CGTM Number 203
August 1981

Revised:
October 1983
November 1985
October 1988
October 1990
April 1993

THE UNIFIED GRAPHICS SYSTEM FOR FORTRAN 77

PROGRAMMING MANUAL

ROBERT C. BEACH
COMPUTATION RESEARCH GROUP
STANFORD LINEAR ACCELERATOR CENTER
STANFORD, CALIFORNIA 94309

TABLE OF CONTENTS

old Ann Arbor
4013
4207

TABLE OF CONTENTS

SECTION 1: AN INTRODUCTION TO THE UNIFIED GRAPHICS SYSTEM

The Unified Graphics System is a collection of FORTRAN 77 subroutines which are designed to run on a number of computers and control any graphic device that is capable of drawing lines. This document describes versions of the system running on IBM mainframe computers and VAX computers.

The system supplies a common interface for creating pictures on any graphic device that it supports. This common interface means that programs can be written which will run with a wide variety of graphic devices. At the same time, this system will sometimes allow a programmer to take advantage of many of the special features of specific graphic devices. However, in taking advantage of these features, the programmer limits the number of graphic devices that the program can use.

The subroutines described in this document constitute the nucleus of a graphic programming package. Additional subroutines incorporating some complex algorithms are described in the Unified Graphics System Graphics Algorithm Manual [Bea81]. That document includes subroutines that make it easy for a programmer to produce projective or parallel views of three-dimensional objects, graphs of functions with linear or logarithmic scaling on the axes, contour plots, and other such pictures. It also contains a number of additional examples using the Unified Graphics System.

The Unified Graphics System is a collection of very general purpose subroutines. In any specific graphic application it is often 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 a number of packages available which use the Unified Graphics System and greatly simplify the task of producing graphs, scatter plots, and histograms. TOP DRAWER [Cha76, Cha77, and Cha79] is a very versatile program which reads data cards and produces graphs and histograms. At SLAC, TOP DRAWER is really only supported on the IBM computers but the source code is easily transported to a VAX. A group of subroutines, HANDYPAK, is described in [Boy80]. The HANDYPAK subroutines can send their output to the printer as well as the graphic devices supported by the Unified Graphics System. HANDYPAK is available on both the VAX and IBM computers. 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 1.1: CLASSIFICATION OF GRAPHIC DEVICES

The Unified Graphics System recognizes and supports a number of different classes of graphic devices. The first way that graphic devices may be classified is by the amount of interaction that is possible:

1. Non-Interactive Devices: When pictures are generated for this class of device, the pictures are written to a file that usually resides on disk. At a later time, this file is transmitted to the graphic device and the pictures are drawn.
2. Slave-Display Devices: For this type of graphic device, the pictures are written directly to the device and are drawn immediately. The interaction, if any is possible, must be done outside of the Unified Graphics System. An output-only device must always be used in the slave-display mode.
3. Interactive Devices: The pictures are written directly to the display device and are drawn immediately. A full set of interactive control subroutines is available.

Some graphic devices fall into one and only one of the above classes, while other graphic devices may be utilized in more than one way.

When a simple terminal is the graphic device, it may appear that there is little difference between the slave-display and interactive modes. This is not the case. In the slave-display mode, the application program has as much control as possible over the graphic device. Interaction may therefore be accomplished with simple FORTRAN I/O statements. In the interactive mode, the Unified Graphics requires complete control over the device and FORTRAN I/O to the terminal may cause trouble.

The second way that graphic devices can be classified is by the properties of the drawing medium. Most of these distinctions apply only to slave-display and interactive devices. The three classes supported by the Unified Graphics System and their properties are:

1. Storage Display Devices: Once a picture is written to this type of device, no partial picture erasure is possible. Non-interactive devices are best thought of as belonging to this class.
2. Raster-Scan Display Devices: This class is very similar to a storage display device except that parts of a picture may be deleted by drawing over a section of the picture to erase it. Rectangular areas may also be erased without affecting other parts of the display.
3. Refresh Display Devices: A description of the picture is retained in a local memory buffer in the form of its constituent lines and characters. The picture is refreshed from this buffer a number of times a second. It is typical of this class of display device that quite a bit of picture manipulation is possible; parts

of a picture may be named and, temporarily or permanently, deleted from a picture. Light pens are often available with this type of device and the detectability of picture items may be manipulated.

