angle of zero degrees. However, the items CROT90, CROT180 and CROT270 may be used to indicate that the hardware character generator can draw characters at 90 degrees, 180 degrees, or 270 degrees in addition to the usual zero degrees. The final item that may occur in the options list of UGOPEN is PASSLS. Normally, all of the given line segments with line structure will be broken down into short lines and points. If PASSLS is given, no line structure is generated; instead, flags are set in the output data indicating that line structure was requested. The purpose of all of these options is to allow the pseudo-device to simulate a wide variety of graphic devices. The only difficulty in specifying all of these options is that the options scanning module only processes the first 16 items. However, this is an easy limit to increase if difficulty is encountered. The normal programmer and device scaling and scissoring limits are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1000 \\ 0 & 1000 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

A DD statement is required at execution time to define the output data set. A typical DD statement is:

```
//GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.name,DISP=(NEW,KEEP),
//             VOLUME=SER=SCR001,UNIT=2314,
//             SPACE=(TRK,(10,2),RLSE),
//             DCB=(RECFM=VBS,BLKSIZE=1500)
```

The blocksize parameter in the DCB must be at least 512 bytes.

The name of the basic device-dependent module for this device is UGSEQD2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

Six types of records are written into the output data set: (1) beginning of picture record, (2) data mode specification (color, intensity, etc.), (3) point data, (4) line data, (5) text data for a hardware character generator, and (6) end of data record.

Each record starts with two half-word integers. The first half-word gives the number of bytes in the record, while the second half-word gives the type of the record (the value will be 1 to 6 as described above). The maximum length of a record is 268 bytes.

A beginning of picture record (Type 1) only contains the two half-word integers. Its form is:

Len.	Type

The data mode specification record (Type 2) consists of four additional bytes. The first byte specifies line structure for all of the lines that follow (X'01' for SOLD, X'02' for DASH, 'X04' for DDSH, and X'08' for DOTS), the second byte specifies intensity level (X'01' for VDIM, X'02' for DIMM, X'04' for BRIT, and X'08' for VBRT), the third byte specifies color (X'01' for COL1, X'02' for COL2, X'04' for COL3,and X'08' for COL4), and the fourth byte specifies wink mode (X'01' for WINK or X'02' for STDY). The form of this record is:

Len.	Type	Mode Bytes

A point data record (Type 3) consists of two additional half-word integers containing the X and Y coordinates of the point. Its form is:

Len.	Type	X	Y

A line data record (Type 4) consists of three additional half-word integers which contain the X and Y coordinates and a blanking bit (0 for blank or 1 for draw). Its form is:

Len.	Type	X	Y	BBit

A text data record (Type 5) consists of a number of additional items. There are two half-word integers which contain the X and Y coordinates of the first character. Next are two bytes of data, the first of which specifies character orientation (X'01' for zero degrees, X'02' for 90 degrees, X'04' for 180 degrees, and X'08' for 270 degrees) and the second of which specifies character size (X'01' for VSML, X'02' for SMAL, X'04' for LARG, and X'08' for VLRG). Next comes a half-word integer which gives the number of characters following. The last item in the record is the characters, one to a byte, in EBCDIC. The form of this record is:

Len.	Type	X	Y	Data	NTxt	Text...

Finally, an end of data record (Type 6) only contains the two half-word integers. Its form is:

Len.	Type

A set of FORTRAN statements which will read a data record on the IBM 360/370 is:

```
      INTEGER*2     RECD(134), RECI
      LOGICAL*1     RECL(268)
      EQUIVALENCE   (RECD(1),RECL(1),RECI)
      READ(file) RECI,(RECL(I),I=3,RECI)
```

The executable instruction reads a single record into the array RECD.

In PL/1, the equivalent set of statements are:

```
      DECLARE 1 RECD,
                2 LENG FIXED BINARY(15,0),
                2 TYPE FIXED BINARY(15,0),
                2 DATA CHARACTER(264);
      OPEN FILE(name) SEQUENTIAL INPUT RECORD;
      READ FILE(name) INTO(RECD);
```

After a record has been read in, other based variable templates can be overlaid on RECD.DATA depending on the value of RECD.TYPE.

This section gives a detailed description of each of the interactive graphic devices that are supported by this system. For each device there is a description of the hardware, including its features and limitations. If there is anything unusual about the manner in which a subroutine relates to a device, it will be mentioned in these sections.

SECTION 5.1: THE IBM 2250 DISPLAY CONSOLE

The IBM 2250 Display Console consists of a CRT with a 12 inch by 12 inch working area, a keyboard, a light pen, and an audible alarm. There are three IBM 2250/Model 2 consoles at SLAC: two are on the first floor of the Computer Building and one is in End Station A. These units can display points, lines, and two sizes of characters at a single intensity level. 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 all three display consoles. A program running on the central computer system and using an IBM 2250 will write display orders into an area of the buffer allocated to the display. The orders 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.

This device is selected by the options item IBM2250 in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME and MAXENUM. In addition, UGOPEN will recognize the items MAXESIZ=<value>, MAXDFIL=<value>, and GENLS. MAXESIZ gives the maximum number of bytes that may be in an element after it has been translated to device orders; the default value is 4096. MAXDFIL gives the maximum number of bytes that may be in a complete display file at one time; the default value is 4096 and the maximum allowable value is 5376. GENLS indicates that line structure is to be generated; normally line structure parameters will be ignored on this device and all lines will be drawn as solid lines. The reason for this is that breaking a line up into dashes and/or dots results in a very significant increase in the amount of data put into the display file. The character sizes available on this device are:

```
SMAL   SPACING=0.01367   74 Characters/Line
LARG   SPACING=0.02051   49 Characters/Line
```

The programmer and device scaling and scissoring limits are:

```
┌          ┐          ┌          ┐
│ 0.0  1.0 │          │ 0   4095 │
│ 0.0  1.0 │          │ 0   4095 │
└          ┘          └          ┘
```

When UGRINF is called with the GET option, IARRAY(1) will contain

the number of words used in the largest graphic element which has been transmitted to the device, IARRAY(2) will contain the number of bytes in the largest graphic element after it has been translated to device-dependent orders, IARRAY(3) will contain the largest number of elements that were present in a single picture, IARRAY(4) will contain the number of bytes in the largest display file, and IARRAY(5) will contain the number of bytes in the last graphic element after it has been translated to device-dependent orders. Other entries in IARRAY will be zero.

Graphic elements may be fully manipulated on this device. The attentions KBRD and LPEN may be utilized. An arbitrary number of keyboard input buffers may be in a single display file and they should consist of an even number of characters. The LPEN attention returns the identification of the element being pointed at, a byte offset into the element, and an approximate X and Y coordinate of the item being selected. The non-detectability of elements on this device is produced by software; therefore, detectable elements should be well separated from other elements on the screen even if the other elements are non-detectable.

A DD statement is required at execution time to select the display console. A typical DD statement is:

```
//GO.UGDEVICE  DD UNIT=SCOPE1
```

This card selects unit number one in the Computer Building.

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 name of the basic device-dependent module for this device is UGIB502. In addition to the errors described with each subroutine, this module may detect errors whose description is:

- 1(4): An asynchronous buffer error has occurred in the IBM 2840; the unit is malfunctioning.
- N(4): Where N is greater than 255: An extremely unusual error condition has been detected.

SECTION 5.2: THE TEKTRONIX 4013 DISPLAY TERMINAL

This section describes a method of using the TEKTRONIX 4013 in an interactive manner in conjunction with the HILTEN Terminal Manager. The TEKTRONIX 4013 is a storage display terminal with an eight inch wide by six inch high screen, a keyboard, a set of thumb wheels to control cross-hairs on the screen, and an audible alarm. This device can display points, lines, and characters of a single size at a single intensity level. Selective erasure on this device is impossible; once an item is written on the screen, it remains there until the entire screen is cleared.

This device is selected by the options item TEK4013 in subroutine UGOPEN. The only standard item which will be processed by UGOPEN is FULSCR. In addition, UGOPEN will recognize the items SSNAME=<value> and MAGICWD=<value> whose values are character strings of one to eight characters, and BAUDRATE=<value>. SSNAME gives a sub-system name and MAGICWD gives a magic-word. The use of these items will be described below. BAUDRATE gives the speed at which data will be transmitted to the TEKTRONIX 4013. Its value may be any number from 300 to 9600 and its default value is 2400. Transmitting at a higher baud rate than specified can cause distorted or partial pictures. Points with the VDIM or DIMM option are plotted as single dots; all other points are plotted as four adjacent dots. The character sizes available on this device are:

LARG SPACING=0.01797 56 Characters/Line(74 with FULSCR)

The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 122 & 901 \\ 0 & 779 \end{bmatrix}$$

When FULSCR is used, the values are:

$$\begin{bmatrix} -0.15661 & 1.15661 \\ 0.0 & 1.0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1023 \\ 0 & 779 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

The attentions KBRD and SPTR may be utilized but only one of them should be enabled at one time. When both are enabled, SPTR takes precedence and KBRD is ignored. Only one keyboard input buffer may be on the screen at one time and the initial value of STRING in UGKPUT is ignored and blanks are used instead. The LCASE item may be used in UGKGET. Enabling the SPTR attention causes the cross-hairs to be put on the screen; the cross-hairs are removed when SPTR is disabled. Actually, the keyboard is not unlocked or the cross-hairs put on the screen until UGRATN has been called. The RDSPTR item in UGCTRL <u>cannot</u> be used. The SPTR attention returns the EBCDIC character of the key pushed to generate the attention as the key number in addition to the X and Y coordinate of the cross-hairs on the screen.

The TEKTRONIX 4013 terminals at SLAC are usually run under the control of the HILTEN Terminal Manager and no special DD statement is required at execution time. To run an interactive program using one of these devices, the user should: (1) sign onto a terminal and begin using the WYLBUR text editing system in the usual manner. (2) submit a job which uses this device and wait until the job identifies itself to HILTEN as a sub-system by calling UGOPEN with the TEK4013 options item. The job will then go into a wait state until the users terminal is connected to the sub-system. The command "SHOW SYSTEMS" will list the sub-systems known to HILTEN. If the SSNAME item in UGOPEN was not used, then the sub-system name will be the same as the job name. (3) When the sub-system is known to HILTEN, the console operator may type its name to connect the terminal to the sub-system. However, if the MAGICWD item was given in UGOPEN, HILTEN will ask for the magic word before this connection is made. Once a terminal is connected to the sub-system, no other terminal will be allowed to use it.

The KBRD attention is generated by the RETURN key. When the cross-hairs are on the screen, the SPTR attention is generated by depressing a key on the keyboard, <u>not</u> the carriage return, and then (on some terminals) the carriage return. Hitting the BREAK key while UGEPUT is sending data to the screen will terminate that call to UGEPUT with no other effect on the program. If the console operator has been entering data into a keyboard input buffer and wishes to cancel what has been entered and start all over, the console operator may hit the break key to do this. In addition, the console operator may temporarily return to WYLBUR by typing $WYL and hitting the RETURN key. When finished using WYLBUR, the console operator may type the name of the sub-system to resume where the sub-system was exited.

There is a restriction which applies only to this device. In general, a program may control a number of graphic devices, including a number of interactive devices. However, only one of these devices may be a TEKTRONIX 4013 running as a HILTEN sub-system. The user is also prohibited from making any other use of HILTEN in the same program. These restrictions were dictated by HILTEN and the way the Unified Graphics System uses it.

The name of the basic device-dependent module for this device is UGTK402. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

A HILTEN sub-system can also ABEND in a number of ways. A message will usually be printed along with your JCL in these cases.

SECTION 5.3: THE SLAC EXPERIMENTAL DISPLAY CONSOLE

This interactive display console consists of a CRT, a keyboard, a light pen, a mouse with three buttons on it, and an audible alarm. The unit was built at SLAC and is located on the second floor of the Central Lab Annex. This unit can display points, lines, and four sizes of characters at a single intensity level. Lines may be drawn solid or dashed and any display item may be put in the wink mode. The display hardware for this device will support a four color CRT; unfortunately, only a black and white CRT is connected to it at present. The display hardware includes a display buffer of 4096 bytes. The picture is re-generated from this buffer at a rate of 60 times a second. The display unit is connected to an IBM SYSTEM/7 computer which is connected to the central computing facility. Communication between the central computer and the SYSTEM/7 is through the SLAC Real-Time Network [DEN-1, DEN-2, GRA-1, DEN-3].