SECTION 1.2: A BRIEF DESCRIPTION OF THE SUBROUTINES

This section and the following sub-sections are 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 subroutine will be found in Section 2.

One of the common features of 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 optional information is usually used to change the default action of the subroutine or to specify device-dependent parameters.

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 of full word integers is allocated in the program.
2. The array is initialized by calling subroutine UGINIT.
3. Other subroutines, UGMARK, UGLINE, UGTEXT, etc., are called to pack the descriptions of marks, lines, text material, etc., into the array.

The information which has been packed into an array in this manner is called a "graphic segment". A later section will describe a subroutine, UGWRIT, which will transmit a graphic segment to a graphic device. A complete picture consists of one or more graphic segments. In essence, a graphic segment 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-dependent orders and transmitted to the graphic device.

The programmer specifies the positions of marks or points, end points of straight line segments, and centers of characters by giving floating point X and Y coordinates in the "world coordinate system". The default "window" on this system is a square area with coordinates of (0.0,0.0) at its lower left

corner and coordinates of (1.0,1.0) at its upper right corner. The picture is clipped at the outer boundary of this window. Subroutines are provided which allow the programmer to change this window and use a rectangular instead of a square drawing space. The subroutines that allow a programmer to manipulate the drawing space and window are described in a later section in this Introduction and in the section on Controlling the Viewing Window in Section 2. For the moment, however, we shall assume that the default drawing space and window are being used.

An example of statements which allocate and initialize a graphic segment are:

```
      INTEGER*4     SEGMNT(500)
        ...
      CALL UGINIT('CLEAR',SEGMNT,500)
```

The first argument of UGINIT 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 segment, the programmer may write:

```
      CALL UGMARK('BRIGHT',0.1,0.2,SEGMNT)
```

When the graphic segment 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, the intensity level item is ignored. This subroutine can also draw things other than points. A total of 10 additional marks are available which are selected by the digits 0 through 9. Thus the statement:

```
      CALL UGMARK('MARK=2,SIZE=0.01',0.1,0.2,SEGMNT)
```

draws a diamond shaped plotting symbol centered about the given coordinates. Its size is about 0.01 units in the world coordinate system.

A simple way to specify lines is to give the end points, one at a time, to UGLINE 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 may write:

```
      CALL UGLINE(' ',0.3,0.4,0,SEGMNT)
      CALL UGLINE(' ',0.5,0.6,1,SEGMNT)
```

The fourth argument is the blanking bit; a zero means do not draw when moving to the given point, and a one means draw a straight line. If, immediately after these two lines of code, the programmer writes:

```
      CALL UGLINE(' ',0.7,0.8,1,SEGMNT)
```

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

In addition to subroutines UGMARK and UGLINE which supply a single mark or line end point at a time, there are subroutines which supply a group of marks or line end points at once. These subroutines are called the "polymarker" and "polyline" subroutines. Suppose the arrays XARRAY and YARRAY contain the X and Y coordinates of points. Then the statement:

```
      CALL UGPMRK(' ',XARRAY,YARRAY,NPTS,SEGMNT)
```

will add the NPTS marks in the arrays to the graphic segment. The statement:

```
      CALL UGPLIN(' ',XARRAY,YARRAY,NPTS,1,1,SEGMNT)
```

will draw lines from the first point to the second, from the second point to the third, etc. The fifth and sixth arguments in the example force line segments 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.

A programmer may add the description of character data to a graphic segment by calling subroutine UGTEXT. The specification of text is more complex than that of marks and lines because of the great diversity of graphic devices in this area. Some graphic devices do not have hardware character generators, while others have character generators of great versatility.

The first thing that must be done is to define the "basic character set" of the Unified Graphics System. This character set consists of upper case Roman letters, the numerals, and the special characters: blank, plus sign, minus sign, asterisk, slash mark, equals sign, period, comma, left parenthesis, and right parenthesis. These characters are exactly those common to the 8-bit EBCDIC, 7-bit ASCII, and 6-bit BCD character codes, and it is only these characters that UGTEXT is guaranteed to process correctly.

Consider the following statement:

```
      CALL UGTEXT(' ',0.1,0.2,'AAA',SEGMNT)
```

Since the first argument is null in this statement, the characters will be drawn at a default size. If the graphic device has a hardware character generator, it will probably be used; if the device does not have a hardware character generator, the characters will be produced by a programmed character generator out of short line segments called "strokes". 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 argument gives the character string. Now consider the statements:

```
      CALL UGTEXT('SIZE=0.02,HARDGN',0.3,0.4,'AB',SEGMNT)
      CALL UGTEXT('DSIZE=0.02,SOFTGN',0.5,0.6,'AB',SEGMNT)
```

In the first statement, the SIZE item gives the spacing between the centers of consecutive characters. The HARDGN item says that the hardware character generator is to be used even if it cannot match the 0.02 size very well. In the second statement, the DSIZE item gives the character spacing relative to the default drawing space instead of the current window. This provides a means of specifying character size that is independent of the current window. If you are using the default drawing space and window, then SIZE and DSIZE produce the same result. The SOFTGN item says that the characters must be produced by the programmed stroke generator. The first argument can also be used to specify other options like intensity levels and orientation of the characters. The orientation item is ANGLE=<value> where <value>

is the angle, in degrees, that the characters make with the horizontal in a counter-clockwise direction. While the X and Y coordinates usually specify the center of the leftmost character, it is also possible to have the X and Y values specify the center of the rightmost character or the center of the entire string. This feature is very useful in centering titles or positioning labels near tic marks on the axes.

Another subroutine, UGXTXT, is also supplied for adding text to a graphic segment. In this subroutine, two text character strings are given, a "primary" character string, and a "secondary" string. UGXTXT is capable of drawing a very large number of characters and has a flexible subscripting and superscripting ability. For example, consider the statement:

```
      CALL UGXTXT('SIZE=0.01',0.1,0.2,'ABC',
     X                                ' LG',SEGMNT)
```

This statement will plot three characters starting at the given position: an upper case A, a lower case B (because the second secondary character is L), and a lower case Greek Xi (because the third secondary character is G). The characters produced by UGXTXT are always drawn by a programmed stroke generator and never by the hardware character generator.

When the characters supplied to either UGTEXT or UGXTXT appear on the graphic device, they will always be drawn at their normal aspect ratio. That is, they will not be stretched or compressed in the horizontal of vertical direction, and this is true whether the characters are produced by the hardware character generator or a programmed character generator. It is, however, still possible to produce characters that are transformed in any way the user desires. A subroutine, UGCTOL, is provided which takes a pair of character strings similar to UGXTXT and returns the X and Y coordinates of the ends of the strokes in the characters. The user may then transform these coordinates and then call UGPLIN to add the line segments to the graphic segment. For example, when the default window is in effect, the statements:

```
      REAL          XARRAY(64),YARRAY(64)
      INTEGER       NCOORD,BBITS(2)
        ...
      CALL UGCTOL('SIZE=0.01',0.1,0.2,'ABC',' LG',
     X            64,XARRAY,YARRAY,NCOORD,BBITS)
      CALL UGPLIN(' ',XARRAY,YARRAY,NCOORD,BBITS,-NCOORD,SEGMNT)
```

will generate the same display as the earlier call to UGXTXT. The principal difference is that the graphic segment must be larger to hold the line segments instead of the character strings.

In addition to the two-dimensional graphic primitives that have been described so far, the Unified Graphics System also supports three-dimensional points, lines, and character strings. This facility will be described in a later section of this Introduction and throughout the rest of this document.

On color graphic devices, it is often possible to draw polygonal areas of solid color. If a polygon consists of NPTS vertices and

these vertices are stored in the arrays XARRAY and YARRAY, then the statement:

```
CALL UGPFIL('RED',XARRAY,YARRAY,NPTS,SEGMNT)
```

will draw the polygon with the interior filled with the color red. On graphic devices that cannot perform the polygon-fill operation, the polygon will usually just be outlined. Programs that make extensive use of polygon-fill areas will probably not work very satisfactorily on devices which do not support this feature. Since the action on a device without the feature does not clear the interior of the polygon, the resulting picture can be quite different.