This device is selected by the options item SLACXTV in subroutine UGOPEN. The only standard item which will be processed by UGOPEN is MAXENUM. In addition, UGOPEN will recognize the items MAXESIZ=<value>, RTRD=<value>, and RTWD=<value>. MAXESIZ gives the maximum number of bytes that may be in an element after it has been translated to device orders; the default value is 4096. RTRD and RTWD give the names of the data read and data write routes in the Real-Time Network; their default values are UGRDDATA and UGWTDATA, respectively. The line structure parameters DDSH and DOTS will result in dashed lines being drawn. The reason for this is that breaking a line up into dot-dashes or dots by software would result in a very significant increase in the amount of data put into the display file. The character sizes available on this device are:

VSML	SPACING=0.00782	128 Characters/Line
SMAL	SPACING=0.01564	64 Characters/Line
LARG	SPACING=0.02346	42 Characters/Line
VLRG	SPACING=0.03128	32 Characters/Line

The programmer and device scaling and scissoring limits are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1023 \\ 0 & 1023 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device, IARRAY(2) will contain the number of bytes in the largest graphic element after it has been translated to device-dependent orders, IARRAY(3) will contain the largest number of elements that were present in a single picture, IARRAY(4) will contain the number of bytes in the largest display file, and IARRAY(5) will contain the number of bytes in the last graphic element after it has been translated to device-dependent orders. Other entries in IARRAY will be zero.

Graphic elements may be fully manipulated on this device. The attentions KBRD, LPEN, and SPTR may be utilized. An arbitrary number of keyboard input buffers may be in a single display file and they should consist of a multiple of four characters. The LPEN attention returns the identification of the element being

pointed at and a byte offset into the element but does not return the X and Y coordinates of the item being selected. The non-detectability of elements on this device is produced by software; therefore, detectable elements should be well separated from other elements on the screen, even if the other elements are non-detectable. Enabling the SPTR attention causes a cursor for the mouse to be put on the screen; the cursor is removed when SPTR is disabled. The SPTR attention returns the button number that was pushed on the mouse in addition to giving the X and Y coordinates of the mouse cursor on the screen; the buttons are numbered 4, 2, 1 from left to right.

The setup required for using this device is quite complicated. The setup and JCL for running a minimal job which does not use the Real-Time Network for anything but communicating with the graphic device is:

```
//SETUP    EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DSNAME=&&TPDS,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(TRK,(1,,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSIN     DD *
./        ADD    NAME=UGRTDATA
RTCMD RTLOG: RNAME UGMAIN=31
RTCMD RTLOG: RNAME UGS7CMND=30
RTCMD RTLOG: RNAME UGS7DATA=29
RTCMD RTLOG: RNAME UGRDDATA=63
RTCMD RTLOG: RNAME UGWTCMND=62
RTCMD RTLOG: RNAME UGWTDATA=61
RTCMD RTLOG: ROUTE UGRDDATA NOSAMPLE NODE=UGMAIN
RTCMD RTLOG: ROUTE UGWTCMND NOSAMPLE NODE=UGS7CMND
RTCMD RTLOG: ROUTE UGWTDATA NOSAMPLE NODE=UGS7DATA
RTCMD RTLOG: BUFPOOL BUFSIZE=4096 BUFNO=1
RTCMD RTLOG: ATTACH userprog NODE=UGMAIN -
             NOTIFY=RTLOG PRTY=*-2
RTCMD RTLOG: ATTACH RTSYS7 NODE=(UGS7CMND,UGS7DATA) -
             NOTIFY=RTLOG PRTY=*
RTCMD RTLOG: SHOW NETWORK
./        ENDUP
//RUN      EXEC PGM=REALTIME,PARM=UGRTDATA
//STEPLIB       DD DSNAME=WYL.gg.uuu.loadmods,DISP=(OLD,PASS)
//              DD DSNAME=WYL.CG.RCB.UGRUNLIB,DISP=SHR
//              DD DSNAME=WYL.SF.RTP.LINKLIB,DISP=SHR
//SYSLINK       DD UNIT=(LINK05,,DEFER),VOLUME=SER=LINK05
//SYSSBCU       DD UNIT=SBCW3
//SYS7LIB       DD DSNAME=WYL.EA.UGG.S7.LOADLIB(UGGWCP),DISP=SHR
//RTINIT        DD DSN=&&TPDS,DISP=(OLD,PASS)
//RTLOG         DD SYSOUT=A
//FT06F001      DD SYSOUT=A
//SYSPRINT      DD SYSOUT=A
```

Where WYL.gg.uuu.loadmods is the name of a library containing the program "userprog" which will be executed. "userprog" must be an executable load module.

The keyboard contains a number of keys with a special function. The key labeled LF will move the cursor from one keyboard input buffer to another, FS advances the cursor within a buffer, and BS

backspaces it. The keyboard attention is generated by the ESC key. The light pen attention is generated by pointing at a light pen detectable element and depressing the button on the light pen. The screen pointer attention is generated by pushing one of the three buttons on the mouse.

The name of the basic device-dependent module for this device is UGSLTV2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

This section contains some examples of programs which use the Unified Graphics System. These examples include the complete JCL that is required to run the program.

SECTION 6.1: PLOTTING A MESH SURFACE IN FORTRAN

This FORTRAN program produces a perspective view of a mesh surface. The program, with its comments, is relatively self-explanatory; however, a few items should be noted. First, notice that an error processing subroutine is used to write out the contents of the array ELEMNT when it becomes full. Without this error processing subroutine, the array ELEMNT would have to be more than six times larger than its present size. Also, notice how the graphic device is selected by the program; the first thing the program does is to read a card containing the character string which is to be passed to UGOPEN. This means that this program can be changed to generate a picture on any other device simply by changing the card in the input stream and the DD card for the GO.UGDEVICE statement; absolutely no program changes are required to use a different device. The program, as shown, generates the picture on the CALCOMP Drum Plotter (Model 1136) with 10 inch paper. Multiple pens are used to produce the different intensity levels.

```
//uuuMSURF JOB 'uuu$gg'
//TEST    EXEC FORTHCG,
//             LKEDLB1='WYL.CG.RCB.UGFTNLIB',
//             GOSL2='WYL.CG.RCB.UGRUNLIB'
//FORT.SYSIN   DD *
C  SAMPLE PROGRAM:  PLOTTING A MESH SURFACE
C
      LOGICAL*1     PARM(72)
      REAL*4        PROGLN(2,2)/0.0,0.0,1.0,0.8/
      REAL*4        XBDR(11)/0.0,1.0,1.0,0.0,0.0,0.05,
     X                       0.05,0.95,0.95,0.05,0.0/
      REAL*4        YBDR(11)/0.0,0.0,0.8,0.8,0.0,0.05,
     X                       0.75,0.75,0.05,0.05,0.0/
      INTEGER*4     BBITS/ZF0000000/
      REAL*4        REFP(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        PTRAN(31)
      REAL*4        ARRAY(32,52)
      REAL*4        WKAREA(780)
      COMMON        ELEMNT(1000)
      INTEGER*4     C20(5)
```

```
      REAL*4          X,Y,R
      INTEGER*4       I,J
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  ASSURE THAT A FRESH PLOTTING AREA IS AVAILABLE, FILL
C  IN THE ARRAY WHICH DEFINES THE SURFACE, AND GENERATE
C  THE PROJECTIVE TRANSFORMATION.
      READ (5,100) PARM
  100 FORMAT(72A1)
      CALL UGOPEN(PARM,99)
      CALL UGCSET('DUPLEX*')
      CALL UGRSCL('*',PROGLN)
      CALL UGPICT('CLEAR*',0)
      DO 102 I=2,52
        X=I-27
        ARRAY(1,I)=X
        DO 101 J=2,32
          Y=J-17
          ARRAY(J,1)=Y
          R=SQRT(X**2+Y**2)
          ARRAY(J,I)=((750.0/(R**2+75.0))+5.0)*COS(0.4*R)
  101   CONTINUE
  102 CONTINUE
      CALL UGPROJ('*',REFP,VDIR,HDIR,100.0,75.0,PTRAN)
C
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 UGEINT('CLEAR*',ELEMNT,1000)
      CALL UGELNS('VBRT*',XBDR,YBDR,10,BBITS,-5,ELEMNT)
      CALL UGXHCH('*','DIMM*',XBDR,YBDR,11,
     X  WKAREA,780,ELEMNT)
      CALL UGCHAR('PTOX*','MESH SURFACE EXAMPLE',
     X                    ' LLL  LLLLLL  LLLLLL',C20,20)
      CALL UGETXT('SPACING=0.04*',0.2,0.65,C20,20,ELEMNT)
      CALL UG3DMS('UPPER*','*',ARRAY,32,52,
     X  PTRAN,WKAREA,780,ELEMNT)
      CALL UG3DMS('LOWER,NOCONN*','VDIM*',ARRAY,32,52,
     X  PTRAN,WKAREA,780,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
C
C  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C  AND THE PROGRAM RETURNS TO THE SYSTEM.
      CALL UGCLOS('*')
      RETURN
C
      END
C
C  ERROR PROCESSING SUBROUTINE
C
      SUBROUTINE UGXERR(LEVEL,NAME,INDEX)
C
      INTEGER*4  LEVEL,NAME(2),INDEX
      COMMON     ELEMNT(1000)
C
```

```
C   IF THE ERROR IS AN INDICATION THAT NO MORE SPACE IS
C   AVAILABLE IN THE GRAPHIC ELEMENT THEN THE CURRENT
C   CONTENTS OF THE ELEMENT ARE TRANSMITTED TO THE
C   GRAPHIC DEVICE, THE ELEMENT IS RE-INITIALIZED, AND
C   THE ERROR INDICATOR IS RE-SET.
        IF (INDEX.NE.9) GO TO 100
          CALL UGEPUT('*',0,ELEMNT)
          CALL UGEINT('CONTINUE*',ELEMNT,1000)
          LEVEL=0
  100 CONTINUE
        RETURN
C
        END
//GO.UGDEVICE  DD DSNAME=&&PLOT,DISP=(NEW,PASS),
//             VOLUME=SER=PLOT,UNIT=T9-1600,LABEL=(1,SL),
//             DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
//GO.SYSIN     DD *
  CALDRSM,PEN2=VDINDINM,PEN3=VBRT*
```

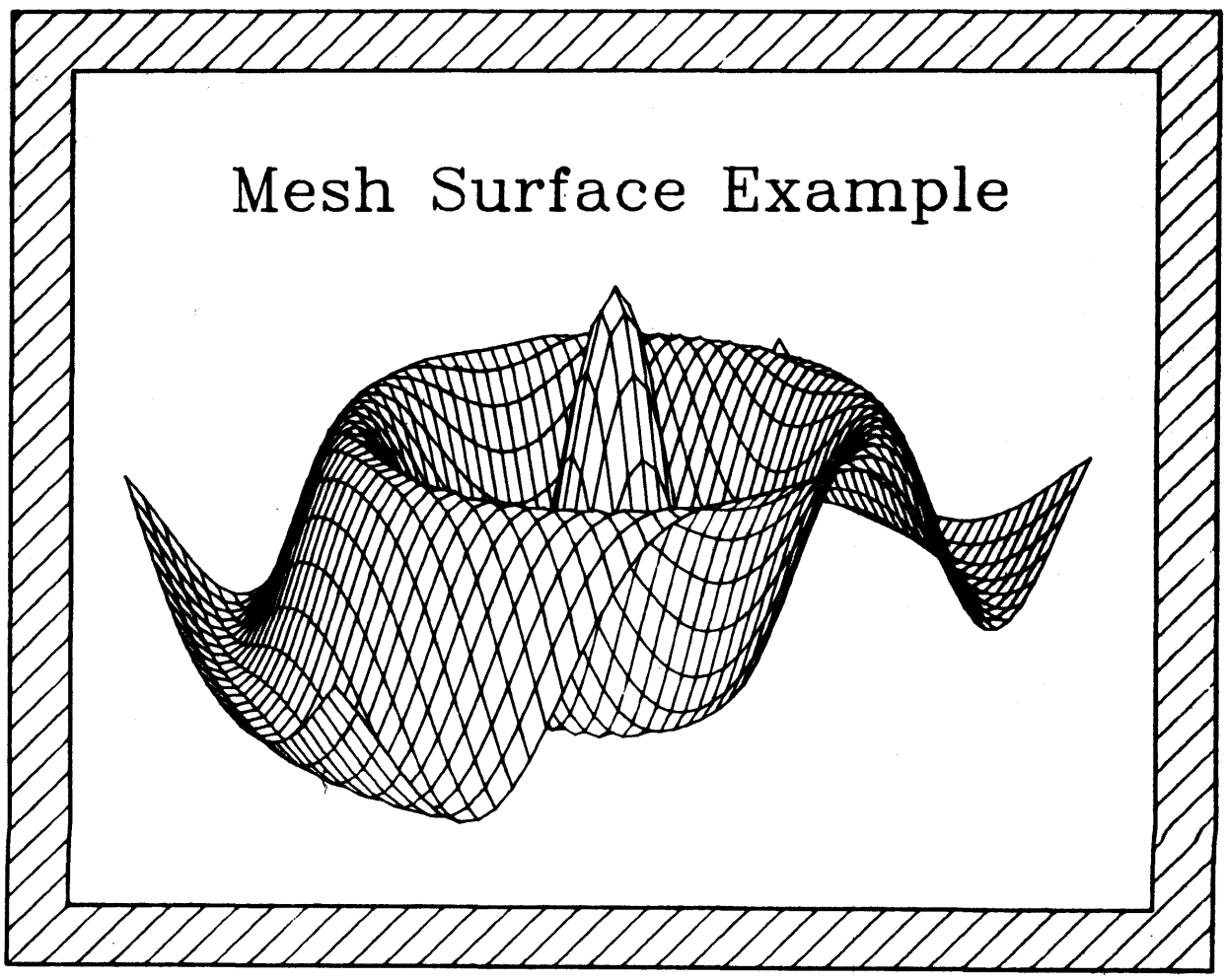


Figure 6.1.1: The Mesh Surface.