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 appear 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 Unified Graphics System error processor to identify this problem and cause the picture data to be transmitted to the graphic device whenever the array becomes full.

SECTION 1.2.2: NON-INTERACTIVE DISPLAY DEVICES

Subroutine UGOPEN is used to initialize a graphic device. It should usually 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. Consider the statement:

```
CALL UGOPEN('SEQ4010',99)
```

The options item, SEQ4010, specifies that the Unified Graphics System is to generate pictures for a TEKTRONIX 4010 series device and save the pictures in a sequential file. The second argument in UGOPEN is the graphic device identification and is only important when more than one graphic device is being used by the program. The options list can also be used to supply a number of other parameters, some of them device-dependent.

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 statement always signals that a new picture is to be started. On a TEKTRONIX 4010, the screen is cleared. On a mechanical device like a drum plotter, the pen is moved over to a fresh drawing area.

The subroutine which adds a graphic segment to the current picture is UGWRIT. The statement:

```
CALL UGWRIT(' ',0,SEGMNT)
```

will take the graphic data in the array SEGMNT, transform it to device-dependent orders, and transmit them to the device. After subroutine UGWRIT has been called, the array SEGMNT may be reused. SEGMNT should be re-initialized by calling subroutine UGINIT before any new graphic items are added to it.

The second argument in both UGPICT and UGWRIT have meaning only for some slave-display and interactive devices. For non-interactive devices, they should normally be zero.

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 Unified Graphics System to write out any final buffers of graphic data that are necessary and terminate the use of the graphic device.

SECTION 1.2.3: SLAVE-DISPLAY DEVICES

All of the statements in the preceding two sections remain true of slave-display devices. In addition, the programmer must be aware of the possibility that the graphic device could be a raster-scan device or a refresh display device.

If the graphic device is a raster-scan device, a graphic segment may be erased from the screen by the statement:

```
      CALL UGWRIT('ERASE',0,SEGMNT)
```

The segment is erased by retracing the segment in the erase mode. The array SEGMNT could be regenerated immediately before this statement, or the programmer could have saved the contents of the array when it was initially created. The choice is one of the many trade-offs between execution time and memory space that are available to the programmer.

Display control is more complex for refresh display devices. On such devices, each graphic segment can be the subject of extensive manipulation. For example, the statement:

```
      CALL UGWRIT(' ',7,SEGMNT)
```

gives the graphic segment an identification of 7. This numeric identification is the name by which the programmer will refer to the graphic segment when it is to be manipulated. If a second graphic segment with the identification of 7 is transmitted, and the graphic device is a refresh display device, then the new segment replaces the first one. Graphic segments that are transmitted with a zero identification are considered to be unidentified. A picture may contain many unidentified graphic segments but they cannot be manipulated.

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

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

will change segment 7 and put it into the "omit" state. A segment in the omit state will not appear on the display screen. The omit option could also have been specified in subroutine UGWRIT. It can be put back into the "include" state by the statement:

```
      CALL UGPICT('INCLUDE',7)
```

This statement causes the segment to reappear on the display screen. Thus, this include-omit switching can be used to temporarily blank out a segment and then restore it without regenerating it with subroutines UGMARK, etc. and then retransmitting it with subroutine UGWRIT. Finally, the statement:

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

will delete segment 7 and only segment 7 from the picture.

SECTION 1.2.4: INTERACTIVE GRAPHIC DEVICES

Interactive graphic devices have "control units" that may be manipulated by the operator of the graphic device. Examples of control units are keyboards, light pens, and a mouse. A group of subroutines are provided in order that the program may detect the actions of the graphic device operator.

Before a control unit may be used, it must be "enabled". After a control unit has been enabled, the operator may use the unit and the program may detect this use. The keyboard is enabled by the statement:

```
      CALL UGENAB('KEYBOARD')
```

Once the keyboard is enabled, an input buffer will be put on the screen. The program may then enter a wait state and wait for the operator to react. This is done by the statement:

```
      CALL UGEVNT(' ',-1.0,STRING,IARRAY,XARRAY,YARRAY)
```

where STRING is a character string, IARRAY is an integer array, and XARRAY and YARRAY are floating point arrays. At this time, the graphic device operator may enter characters with the keyboard and then hit the "carriage return" or "enter" key. Striking this key causes an "event" to occur, and subroutine UGEVNT will return. For a keyboard event, IARRAY(1) will be set to one, IARRAY(2) will contain the number of characters entered, and STRING will contain the characters.

Light pens operate similarly to a keyboard but are only found on refresh display devices. A light pen is a "pick" control unit; that is, it may select an item that is being displayed on the screen. When a graphic segment is transmitted to a graphic device with UGWRIT, it may be put into the pick state. Only those graphic segments in the pick state can be selected by a pick device. For example,

```
      CALL UGWRIT('PICK',8,SEGMNT)
```

creates a segment with an identification of 8 and puts it into the pick state instead of the default "nopick" state. UGPICT may manipulate the pick-nopick state of a graphic segment. If the pick control unit is enabled by:

```
      CALL UGENAB('PICK')
```

and UGEVNT is called to enter the wait state, the graphic device operator may then use the pick device to select any of the graphic segments that are in the pick state. When the operator closes a switch on the pick device, an event is created and control is returned to the user's program. A pick event causes IARRAY(1) to be set to two and IARRAY(2) to contain the identification of the selected graphic segment. In addition, IARRAY(3) will contain the pick identification of the selected graphic item within the graphic segment. The pick identification of a graphic item is defined when the graphic item is put into a graphic segment. For example, the statement:

```
      CALL UGMARK('PICKID=17',0.1,0.2,SEGMNT)
```

defines a mark with a pick identification of 17 and this value will be returned in IARRAY(3) if this mark is selected with the pick device. If the PICKID option is not used, the graphic item will usually have a pick identification of zero.

As supported by the Unified Graphics System, a control unit like a mouse is a different sort of device. It does not generate events, but may be read at any time after it has been enabled. A mouse is a "locator" control unit and is enabled by:

```
      CALL UGENAB('LOCATOR')
```

It is read by the statement:

```
      CALL UGECTL('LOCATOR',STRING,IARRAY,XARRAY,YARRAY)
```

This subroutine will wait for the graphic device operator to signal that the locator is positioned. When UGECTL returns, XARRAY(1) and YARRAY(1) will contain the coordinates, in the current window, of the selected point.

Actually, a mouse can also be used as a pick control unit on some graphic devices. The significant difference between a pick and locator control unit is that a pick returns the identification of something being displayed on the screen while a locator returns a position on the screen. A pick answers the question "what" while a locator answers the question "where".

Control units may be disabled by calling subroutine UGDSAB. The control units described above will all be disabled by:

```
      CALL UGDSAB('KEYBOARD,PICK,LOCATOR')
```

Additional control units are described in the section on Graphic Device Input Control in Section 2.

SECTION 1.2.5: THE DRAWING SPACE, WINDOWS, AND VIEW PORTS

The previous sections avoided the details of how the coordinate system of the window and the drawing space may be manipulated. Instead, they assumed that the default drawing space and window were in effect. This section will discuss the other alternatives.

To do this, some concepts must be clarified and others defined. The "drawing space" is the area on the graphic device that is available for use. The drawing space has a coordinate system imposed on it so that a user can refer to parts of it. The coordinate system of the graphic items is the "world coordinate system", and this coordinate system is potentially distinct from the coordinate system of the drawing space. The relation between the two is that a rectangular subset of the world coordinate system (the "window") is mapped onto a rectangular subset of the drawing space (the "view port"). Any graphic item (lines, points, or text) that extends out of the window will be scissored at the window boundaries.