SECTION 6.2: AN INTERACTIVE DRAWING PROGRAM IN FORTRAN

The program in this section illustrates how a program can use more than one graphic device. In the program, as shown, the console operator may manipulate the cross-hairs on a TEKTRONIX 4013 to draw pictures on the CRT. At a command from the console operator, the picture can be written to the VERSATEC Electrostatic Plotter, Model 1200A. Thus, the VERSATEC is acting as a hard-copy device for the TEKTRONIX 4013. Any non-interactive device may be easily substituted for the VERSATEC by changing the proper call to UGOPEN and the DD card for GO.UGAUXDEV. Changing the program to some interactive device other than the TEKTRONIX 4013 is difficult, however, because this program utilizes the cross-hair pointer in a manner that is peculiar to the TEKTRONIX 4013.

The program first puts the picture shown in Figure 6.2.1 on the screen and enables the cross-hairs. The console operator may then position the cross-hairs and strike a key. Striking the "B" key causes the beam to move to the point designated by the cross-hairs, without drawing, while striking the "D" key causes the beam to draw to the point. "E" erases the last point from the computer memory but does not erase it from the screen. "C" erases the current picture, and "R" regenerates the current picture. "S" sends the current picture to the VERSATEC and "T" terminates the program. Striking any other key causes the display in Figure 6.2.1 to be restored, but does not modify the current picture.

```
//uuuIDRAW JOB 'uuu$gg'
//*MAIN        CLASS=I,SYSTEM=SYA
//TEST    EXEC FORTHCG,
//             LKEDLB1='WYL.CG.RCB.UGFTNLIB',
//             GOSL2='WYL.CG.RCB.UGRUNLIB'
//FORT.SYSIN   DD *
C  SAMPLE PROGRAM:  AN INTERACTIVE DRAWING PROGRAM
C
      INTEGER*4     ELEMNT(1100)
      REAL*4        XPTS(500),YPTS(500)
      INTEGER*4     BBTS(499)
      INTEGER*4     NPTS
      REAL*4        PICLIM(2,2)
      INTEGER*4     DUMLIM(2,2)
      LOGICAL*1     ATCD(4)
      INTEGER*4     IARY(8)
      REAL*4        XARY(2)
      INTEGER*4     TEM1
      LOGICAL*1     TEM2(4)
      EQUIVALENCE   (TEM1,TEM2(1))
      LOGICAL*1     B/'B'/, D/'D'/, E/'E'/, C/'C'/,
     X              R/'R'/, S/'S'/, T/'T'/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICES,
C  SET THE SCALING AND SCISSORING LIMITS FOR THE
C  AUXILIARY DEVICE, ENABLE THE SCREEN POINTER, AND SET
C  THE END POINT COUNT TO ZERO.
```

```
      CALL UGOPEN('TEK4013,FULSCR,MAGICWD=ABCD*',99)
      CALL UGSCAL('GET,PROGL*',PICLIM,DUMLIM)
      CALL UGOPEN('VEP12FF,FULSCR,DDNAME=UGAUXDEV*',98)
      CALL UGRSCL('*',PICLIM)
      CALL UGSLCT('*',99)
      CALL UGRATN('SPTR*')
      NPTS=0
C
C  GENERATE THE INSTRUCTIONS DISPLAY:  THE SCREEN IS
C  CLEARED AND THE INSTRUCTIONS ARE WRITTEN ON THE
C  SCREEN.
  100 CALL UGPICT('CLEAR*',0)
      CALL UGEINT('CLEAR*',ELEMNT,1100)
      CALL UGETXT('LARG*',0.20,0.90,
     X  'INTERACTIVE DRAWING PROGRAM',
     X  27,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.70,
     X  'THE FOLLOWING KEYS PERFORM THE LISTED ACTIONS',
     X  45,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.65,
     X  '  B...BLANK TO CURRENT CROSS-HAIR POSITION',
     X  42,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.60,
     X  '  D...DRAW TO CURRENT CROSS-HAIR POSITION',
     X  41,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.55,
     X  '  E...ERASE LAST ENTERED POINT',
     X  30,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.50,
     X  '  C...CLEAR ALL PREVIOUSLY ENTERED POINTS',
     X  41,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.45,
     X  '  R...REGENERATE THE CURRENT PICTURE',
     X  36,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.40,
     X  '  S...SAVE PICTURE ON AUXILIARY DEVICE',
     X  38,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.35,
     X  '  T...TERMINATE THE PROGRAM',
     X  27,ELEMNT)
      CALL UGETXT('LARG*',0.10,0.30,
     X  'ALL OTHER KEYS WILL RESTORE THIS DISPLAY',
     X  40,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
C
C  WAIT FOR AN ATTENTION:  THE PROGRAM GOES INTO THE
C  WAIT STATE UNTIL THE CONSOLE OPERATOR HITS A KEY
C  TO CREATE AN ATTENTION.  THE KEY IS EXAMINED AND
C  THE PROGRAM TRANSFERS TO THE PROPER SECTION.
  200 CALL UGRATN('*',-1.0,ATCD,IARY,XARY)
      TEM1=IARY(6)
      IF (TEM2(4).EQ.B) GO TO 300
      IF (TEM2(4).EQ.D) GO TO 310
      IF (TEM2(4).EQ.E) GO TO 320
      IF (TEM2(4).EQ.C) GO TO 330
      IF (TEM2(4).EQ.R) GO TO 340
      IF (TEM2(4).EQ.S) GO TO 350
```

```
      IF (TEN2(4).EQ.T) GO TO 360
      GO TO 100
C
C  PROCESS OPERATOR ACTIONS:  THE ACTION INDICATED BY
C  THE CONSOLE OPERATOR IS CARRIED OUT.
C  OPERATION:  BLANK TO CURRENT CROSS-HAIR POSITION.
  300 IF (NPTS.GE.500) GO TO 301
      NPTS=NPTS+1
      XPTS(NPTS)=XARY(1)
      YPTS(NPTS)=XARY(2)
      IF (NPTS.GT.1) BBTS(NPTS-1)=0
      GO TO 200
  301 CALL UGEINT('CLEAR*',ELEMNT,1100)
      CALL UGETXT('LARG*',0.20,0.10,
     X  'ERROR...POINT BUFFER IS FULL',
     X  28,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
      GO TO 200
C  OPERATION:  DRAW TO CURRENT CROSS-HAIR POSITION.
  310 IF (NPTS.GE.500) GO TO 301
      IF (NPTS.EQ.0) GO TO 300
      NPTS=NPTS+1
      XPTS(NPTS)=XARY(1)
      YPTS(NPTS)=XARY(2)
      BBTS(NPTS-1)=1
      CALL UGEINT('CLEAR*',ELEMNT,1100)
      CALL UGELIN('*',XPTS(NPTS-1),YPTS(NPTS-1),0,ELEMNT)
      CALL UGELIN('*',XPTS(NPTS),YPTS(NPTS),1,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
      GO TO 200
C  OPERATION:  ERASE LAST ENTERED POINT.
  320 IF (NPTS.GT.0) NPTS=NPTS-1
      GO TO 200
C  OPERATION:  CLEAR ALL PREVIOUSLY ENTERED POINTS.
  330 NPTS=0
      CALL UGPICT('CLEAR*',0)
      GO TO 200
C  OPERATION:  REGENERATE THE CURRENT PICTURE.
  340 CALL UGPICT('CLEAR*',0)
      CALL UGEINT('CLEAR*',ELEMNT,1100)
      CALL UGELNS('*',XPTS,YPTS,NPTS,BBTS,NPTS-1,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
      GO TO 200
C  OPERATION:  SAVE PICTURE ON AUXILIARY DEVICE.
  350 CALL UGSLCT('*',98)
      CALL UGPICT('CLEAR*',0)
      CALL UGEINT('CLEAR*',ELEMNT,1100)
      CALL UGELNS('*',XPTS,YPTS,NPTS,BBTS,NPTS-1,ELEMNT)
      CALL UGEPUT('*',0,ELEMNT)
      CALL UGSLCT('*',99)
      GO TO 200
C  OPERATION:  TERMINATE THE PROGRAM.
  360 CALL UGCLOS('ALL*')
      RETURN
C
      END
//GO.UGAUXDEV  DD SYSOUT=P,
```

```
//          DCB=(RECFM=FBA,LRECL=134,BLKSIZE=1340)
```

```
INTERACTIVE DRAWING PRØGRAM


THE FØLLØWING KEYS PERFØRM THE LISTED ACTIØNS
  B...BLANK TØ CURRENT CRØSS-HAIR PØSITIØN
  D...DRAW TØ CURRENT CRØSS-HAIR PØSITIØN
  E...ERASE LAST ENTERED PØINT
  C...CLEAR ALL PREVIØUSLY ENTERED PØINTS
  R...REGENERATE THE CURRENT PICTURE
  S...SAVE PICTURE ØN AUXILIARY DEVICE
  T...TERMINATE THE PRØGRAM
ALL ØTHER KEYS WILL RESTØRE THIS DISPLAY
```

Figure 6.2.1: The Initial Picture Showing the Instructions.

SECTION 6.3: A SIMPLE GRAPH PLOTTER IN PL/1

This section contains a simple PL/1 program which reads some data points and generates a graph from them. One of the things that this program illustrates is a non-trivial manipulation of the scaling limits. The programs in the preceding sections drew their pictures by transforming the data to the default scaling limits. This program transforms the scaling limits so that the data can remain in its own coordinate system.

After the output device has been opened, subroutine UGRSCL is called and a drawing space with an X range of 0.0 to 1.0 and a Y range of 0.0 to 0.8 is obtained. The titles and axes are drawn in this coordinate system. Next, the programmer coordinate system is changed by calling subroutine UGSCAL so that the limits of the data correspond to the area defined by the axes. The

given data points are drawn in this second coordinate system. Notice that this use of UGSCAL is device-independent and the resulting program could produce the same picture on any device. The only changes required to use another device is the call to UGOPEN and the DD card for GO.UGDEVICE. The program, as shown, saves the picture in a partitioned data set for later viewing on a TEKTRONIX 4013.

There are, however, two problems that have been introduced by this manipulation of the scaling and scissoring limits. The first problem is that this picture cannot be successfully saved in a partitioned data set pseudo-device. Use of that pseudo-device requires that all of the graphic elements depend on the same scaling and scissoring limits. The second problem is that the limits used to plot the data do not preserve the ratio described in the section on Scaling and Scissoring. This means that any use of the ANGLE=<value> options item for plotting text in the coordinate system of the data would be ambiguous.

```
//uuuSPLOT JOB 'uuu$gg'
//TEST    EXEC PX1CG,
//             PL1LB1='WYL.CG.RCB.UGPL1DCL',
//             PL1PRM='WEST,ATTRIBUTES,XREF,INCLUDE',
//             LKEDLB1='WYL.CG.RCB.UGPLOLIB',
//             GOSL1='WYL.CG.RCB.UGRUNLIB'
//PL1.SYSIN    DD *
  /*  SAMPLE PROGRAM:  A SIMPLE GRAPH PLOTTER  */

  GRAPH: PROCEDURE OPTIONS(MAIN);

  % INCLUDE UGEINT, UGETXT, UGELNS, UGAXIS;
  % INCLUDE UGOPEN, UGCLOS;
  % INCLUDE UGEPUT, UGPICT;
  % INCLUDE UGSCAL, UGRSCL;
  % INCLUDE UGCSET, UGCHAR;
  % INCLUDE UGDXDY;

  DECLARE PICLIM(2,2) FLOAT BINARY(21)
          INITIAL(0.0,1.0,0.0,0.8);
  DECLARE DATLIM(2,2) FLOAT BINARY(21);
  DECLARE DUMLIM(2,2) FIXED BINARY(31,0);
  DECLARE (TTLPRI,TTLSEC,XAXPRI,XAXSEC,YAXPRI,YAXSEC,
          EXTCHR) CHARACTER(60);
  DECLARE (XARRAY,YARRAY)(100) FLOAT BINARY(21);
  DECLARE (XDALO,YDALO,XDAHI,YDAHI) FLOAT BINARY(21);
  DECLARE (XAXLO,YAXLO,XAXHI,YAXHI) FLOAT BINARY(21);
  DECLARE ELEMENT(500) FIXED BINARY(31,0);
  DECLARE (NPTS,NLAB,I) FIXED BINARY(15,0);

  /*  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
      SELECT THE DUPLEX CHARACTER GENERATOR.  */
        CALL UGOPEN('PDS4013,FULSCR',99);
        CALL UGCSET('DUPLEX');

  /*  GET THE DATA:  FIRST THE TITLE, AXIS LABELS, AND THE X
      AND Y COORDINATES ARE READ AND THEN THE EXTENT OF THE
```

```
DATA IS DETERMINED.  */
  GET FILE(SYSIN) EDIT(TTLPRI,TTLSEC,XAXPRI,XAXSEC,
    YAXPRI,YAXSEC) (COLUMN(1),A(60));
  GET FILE(SYSIN) EDIT(NPTS) (COLUMN(1),F(10));
  GET FILE(SYSIN) EDIT((XARRAY(I),YARRAY(I)
    DO I=1 TO NPTS)) (COLUMN(1),6 F(10,4));
  XDALO=XARRAY(1);
  YDALO=YARRAY(1);
  XDAHI=XARRAY(1);
  YDAHI=YARRAY(1);
  DO I=2 TO NPTS;
    XDALO=MIN(XDALO,XARRAY(I));
    YDALO=MIN(YDALO,YARRAY(I));
    XDAHI=MAX(XDAHI,XARRAY(I));
    YDAHI=MAX(YDAHI,YARRAY(I));
  END;

/*  PLOT TITLES AND AXES:  FIRST A FRESH PLOTTING SPACE IS
    CREATED, THEN THE INITIAL SCALING AND SCISSORING LIMITS
    ARE SET UP, THEN THE ELEMENT IS CLEARED AND THE TITLES
    ARE PLOTTED FOLLOWED BY THE AXES.  FINALLY THE ELEMENT
    IS TRANSMITTED TO THE DEVICE.  */
  CALL UGPICT('CLEAR',0);
  CALL UGRSCL('',PICLIM);
  CALL UGEINT('CLEAR',ELEMENT);
  CALL UGCHAR('PTOX',TTLPRI,TTLSEC,EXTCHR);
  CALL UGETXT('SPACING=0.04',0.10,0.73,EXTCHR,
    ELEMENT);
  CALL UGCHAR('PTOX',XAXPRI,XAXSEC,EXTCHR);
  CALL UGETXT('SPACING=0.03',0.20,0.07,EXTCHR,
    ELEMENT);
  CALL UGCHAR('PTOX',YAXPRI,YAXSEC,EXTCHR);
  CALL UGETXT('SPACING=0.03,ANGLE=90',
    0.05,0.15,EXTCHR,ELEMENT);
  CALL UGDXDY('',XDALO,XDAHI,7,10,XAXLO,XAXHI,NLAB);
  CALL UGAXIS('LDEC=3','','SPACING=0.015',
    0.15,0.20,0.95,XAXLO,XAXHI,NLAB,ELEMENT);
  CALL UGDXDY('',YDALO,YDAHI,7,10,YAXLO,YAXHI,NLAB);
  CALL UGAXIS('VERT,LDEC=0','','SPACING=0.015',
    0.20,0.15,0.65,YAXLO,YAXHI,NLAB,ELEMENT);
  CALL UGEPUT('',0,ELEMENT);

/*  PLOT THE DATA:  FIRST THE SCALING AND SCISSORING LIMITS
    ARE MODIFIED SO THAT THE DATA WILL MATCH THE AXES, THEN
    THE ELEMENT IS CLEARED, THE DATA IS PLOTTED, AND THE
    ELEMENT IS TRANSMITTED.  */
  DATLIM(1,1)=XAXLO+
    (XAXHI-XAXLO)*(PICLIM(1,1)-0.20)/(0.95-0.20);
  DATLIM(1,2)=XAXLO+
    (XAXHI-XAXLO)*(PICLIM(1,2)-0.20)/(0.95-0.20);
  DATLIM(2,1)=YAXLO+
    (YAXHI-YAXLO)*(PICLIM(2,1)-0.15)/(0.65-0.15);
  DATLIM(2,2)=YAXLO+
    (YAXHI-YAXLO)*(PICLIM(2,2)-0.15)/(0.65-0.15);
  CALL UGSCAL('PUT,PROGL',DATLIM,DUMLIM);
  CALL UGEINT('CLEAR',ELEMENT);
  CALL UGELNS('',XARRAY,YARRAY,NPTS,'1'B,ELEMENT);
```

```
        CALL UGEPUT('',0,ELEMENT);

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

    END GRAPH;
  //GO.UGDEVICE  DD DSNAME=WYL.gg.uuu.PICT,DISP=(NEW,KEEP),
  //             VOLUME=SER=SCR001,UNIT=2314,
  //             SPACE=(TRK,(10,1,10),RLSE),
  //             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
  //GO.SYSIN     DD *
    THE DISCOVERY OF Y(3095)
     LL  LLLLLLLL LL G
              ENERGY (GEV)
               LLLLL   L
      CROSS SECTION (NB)
       LLLL  LLLLLL  LL
          91
```

3.08702	26.64	3.08748	23.88	3.08914	19.12
3.08934	28.29	3.08938	24.32	3.08988	18.36
3.09008	18.29	3.09020	27.48	3.09090	34.18
3.09106	27.47	3.09120	29.80	3.09134	43.95
3.09150	35.34	3.09222	126.90	3.09274	260.20
3.09278	276.67	3.09296	293.51	3.09312	279.88
3.09380	761.47	3.09384	1121.26	3.09396	1124.99
3.09414	1401.76	3.09416	1237.14	3.09418	1787.48
3.09420	2006.46	3.09424	2072.16	3.09424	2116.46
3.09426	1576.38	3.09428	1474.64	3.09428	2159.31
3.09440	1426.50	3.09448	1755.02	3.09448	1715.47
3.09450	1864.18	3.09460	1648.21	3.09464	2309.35
3.09470	2254.99	3.09470	2390.31	3.09472	2375.27
3.09476	1693.46	3.09476	2015.47	3.09478	2043.66
3.09482	1798.43	3.09484	2560.76	3.09484	1808.28
3.09490	3163.26	3.09492	2376.27	3.09494	2329.92
3.09498	2614.89	3.09498	2272.29	3.09502	2436.82
3.09504	2812.99	3.09506	1969.30	3.09510	2432.32
3.09510	2301.46	3.09510	1933.90	3.09512	2769.29
3.09512	2881.40	3.09514	3344.97	3.09520	2053.62
3.09524	1985.32	3.09526	2017.90	3.09528	2367.41
3.09542	2434.28	3.09550	2233.71	3.09576	1869.32
3.09576	2018.33	3.09580	1538.77	3.09584	1591.03
3.09592	2067.37	3.09606	1409.14	3.09606	1762.93
3.09610	1688.89	3.09610	1450.55	3.09618	1307.14
3.09618	1576.81	3.09620	1389.69	3.09628	1139.31
3.09662	1190.66	3.09678	560.21	3.09700	796.24
3.09702	580.49	3.09750	476.03	3.09778	291.48
3.09780	295.05	3.09810	331.34	3.09898	141.51
3.10082	102.40	3.10198	115.01	3.11122	56.67
3.12954	39.80				

The data used in this example was supplied in a private communication from the SLAC-LBL group working with the SPEAR storage ring at SLAC. The probable error in the measurement of cross-section is 50-350 nb near the peak. The cross-section

measurements are, therefore, only known to 2-3 significant figures and not the 6 figures that a naive reading of the data might indicate.

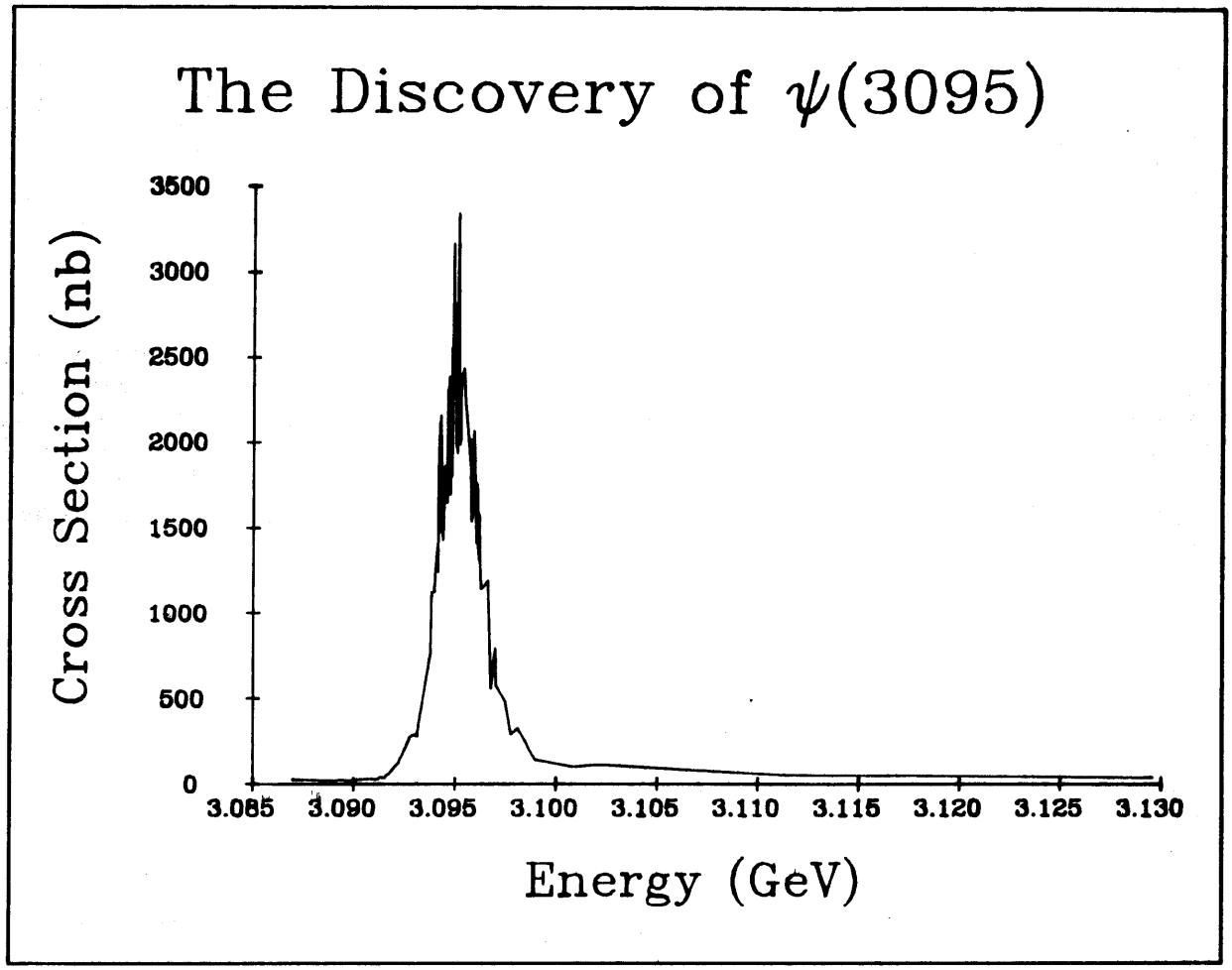


Figure 6.3.1: A Graph Produced by the Program.

SECTION 6.4: INTERACTIVE LISSAJOUS FIGURES IN PL/1

This section contains an interactive PL/1 program. 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. This picture is shown in Figure 6.4.1. 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. A typical picture is shown in Figure 6.4.2. 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. This program also illustrates a second manner in which a program can be written which does not make any references to a specific graphic device. The program, as shown, runs on an IBM 2250, but it can be changed to run on any re-fresh display console with a light pen by changing the parameter string being passed to the program and by changing the DD card for the GO.UGDEVICE statement.

Notice that this program always causes the display to be turned off when an attention occurs, and never turn 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.

```
//uuuLISAJ JOB 'uuu$gg'
//*MAIN        CLASS=I
//TEST    EXEC PX1CG,
//             PL1LB1='WYL.CG.RCB.UGPL1DCL',
//             PL1PRM='NEST,ATTRIBUTES,XREF,INCLUDE',
//             LKEDLB1='WYL.CG.RCB.UGPLOLIB',
//             GOSL1='WYL.CG.RCB.UGRUNLIB',
//             GOPRM='/IBM2250'
//PL1.SYSIN    DD *
  /***  SAMPLE PROGRAM:  INTERACTIVE LISSAJOUS FIGURES  ***/

  LISSAJ: PROCEDURE(PARM) OPTIONS(MAIN);

  DECLARE PARM CHARACTER(100) VARYING;

  % INCLUDE UGEINT, UGELIN, UGETXT, UGAXIS;
  % INCLUDE UGOPEN, UGCLOS;
  % INCLUDE UGEPUT, UGPICT;
  % INCLUDE UGEATN, UGDATN, UGRATN;
  % INCLUDE UGKPUT, UGKGET;

  DECLARE 1 ATTND,
            2 ATCODE CHARACTER(4),
            2 IARRAY(8) FIXED BINARY(15,0),
            2 XARRAY(2) FLOAT BINARY(21);
  DECLARE PARMS CHARACTER(20);
  DECLARE VPARM CHARACTER(20) VARYING;
  DECLARE XPARM CHARACTER(22);
  DECLARE ELEMENT(1000) FIXED BINARY(31,0);
  DECLARE BBIT BIT(1);
  DECLARE (A,T,X,Y) FLOAT BINARY(21);
```

```
DECLARE (M,N,MAX,GCD,GCDX,I) FIXED BINARY(15,0);

/*  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE, STOP
    THE DISPLAY, ENABLE THE LIGHT PEN, GENERATE THE TITLE,
    AND INITIALIZE THE LISSAJOUS FIGURE PARAMETERS.  */
      CALL UGOPEN(PARM,99);
      CALL UGPICT('OFF',0);
      CALL UGEATN('LPEN,OFF');
      CALL UGEINT('CLEAR',ELEMENT);
      CALL UGETXT('LARG',0.244,0.97,
        'LISSAJOUS FIGURE GENERATOR',ELEMENT);
      CALL UGELIN('',0.0,0.05,'0'B,ELEMENT);
      CALL UGELIN('',1.0,0.05,'1'B,ELEMENT);
      CALL UGEPUT('',0,ELEMENT);
STM1: M=2; N=3; A=0.25;

/*  PUT PARAMETER REQUEST ON THE SCREEN:  THE DISPLAY
    CONSISTS OF A DESCRIPTION OF THE LISSAJOUS FIGURES, A
    KEYBOARD INPUT BUFFER FOR ENTERING THE PARAMETERS, AND
    A LIGHT PEN MESSAGE FOR TERMINATING THE PROGRAM.  */
STM2: CALL UGEINT('CLEAR',ELEMENT);
      CALL UGETXT('SMAL',0.199,0.7,
        'THE EQUATION OF A LISSAJOUS FIGURE IS:',ELEMENT);
      CALL UGETXT('SMAL',0.269,0.68,
        'X=COS(2*PI*(M*T/N+A))',ELEMENT);
      CALL UGETXT('SMAL',0.269,0.66,
        'Y=COS(2*PI*T)',ELEMENT);
      CALL UGETXT('SMAL',0.199,0.5,
        'YOU MAY NOW CHANGE THE PARAMETERS M, N, AND A',
        ELEMENT);
      CALL UGETXT('SMAL',0.199,0.48,
        'AND GENERATE A KEYBOARD ATTENTION; OR YOU MAY',
        ELEMENT);
      CALL UGETXT('SMAL',0.199,0.46,
        'LIGHT PEN THE TERMINATE MESSAGE TO QUIT.',
        ELEMENT);
      CALL UGEPUT('',10,ELEMENT);
      PUT STRING(PARMS) EDIT(M,N,A) (2 F(5),F(10,4));
      CALL UGKPUT('SMAL',0.371,0.3,11,PARMS);
      CALL UGEINT('CLEAR',ELEMENT);
      CALL UGETXT('SMAL',0.7,0.025,'TERMINATE',ELEMENT);
      CALL UGEPUT('ON,DETC',1,ELEMENT);

/*  PROCESS OPERATOR ACTION:  THE KEYBOARD IS ENABLED AND
    THE PROGRAM WAITS FOR AN ATTENTION.  WHEN AN ATTENTION
    IS RECEIVED, THE KEYBOARD IS DISABLED.  A LIGHT PEN
    ATTENTION CAUSES THE PROGRAM TO TERMINATE.  A KEYBOARD
    ATTENTION CAUSES THE INPUT BUFFER TO BE READ, THE
    PICTURE TO BE CLEARED, AND THE INPUT BUFFER TO BE
    CONVERTED TO NUMERICS AND CHECKED.  THE CONVERSION ON
    CONDITION IS USED TO CHECK FOR CONVERSION ERRORS.  */
      CALL UGEATN('KBRD,OFF');
      CALL UGRATN('',-1,ATTND);
      CALL UGDATN('KBRD');
      IF ATTND.ATCODE='LPEN' THEN GO TO STM4;
      CALL UGKGET('',11,VPARM,I);
      CALL UGPICT('CLEAR',1);
```

```
        CALL UGPICT('CLEAR',11);
        CALL UGPICT('CLEAR',10);
        XPARM=VPARM||' X';
        ON CONVERSION GO TO STM1;
        GET STRING(XPARM) LIST(M,N,A);
        REVERT CONVERSION;
        IF (M<=0)|(N<=0) THEN GO TO STM1;

/*  CREATE THE LISSAJOUS FIGURE:  THE FIGURE IS GENERATED
    BY LETTING THE PARAMETER T RUN FROM ZERO TO (N DIVIDED
    BY THE GREATEST COMMON DIVISOR OF M AND N).  THEN A
    HORIZONTAL AND VERTICAL AXIS AND TWO LIGHT PEN MESSAGES
    ARE ADDED TO THE PICTURE.  */
        GCD=M; GCDX=N;
        DO WHILE(GCD¬=GCDX);
          IF GCD>GCDX THEN GCD=GCD-GCDX;
                      ELSE GCDX=GCDX-GCD;
        END;
        MAX=N/GCD;
        CALL UGEINT('CLEAR',ELEMENT); BBIT='0'B;
        DO I=1 TO 401;
          T=FLOAT((I-1)*MAX)/400;
          X=COS(6.2831853*(((M*T)/N)+A));
          Y=COS(6.2831853*T);
          CALL UGELIN('',0.4*X+0.5,0.4*Y+0.5,BBIT,ELEMENT);
          BBIT='1'B;
        END;
        CALL UGEPUT('',10,ELEMENT);
        CALL UGEINT('CLEAR',ELEMENT);
        CALL UGAXIS('HORZ,NSTH=4','','SMAL',0.5,0.1,0.9,
          -1.0,1.0,5,ELEMENT);
        CALL UGAXIS('VERT,NSTH=4','','SMAL',0.5,0.1,0.9,
          -1.0,1.0,5,ELEMENT);
        CALL UGEPUT('',11,ELEMENT);
        CALL UGEINT('CLEAR',ELEMENT);
        CALL UGETXT('SMAL',0.2,0.025,'NEW FIGURE',ELEMENT);
        CALL UGEPUT('DETC',1,ELEMENT);
        CALL UGEINT('CLEAR',ELEMENT);
        CALL UGETXT('SMAL',0.7,0.025,'DELETE AXES',ELEMENT);
        CALL UGEPUT('ON,DETC',2,ELEMENT);

/*  PROCESS OPERATOR ACTION:  THE PROGRAM WAITS FOR AN
    ATTENTION.  A LIGHT PEN DETECT ON THE 'DELETE AXES'
    MESSAGE CAUSES THE AXES ONLY TO BE DELETED, OTHERWISE
    THE ENTIRE PICTURE IS DELETED AND THE FIRST DISPLAY IS
    RESTORED.  */
STM3: CALL UGRATN('',-1,ATTND);
        CALL UGPICT('CLEAR',2);
        CALL UGPICT('CLEAR',11);
        IF ATTND.IARRAY(2)=2 THEN DO;
          CALL UGPICT('ON',0);
          GO TO STM3;
        END;
        CALL UGPICT('CLEAR',1);
        CALL UGPICT('CLEAR',10);
        GO TO STM2;
```

```
/*  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
    AND THE PROGRAM RETURNS TO THE SYSTEM.  */
STM4: CALL UGCLOS('');
      RETURN;

END LISSAJ;
//GO.UGDEVICE  DD UNIT=SCOPE1
```

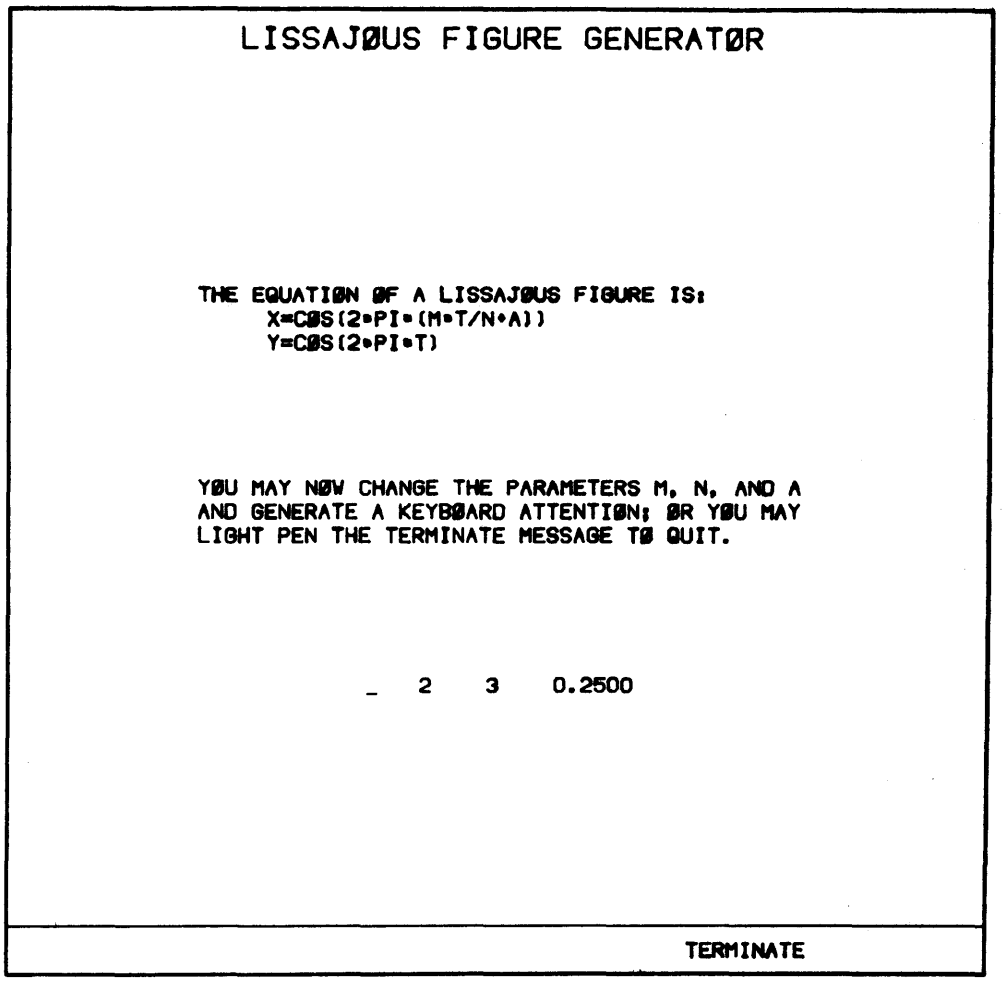


Figure 6.4.1: The First Picture Showing Suggested Parameters.

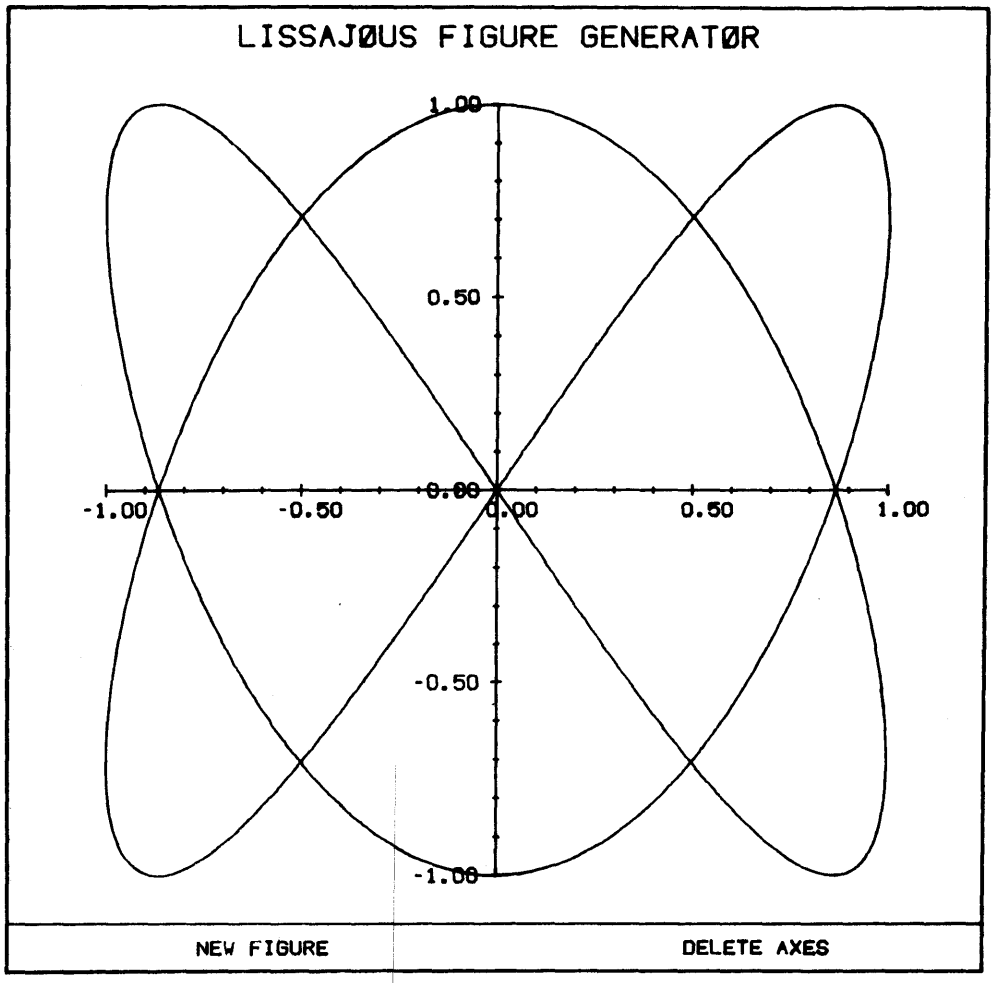


Figure 6.4.2: The Second Picture when the Suggested Parameters are Used.

This section summarizes the calling sequences for all of the FORTRAN and PL/1 callable subroutines in the Unified Graphics System.

SECTION 7.1: THE FORTRAN CALLING SEQUENCES

This sub-section summarizes the FORTRAN calling sequences.

```
CALL UGEINT(OPTIONS,ELEMENT,NELEM)
CALL UGEPNT(OPTIONS,X,Y,ELEMENT)
CALL UGELIN(OPTIONS,X,Y,BBIT,ELEMENT)
CALL UGETXT(OPTIONS,X,Y,TEXT,NTEXT,ELEMENT)
CALL UGEPTS(OPTIONS,XARRAY,YARRAY,NPTS,ELEMENT)
CALL UGELNS(OPTIONS,XARRAY,YARRAY,NPTS,BBITS,NBBITS,ELEMENT)
CALL UGAXIS(OPTIONS,LOPTIONS,TOPTIONS,INDVAR,LOVAR,HIVAR,
            LOLAB,HILAB,NLAB,ELEMENT)
CALL UGXHCH(OPTIONS,LOPTIONS,XARRAY,YARRAY,NPTS,
            WKAREA,LDIM,ELEMENT)
CALL UG3DHS(OPTIONS,LOPTIONS,ARRAY,MDIM,NDIM,TRANS,
            WKAREA,LDIM,ELEMENT)
CALL UGCNTR(OPTIONS,POPTIONS,SOPTIONS,TOPTIONS,
            ARRAY,MDIM,NDIM,CNTRLO,CNTRHI,NCNTR,
            WKAREA,LDIM,ELEMENT)

CALL UGOPEN(OPTIONS,IDENT)
CALL UGCLOS(OPTIONS)
CALL UGSLCT(OPTIONS,IDENT)
CALL UGDINF(OPTIONS,STRING,IVALUE,XVALUE)
CALL UGRINF(OPTIONS,IARRAY)
CALL UGODEV(OPTIONS,NDEV,IARRAY)

CALL UGEPUT(OPTIONS,IDENT,ELEMENT)
CALL UGPICT(OPTIONS,IDENT)
CALL UGCTRL(OPTIONS,IARRAY,XARRAY)
CALL UGEGET(OPTIONS,ELEMENT,NELEM)

CALL UGEATN(OPTIONS)
CALL UGDATN(OPTIONS)
CALL UGRATN(OPTIONS,TIME,ATCODE,IARRAY,XARRAY)

CALL UGKPUT(OPTIONS,X,Y,IDENT,STRING,NSTRING)
CALL UGKGET(OPTIONS,IDENT,STRING,MSTRING,NSTRING,ICUR)

CALL UGSCAL(OPTIONS,PROGLM,DEVCLM)
CALL UGRSCL(OPTIONS,PROGLM)
CALL UGSCIS(OPTIONS)
```

```
CALL UGCSET(OPTIONS)
CALL UGCHAR(OPTIONS,CPRI,CSEC,CEXT,NCHAR)
CALL UGCTOL(OPTIONS,TEXT,NTEXT,NSIZE,XARRAY,YARRAY,NPTS,BBITS)

CALL UGRERR(LEVEL,SNAME,INDEX)

CALL UGOPTN(OPTIONS,INDATA,EXDATA)

CALL UGDXDY(OPTIONS,LODATA,HIDATA,MINLAB,MAXLAB,
            LOLAB,HILAB,NLAB)
CALL UGPROJ(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,TRANS)
CALL UGORTH(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,TRANS)
CALL UG3TO2(TRANS,PT3D,PT2D)
CALL UGCNVF(NUMBER,FWID,FDEC,STRING)
```

SECTION 7.2: THE PL/1 CALLING SEQUENCES

This sub-section summarizes the PL/1 calling sequences and gives the ENTRY declarations for the subroutines. The calling sequences are given first.

```
CALL UGEINT(OPTIONS,ELEMENT);
CALL UGEPNT(OPTIONS,X,Y,ELEMENT);
CALL UGELIN(OPTIONS,X,Y,BBIT,ELEMENT);
CALL UGETXT(OPTIONS,X,Y,TEXT,ELEMENT);
CALL UGEPTS(OPTIONS,XARRAY,YARRAY,NPTS,ELEMENT);
CALL UGELNS(OPTIONS,XARRAY,YARRAY,NPTS,BBITS,ELEMENT);
CALL UGAXIS(OPTIONS,LOPTIONS,TOPTIONS,INDVAR,LOVAR,HIVAR,
            LOLAB,HILAB,NLAB,ELEMENT);
CALL UGXMCH(OPTIONS,LOPTIONS,XARRAY,YARRAY,NPTS,
            WKAREA,ELEMENT);
CALL UG3DMS(OPTIONS,LOPTIONS,ARRAY,TRANS,WKAREA,ELEMENT);
CALL UGCNTR(OPTIONS,POPTIONS,SOPTIONS,TOPTIONS,
            ARRAY,CNTRLO,CNTRHI,NCNTR,WKAREA,ELEMENT);

CALL UGOPEN(OPTIONS,IDENT);
CALL UGCLOS(OPTIONS);
CALL UGSLCT(OPTIONS,IDENT);
CALL UGDINF(OPTIONS,STRING,IVALUE,XVALUE);
CALL UGRINF(OPTIONS,IARRAY);
CALL UGODEV(OPTIONS,NDEV,IARRAY);

CALL UGEPUT(OPTIONS,IDENT,ELEMENT);
CALL UGPICT(OPTIONS,IDENT);
CALL UGCTRL(OPTIONS,IODATA);
  where IODATA is a structure.
CALL UGEGET(OPTIONS,ELEMENT);

CALL UGEATN(OPTIONS);
CALL UGDATN(OPTIONS);
CALL UGRATN(OPTIONS,TIME,ATTND);
  where ATTND is a structure.
```

```
CALL UGKPUT(OPTIONS,X,Y,IDENT,STRING);
CALL UGKGET(OPTIONS,IDENT,STRING,ICUR);

CALL UGSCAL(OPTIONS,PROGLN,DEVCLN);
CALL UGRSCL(OPTIONS,PROGLN);
CALL UGSCIS(OPTIONS);

CALL UGCSET(OPTIONS);
CALL UGCHAR(OPTIONS,CPRI,CSEC,CEXT);
CALL UGCTOL(OPTIONS,TEXT,XARRAY,YARRAY,NPTS,BBITS);

CALL UGRERR(LEVEL,SNAME,INDEX);

CALL UGOPTN(OPTIONS,INDATA,EXDATA);

CALL UGDXDY(OPTIONS,LODATA,HIDATA,MINLAB,MAXLAB,
            LOLAB,HILAB,NLAB);
CALL UGPROJ(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,TRANS);
CALL UGORTH(OPTIONS,REFP,VDIR,HDIR,SCRD,SCRZ,TRANS);
CALL UG3TO2(TRANS,PT3D,PT2D);
```

The following are the ENTRY declarations for the PL/1 callable subroutines.

```
DECLARE UGEINT EXTERNAL ENTRY(CHARACTER(*),
  (*) FIXED BINARY(31,0));
DECLARE UGEPNT EXTERNAL ENTRY(CHARACTER(*),
  FLOAT BINARY(21),FLOAT BINARY(21),
  (*) FIXED BINARY(31,0));
DECLARE UGELIN EXTERNAL ENTRY(CHARACTER(*),
  FLOAT BINARY(21),FLOAT BINARY(21),BIT(1),
  (*) FIXED BINARY(31,0));
DECLARE UGETXT EXTERNAL ENTRY(CHARACTER(*),
  FLOAT BINARY(21),FLOAT BINARY(21),CHARACTER(*),
  (*) FIXED BINARY(31,0));
DECLARE UGEPTS EXTERNAL ENTRY(CHARACTER(*),
  (*) FLOAT BINARY(21),(*) FLOAT BINARY(21),
  FIXED BINARY(15,0),(*) FIXED BINARY(31,0));
DECLARE UGELNS EXTERNAL ENTRY(CHARACTER(*),
  (*) FLOAT BINARY(21),(*) FLOAT BINARY(21),
  FIXED BINARY(15,0),BIT(*),(*) FIXED BINARY(31,0));
DECLARE UGAXIS EXTERNAL ENTRY(CHARACTER(*),CHARACTER(*),
  CHARACTER(*),FLOAT BINARY(21),FLOAT BINARY(21),
  FLOAT BINARY(21),FLOAT BINARY(21),FLOAT BINARY(21),
  FIXED BINARY(15,0),(*) FIXED BINARY(31,0));
DECLARE UGXHCH EXTERNAL ENTRY(CHARACTER(*),CHARACTER(*),
  (*) FLOAT BINARY(21),(*) FLOAT BINARY(21),FIXED BINARY(15,0),
  (*) FLOAT BINARY(21),(*) FIXED BINARY(31,0));
DECLARE UG3DHS EXTERNAL ENTRY(CHARACTER(*),CHARACTER(*),
  (*,*) FLOAT BINARY(21),(*) FLOAT BINARY(21),
  (*) FLOAT BINARY(21),(*) FIXED BINARY(31,0));
DECLARE UGCNTR EXTERNAL ENTRY(CHARACTER(*),CHARACTER(*),
  CHARACTER(*),CHARACTER(*),(*,*) FLOAT BINARY(21),
  FLOAT BINARY(21),FLOAT BINARY(21),FIXED BINARY(15,0),
  (*) FIXED BINARY(31,0),(*) FIXED BINARY(31,0));
```

```
DECLARE UGOPEN EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0));
DECLARE UGCLOS EXTERNAL ENTRY(CHARACTER(*));
DECLARE UGSLCT EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0));
DECLARE UGDINF EXTERNAL ENTRY(CHARACTER(*),
  CHARACTER(8),FIXED BINARY(15,0),FLOAT BINARY(21));
DECLARE UGRINF EXTERNAL ENTRY(CHARACTER(*),
  (8) FIXED BINARY(15,0));
DECLARE UGODEV EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0),(8) FIXED BINARY(15,0));

DECLARE UGEPUT EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0),(*) FIXED BINARY(31,0));
DECLARE UGPICT EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0));
DECLARE UGCTRL EXTERNAL ENTRY(CHARACTER(*),
  1,2 (8) FIXED BINARY(15,0),2 (2) FLOAT BINARY(21));
DECLARE UGEGET EXTERNAL ENTRY(CHARACTER(*),
  (*) FIXED BINARY(31,0));

DECLARE UGEATN EXTERNAL ENTRY(CHARACTER(*));
DECLARE UGDATN EXTERNAL ENTRY(CHARACTER(*));
DECLARE UGRATN EXTERNAL ENTRY(CHARACTER(*),
  FLOAT BINARY(21),1,2 CHARACTER(4),
  2 (8) FIXED BINARY(15,0),2 (2) FLOAT BINARY(21));

DECLARE UGKPUT EXTERNAL ENTRY(CHARACTER(*),
  FLOAT BINARY(21),FLOAT BINARY(21),
  FIXED BINARY(15,0),CHARACTER(*));
DECLARE UGKGET EXTERNAL ENTRY(CHARACTER(*),
  FIXED BINARY(15,0),CHARACTER(*) VARYING,
  FIXED BINARY(15,0));

DECLARE UGSCAL EXTERNAL ENTRY(CHARACTER(*),
  (2,2) FLOAT BINARY(21),(2,2) FIXED BINARY(31,0));
DECLARE UGRSCL EXTERNAL ENTRY(CHARACTER(*),
  (2,2) FLOAT BINARY(21));
DECLARE UGSCIS EXTERNAL ENTRY(CHARACTER(*));

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

DECLARE UGRERR EXTERNAL ENTRY(FIXED BINARY(15,0),
  CHARACTER(8),FIXED BINARY(15,0));

DECLARE UGOPTN EXTERNAL ENTRY(CHARACTER(*),,);

DECLARE UGDXDY EXTERNAL ENTRY(CHARACTER(*),FLOAT BINARY(21),
  FLOAT BINARY(21),FIXED BINARY(15,0),FIXED BINARY(15,0),
  FLOAT BINARY(21),FLOAT BINARY(21),FIXED BINARY(15,0));
DECLARE UGPROJ EXTERNAL ENTRY(CHARACTER(*),
  (3) FLOAT BINARY(21),(3) FLOAT BINARY(21),
```

```
  (3) FLOAT BINARY(21),FLOAT BINARY(21),FLOAT BINARY(21),
  (31) FLOAT BINARY(21));
DECLARE UGORTH EXTERNAL ENTRY(CHARACTER(*),
  (3) FLOAT BINARY(21),(3) FLOAT BINARY(21),
  (3) FLOAT BINARY(21),FLOAT BINARY(21),FLOAT BINARY(21),
  (31) FLOAT BINARY(21));
DECLARE UG3TO2 EXTERNAL ENTRY((31) FLOAT BINARY(21),
  (3) FLOAT BINARY(21),(2) FLOAT BINARY(21));
```

This section describes some subroutines and graphic devices which do not fit very well into the Unified Graphics System.

SECTION 8.1: NON-STANDARD SUBROUTINES

The subroutines listed in this section do not apply to all graphic devices, or have been prepared for some specialized purpose. The use of these subroutines should be avoided whenever possible.

SECTION 8.1.1: SUBROUTINE UGEDTA

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

The calling sequences are:

```
(FORTRAN)  CALL UGEDTA(OPTIONS,X,Y,DATA,NDATA,ELEMENT)
(PL/1)     CALL UGEDTA(OPTIONS,X,Y,DATA,ELEMENT);
```

The parameters in the calling sequences are:

OPTIONS	A character string which may contain any of the following items: SOLD,DASH,DDSH,DOTS Line structure options. VDIM,DIMM,BRIT,VBRT Intensity level options. COL1,COL2,COL3,COL4 Color options. WINK,STDY Wink mode options.
X	A floating point X coordinate associated with the data.
Y	A floating point Y coordinate associated with the data.
DATA	A character string containing the information to be added to the display element.
NDATA	The number of characters in DATA.
ELEMENT	The graphic element which will have the given information added to it.

The errors detected by this subroutine are:

1(2):	The parameter DATA contains more than 255 characters.
9(1):	Not enough room is available in the array ELEMENT to contain the new data.

SECTION 8.1.2: SUBROUTINE UGETRN

This subroutine may be used to transform a graphic element directly into device-dependent orders. At present, this subroutine is usable only when the active device type is TRN4013 or SLACISS. This subroutine performs a very specialized task and it is not needed for most graphic devices. Subroutine UGETRN makes use of the current scaling and scissoring limits and the current character stroke generator.

The calling sequences are:

```
(FORTRAN)  CALL UGETRN(OPTIONS,ELEMENT,STRING,MSTRING,
                       NSTRING)
(PL/1)     CALL UGETRN(OPTIONS,ELEMENT,STRING);
```

The parameters in the calling sequence are:

- OPTIONS — A character string which may contain any of the following items:
  - NOOPT — Do not optimize the orders for the device. Normally, the system tries to generate the minimum number of display orders to produce the picture.
- ELEMENT — The graphic element which is to be translated to device-dependent orders.
- STRING — A character string which will have the translated graphic element inserted into it. In PL/1, this character string must have the VARYING attribute.
- MSTRING — The number of characters available in STRING for the translated element.
- NSTRING — The number of characters actually stored in STRING.

The errors detected by this subroutine are:

- 1(3): No graphic device is active at present.
- 11(2): Not enough space is available in the parameter STRING to contain the translated element.
- 99(2): This subroutine is not supported for the active graphic device.

SECTION 8.1.3: SUBROUTINE UGLATN

This subroutine will return the address of the Event Control Block (ECB) which will be POST'ed when an attention becomes available for the active device. This subroutine is not normally needed by the FORTRAN or PL/1 programmer; it serves a very special purpose for some Assembler Language applications. In particular, the PL/1 user should be warned that the ECB referenced here is <u>not</u> an Event Variable.

The calling sequences are:

```
(FORTRAN)  CALL UGLATN(OPTIONS,ADDRECB)
(PL/1)     CALL UGLATN(OPTIONS,ADDRECB);
```

The parameters in the calling sequences are:

- OPTIONS — This parameter is present for consistency with the other calling sequences; no items will be recognized.

ADDRECB A full word, fixed point parameter which will be set to the address of the ECB if the device is an interactive device or which will be set to a zero if the device is not an interactive graphic device.

The errors detected by this subroutine are:
1(3): No graphic device is active at present.

SECTION 8.2: NON-STANDARD GRAPHIC DEVICES

The graphic devices listed in this section are either not generally available to the SLAC user community or are not utilized in the normal manner.

SECTION 8.2.1: THE VERSATEC PLOTTER AT THE 40" BUBBLE CHAMBER

A VERSATEC Electrostatic Printer/Plotter, Model 1100A, is connected to the NOVA Minicomputer, Model 840, in the Beam Shack (Building 233) near the 40 inch Bubble Chamber. The plotter may be used for graphic output when the NOVA computer is not being used to monitor data collection on the Bubble Chamber. The plotter uses fan-fold paper which is 11 inches wide by 8.5 inches high. The Model 1100A plotter has 100 dots per inch along a scan line and 100 scan lines per inch. The character generator on the unit cannot be used in the plot mode. Programs generating pictures for this device will write their plotting data onto a tape which must be physically carried to the NOVA. A special program on the NOVA reads the tape and writes the data to the VERSATEC plotter.

This device is selected by the options item VEP1100 in subroutine UGOPEN. The standard items which will be processed by UGOPEN are DDNAME, FULSCR, and TITLE. The device-dependent code uses characters 1-8, usually the job name, and characters 10-16, usually the bin number, from TITLE to identify the output. In addition, UGOPEN will recognize two additional items, MAXNSA=<value> and LINENX=<value>. These items control the efficiency with which the picture is produced. MAXNSA has a default value of 4096 and gives the size of the blocks of memory where the line segments which compose the picture will be saved. LINENX has a default value of 128 and is the line multiplicity, that is, the number of scan lines which will be developed concurrently. Increasing either or both of these values will increase memory requirements and reduce execution time. Points with the VDIM or DIMN option are plotted as single dots; all other points are plotted as four adjacent dots. The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 112 & 911 \\ 0 & 799 \end{bmatrix}$$

When FULSCR is used, the values are:

$$\begin{bmatrix} -0.14017 & 1.14017 \\ 0.0 & 1.0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1023 \\ 0 & 799 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

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

```
//GO.UGDEVICE  DD DSNAME=&&PLOT,DISP=(NEW,PASS),
//             VOLUME=SER=uuuuaa,UNIT=T9-800,LABEL=(1,SL),
//             DCB=(RECFM=FB,LRECL=130,BLKSIZE=4160,DEN=2)
```

For this device, all line segments are saved internally (in blocks of length MAXNSA) until the entire picture has been accumulated. When all lines are available, the picture is produced by generating a number (LINENX) of scan lines at a time. The current algorithm for doing this is reasonably efficient; however, the production of pictures for this device can require large amounts of memory.

The name of the basic device-dependent module for this device is UGVP112. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

SECTION 8.2.2: THE VERSATEC PLOTTER AT LASS

A VERSATEC Electrostatic Printer/Plotter, Model 1200A, is available at the Large Aperture Solenoid Spectrometer (LASS). The plotter uses fan-fold paper which is 11 inches wide by 8.5 inches high. The Model 1200A plotter has 200 dots per inch along a scan line and 200 scan lines per inch. The character generator on the unit cannot be used in the plot mode. The plotter is connected to a PDP-11/34 computer which is connected through a Long Line Adapter to the central computing facility. Communication between the central computer and the PDP-11/34 is through the SLAC Real-Time Network [DEN-1, DEN-2, GRA-1, DEN-3].

This device is selected by the options item VEP1200 in subroutine UGOPEN. The standard items which will be processed by UGOPEN are FULSCR and TITLE. The device-dependent code uses characters 1-8, usually the job name, and characters 10-16, usually the bin number, from TITLE to identify the output. In addition, UGOPEN will recognize four additional items: MAXNSA=<value>, LINENX=<value>, GENIL, and RTND=<value>. MAXNSA and LINENX control the efficiency with which the picture is produced. MAXNSA has a default value of 4096 and gives the size of the blocks of memory where the line segments, which compose the picture, will be saved. The line multiplicity, LINENX, has a default value of 16 and is the number of scan lines which will be

developed concurrently in the PDP-11/34. Increasing MAXMSA, when complicated pictures are being generated, can reduce execution time. LINEMX should not normally be changed; however, for very complicated pictures, the memory in the PDP-11/34 may be exceeded. In this case, LINEMX will have to be reduced. GENIL causes the Unified Graphics System to utilize the intensity level options. Normally, lines are drawn as a single string of dots and points with the VDIM or DIMM option are plotted as single dots while all other points are plotted as four adjacent dots. When GENIL is used, VDIM lines are drawn by a single line of dots, DIMM lines are composed of two adjacent lines of dots, BRIT lines are three adjacent dots, and VBRT lines are five adjacent dots. Points are drawn in a similar manner. The thicker lines produce plots that have a much better appearance and can be reproduced better. However, the GENIL option should be used only when finished plots are required. On the average, this option will cause three times as much output to be written to the device. RTWD gives the name of the data write route in the Real-Time Network, its default value is UGWTDATA. The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 268 & 1843 \\ 0 & 1575 \end{bmatrix}$$

When FULSCR is used but GENIL is not used, the values are:

$$\begin{bmatrix} -0.17016 & 1.17016 \\ 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} 0 & 2111 \\ 0 & 1575 \end{bmatrix}$$

When FULSCR and GENIL are used, the values are:

$$\begin{bmatrix} -0.16889 & 1.16889 \\ 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} 2 & 2109 \\ 0 & 1575 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero.

The setup required for using this device is quite complicated. The setup and JCL for running a minimal job which does not use the Real-Time Network for anything but communicating with the graphic device is:

```
//SETUP    EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DSNAME=&&TPDS,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(TRK,(1,,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSIN     DD *
./       ADD   NAME=UGRTDATA
RTCMD RTLOG: MNAME UGMAIN=31
RTCMD RTLOG: MNAME UGS7CMND=30
RTCMD RTLOG: MNAME UGS7DATA=29
RTCMD RTLOG: RNAME UGWTCMND=63
RTCMD RTLOG: RNAME UGWTDATA=62
RTCMD RTLOG: ROUTE UGWTCMND NOSAMPLE MODE=UGS7CMND
RTCMD RTLOG: ROUTE UGWTDATA NOSAMPLE MODE=UGS7DATA
```

```
RTCMD RTLOG: BUFPOOL BUFSIZE=512 BUFNO=1
RTCMD RTLOG: ATTACH userprog MODE=UGMAIN -
             NOTIFY=RTLOG PRTY=*-2
RTCMD RTLOG: ATTACH RTSYS7 MODE=(UGS7CMND,UGS7DATA) -
             NOTIFY=RTLOG PRTY=*
RTCMD RTLOG: SHOW NETWORK
./        ENDUP
//RUN      EXEC PGM=REALTIME,PARM=UGRTDATA
//STEPLIB       DD DSNAME=WYL.gg.uuu.loadmods,DISP=(OLD,PASS)
//              DD DSNAME=WYL.CG.RCB.UGRUNLIB,DISP=SHR
//              DD DSNAME=WYL.SF.RTP.LINKLIB,DISP=SHR
//SYSLINK       DD UNIT=(LINK09,,DEFER),VOLUME=SER=LINK09
//SYSSBCU       DD UNIT=SBCU4
//SYS7LIB       DD DSNAME=WYL.EB.MFG.VERIPL,DISP=SHR
//RTINIT        DD DSN=&&TPDS,DISP=(OLD,PASS)
//RTLOG         DD SYSOUT=A
//FT06F001      DD SYSOUT=A
//SYSPRINT      DD SYSOUT=A
```

Where WYL.gg.uuu.loadmods is the name of a library containing the program "userprog" which will be executed. "userprog" must be an executable load module.

The name of the basic device-dependent module for this device is UGVX122. In addition to the errors described with each subroutine, this module may detect errors whose description is:

1(4): You have made changes to the scaling and scissoring limits which are erroneous and invalid.

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

SECTION 8.2.3: THE TEKTRONIX 4013 DISPLAY TERMINAL

This section describes a third way to use a TEKTRONIX 4013 Display Terminal. The purpose of this method is to supply a graphic capability to existing programs which themselves call MILTEN directly. The use of this method is not encouraged; new applications should use the standard method of developing interactive programs on the TEKTRONIX 4013. In this mode, the Unified Graphics System does <u>not</u> transmit the display orders directly to the device, instead the Unified Graphics System only provides a means of generating the device-dependent orders. It is the users responsibility to transmit the device-dependent orders to the TEKTRONIX 4013. The programmer must also do any screen clearing that is necessary and must do all of the interaction between the computer program and the console operator through calls to MILTEN or some other interface.

This use of the TEKTRONIX 4013 is selected by the options item TRN4013 in subroutine UGOPEN. The only standard item which will be processed by UGOPEN is FULSCR. In addition, UGOPEN will recognize the items BAUDRATE=<value> and LRECL=<value>. BAUDRATE gives the speed at which data will be transmitted to the TEKTRONIX 4013. Its value may be any number from 300 to 9600 and its default value is 2400. Transmitting at a higher baud rate than specified can cause distorted or partial pictures. LRECL

has a default value of 168 and specifies the length of the "records" in the elements translated by UGETRN. Each record, except for the last, will consist of LRECL characters and each record is complete in itself. The last record will leave the TEKTRONIX 4013 in ALPHA mode. The reason for dividing the display orders up like this is that HILTEN can only transmit 168 bytes at a time and a sequence of TEKTRONIX 4013 orders cannot be split at any arbitrary position. Points with the VDIM or DIMM option are plotted as single dots; all other points are plotted as four adjacent dots. The character sizes available on this device are:

LARG SPACING=0.01797 56 Characters/Line(74 with FULSCR)

The programmer and device scaling and scissoring limits without FULSCR are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 122 & 901 \\ 0 & 779 \end{bmatrix}$$

When FULSCR is used, the values are:

$$\begin{bmatrix} -0.15661 & 1.15661 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1023 \\ 0 & 779 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been transmitted to the device. Other entries in IARRAY will be zero. For this device, the subroutines UGEPUT, UGPICT, and UGCTRL will simply return without performing any operations; it is only subroutine UGETRN which performs a useful service.

Since the Unified Graphics System does not do any actual I/O, no special DD statements are required to operate this device in this limited manner.

The name of the basic device-dependent module for this device is UGTRN42. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

SECTION 8.2.4: THE SLAC SLAVE SCOPES

There are a number of slave scopes attached to the XEROX SIGMA/5 computer at SPEAR. The SIGMA/5 computer is connected to the central computing facility through an IBM SYSTEM/7 computer. These display units were designed and built at SLAC. They can display points, lines, and four sizes of characters at a single intensity level. Lines may be drawn solid or dashed, and any display item may be put into the wink mode. The display hardware for this device will support a four color CRT; unfortunately, only black and white CRT's are being used at present. The display hardware includes a display buffer of 4096 bytes. The picture is regenerated from this buffer at a rate of 60 times a second.

This device is also handled quite differently than most of the other graphic devices supported by the Unified Graphics System. The Unified Graphics System does <u>not</u> transmit the display orders directly to the device, instead the Unified Graphics System only provides a means of generating the device-dependent orders. It is the user's responsibilty to transmit the device-dependent orders over the SLAC Real-Time Network [DEN-1, DEN-2, GRA-1, DEN-3] from the central computing facility to the SYSTEM/7 and then the SIGMA/5. The user must also provide programs on the SIGMA/5 to load the device-dependent orders into the display buffer.

No additional items other than SLACXSS will be recognized by UGOPEN for this device. The line structure parameters DDSM and DOTS will result in dashed lines being drawn. The reason for this is that breaking a line up into dot-dashes or dots by software would result in a very significant increase in the amount of data put into the display buffer. The character sizes available on this device are:

VSML	SPACING=0.00782	128	Characters/Line
SMAL	SPACING=0.01564	64	Characters/Line
LARG	SPACING=0.02346	42	Characters/Line
VLRG	SPACING=0.03128	32	Characters/Line

The programmer and device scaling and scissoring limits are:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1023 \\ 0 & 1023 \end{bmatrix}$$

When UGRINF is called with the GET option, IARRAY(1) will contain the number of words used in the largest graphic element which has been translated by the device-dependent code. Other entries in IARRAY will be zero. For this device, the subroutines UGEPUT, UGPICT, and UGCTRL will simply return without performing any operations; it is only subroutine UGETRN which performs a useful service.

Since the Unified Graphics System does not do any actual I/O, no special DD statements are required to operate this device in this limited manner.

The name of the basic device-dependent module for this device is UGSLSS2. In addition to the errors described with each subroutine, this module may detect errors whose description is:

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

This section contains a list of all of the publications that have been referenced in this document.

[BAR-1] J. Barlow and B. Franek, Graphic Representation of Functions of Two Variables, Rutherford High Energy Laboratory, Chilton England, Report No. RHEL/R 259 (August 1972).

[BEA-1] R. C. Beach, The SLAC Unified Graphics System, Internal Operation and Maintenance Manual, Stanford Linear Accelerator Center, Stanford California 94305, CGTM No. 171 (December 1976, Revised January 1978).

[CHA-1] R. B. Chaffee, TOP DRAWER, Stanford Linear Accelerator Center, Stanford California 94305, CGTM No. 178 (September 1976, Revised January 1978).

[CHA-2] R. B. Chaffee, Introduction to Top Drawer, Stanford Linear Accelerator Center, Stanford California 94305, CGTM No. 189 (August 1977).

[COT-1] G. Cottafava and G. Le Moli, Automatic Contour Map, Communications of the Association for Computing Machinery, Vol. 12, No. 7 (July 1969), pp. 386-391.

[DEN-1] I. Denecke, Introduction to SLAC's Real-Time System, Stanford Linear Accelerator Center, Stanford California 94305, Real-Time Note 1 (September 1974).

[DEN-2] I. Denecke and C. Granieri, Message Routing in SLAC's Real-Time System, Stanford Linear Accelerator Center, Stanford California 94305, Real-Time Note 2 (December 1974).

[DEN-3] I. Denecke, L. Hundley, and J. Wells, The Role of the SYSTEM/7 in SLAC's Real-Time System, Stanford Linear Accelerator Center, Stanford California 94305, Real-Time Note 4 (April 1975).

[DIX-1] W. J. Dixon and R. A. Kronmal, The Choice of Origin and Scale for Graphs, Journal of the Association for Computing Machinery, Vol. 12, No. 2 (April 1965), pp. 259-261.

[FAJ-1] R. Fajman and J. Borgelt, WYLBUR: An Interactive Text Editing and Remote Job Entry System, Communications of the Association for Computing Machinery, Vol. 16, No. 5 (May 1973), pp. 314-322.

[GIA-1] T. Gianno, A Mathematical Method for the Automatic Scaling of a Function, Journal of the Association for Computing Machinery, Vol. 11, No. 1 (January 1964), pp. 79-83.

[GRA-1] C. Gramieri and J. Wells, The Command Language of SLAC's Real-Time System, Stanford Linear Accelerator Center, Stanford California 94305, Real-Time Note 3 (January 1974).

[HER-1] A. V. Hershey, Calligraphy for Computers, U. S. Naval Weapons Laboratory, Dahlgren Virginia, Report No. 2101 (August 1967).

[KUB-1] B. Kubert, J. Szabo, and S. Giulieri, The Perspective Representation of Functions of Two Variables, Journal of the Association for Computing Machinery, Vol. 15, No. 2 (April 1968), pp. 193-204.

[LEW-1] C. R. Lewart, Algorithm 463: Algorithms SCALE1, SCALE2, and SCALE3 for Determination of Scales on Computer Generated Plots, Communications of the Association for Computing Machinery, Vol. 16, No. 10 (October 1973), pp. 639-640.

[LOG-1] C. A. Logg, A. M. Boyarski, A. J. Cook, and R. L. A. Cottrell, DPAK and HPAK, A Versatile Display and Histogramming Package, Stanford Linear Accelerator Center, Stanford California 94305, SLAC Report No. 196 (June 1976).

[PRU-1] M. L. Prueitt, Fantastic Computer Pictures Give Us a New Look at Numbers, Popular Science, Vol. 202, No. 2 (February 1973), pp. 102-105.

[PRU-2] M. L. Prueitt, Computer Graphics, 118 Computer-Generated Designs, Dover Publications Inc., New York 10014, (1975).

[WAT-1] S. L. Watkins, Algorithm 483: Masked Three-Dimensional Plot Program with Rotations, Communications of the Association for Computing Machinery, Vol. 17, No. 9 (September 1974).

[WIL-1] H. Williamson, Algorithm 420: Hidden-Line Plotting Program, Communications of the Association for Computing Machinery, Vol. 15, No. 2 (February 1972), pp. 100-103.

[WRI-1] T. Wright, A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables, National Center for Atmospheric Research, Boulder Colorado 80302, Report No. NCAR 72-26 (March 1972).

[WYL-1] WYLBUR/370 Reference Manual, Interactive Systems Group, Stanford Center for Information Processing, Stanford California 94305, (November 1975).

A

B

C

D

E

F

G

H

I

K

L

M

N

T

U

V

W

X

Y

Z