The drawing space, and therefore the aspect ratio of the picture, are selected with subroutine UGDSPC. This subroutine is usually called immediately after UGPICT has been called to start a new picture and must be called before the first graphic segment of a picture is sent to the graphic device. Suppose a picture is to be created that is three times longer in the X direction than it is in the Y direction. A call to UGDSPC that accomplishes this is:

```
      CALL UGDSPC('PUT',30.0,10.0,1.0)
```

The first argument has the PUT option because this subroutine may also be used to retrieve the values defining the current drawing space; the fourth argument must be unity in this case but will not otherwise be described here. It is the second and third arguments which select the drawing space and specify the coordinate system to be imposed on it. In this case, the X coordinate runs from 0.0 to 30.0 and the Y coordinate runs from 0.0 to 10.0. The units per centimeter are the same in both directions so the picture has a three to one aspect ratio. On a graphic device like a drum plotter that has an X direction that is extendable, this picture will simply extend in that direction. On a graphic device with a nearly square screen, the picture will only use a central area with the given aspect ratio. On either device, the picture will be complete and have the same appearance except for its size.

After UGDSPC has been called to generate a new drawing space, a default window and view port are in effect. The default view port coincides with the drawing space and the default window has the same limits as the drawing space or view port. This default window and view port may be changed by a call to UGWDOW. For example, if the previous call to UGDSPC was followed by:

```
      REAL          VPRT(2,2), WDOW(2,2)
      DATA          VPRT/20.0,0.0,30.0,10.0/
      DATA          WDOW/100.0,-50.0,500.0,50.0/
        ...
      CALL UGWDOW('PUT',VPRT,WDOW)
```

then the view port is the right one-third of the drawing space which starts at an X coordinate of 20.0 and extends to an X coordinate of 30.0. The window extends from an X coordinate of 100.0 to 500.0, while the Y coordinate extends from -50.0 to 50.0. Thus, if a programmer wanted to plot a point in the center of the rightmost third of the drawing space, the proper

coordinates are now (300.0, 0.0). Examples of how this facility may be used will be found in the section on Controlling the Viewing Window in Section 2. The window and view port may be redefined as many times as needed within a picture: the only thing that the programmer must remember is that the current window and view port are not used until a graphic segment is sent to the graphic device with subroutine UGWRIT. When a new picture is started, the drawing space is restored to the state of the last call to UGDSPC; any windows in effect are deleted and the default window and view port are restored.

On some raster-scan devices, the statement:

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

will erase everything in the current window and view port but will not erase any graphic item that is outside the view port.

In addition to all of the above, "shields" may be defined in the window. A shield is a rectangular subset of the window. Whenever a point or line segment is within a shield, it is scissored so that the part within the shield is eliminated. As many as four shields may be active at once. Any existing shields are deleted when a new picture is begun or when a new window and view port are defined. Examples of the use of shields will be found in the first example in Section 5.

SECTION 1.2.6: THREE-DIMENSIONAL GRAPHIC DEVICES

There are some high performance graphic devices that maintain the display file in a three-dimensional form and allow the console operator to rotate, translate, and zoom the image being displayed. The Unified Graphics System will support certain functions of these devices. However, the available hardware varies greatly, and the user should be aware that any program that uses this three-dimensional manipulation ability will probably be device-dependent.

The X, Y, and Z coordinates of three-dimensional points, lines, and text strings are given in the "three-dimensional world coordinate system". A subset of the three-dimensional world coordinate system, the "object volume", is projected onto a "three-dimensional view port" to form the final picture. On true three-dimensional graphic devices, the console operator may modify the projection parameters. The three-dimensional world coordinate system and three-dimensional view port are distinct from the two-dimensional world coordinate system and two-dimensional view port. Users of a two-dimensional graphic device may simply ignore all references to three-dimensional primitives and functions and not get into trouble. On the other hand, the entire three-dimensional ability is controlled by only seven additional subroutines.

There are five subroutines to define three-dimensional primitives and add them to a graphic segment. Subroutines UG3MRK and UG3LIN

are the three-dimensional analogues of subroutine UGMARK and UGLINE; they define a three-dimensional mark and end point of a three-dimensional line respectively. UG3PMK and UG3PLN are the three-dimensional polymarker and polyline subroutines. Finally, subroutine UG3TXT is similar to UGTEXT; it defines a text string which is positioned in three-dimensional space. As the lines and marks are moved, the text defined by this subroutine will move with them. The strings defined by UG3TXT will always be produced by a hardware character generator on a three-dimensional device.

The character strings drawn with UG3TXT will always be displayed as if they were in a plane perpendicular to the viewing direction. That is, the characters themselves do not rotate, only the position of the characters rotates. If characters that fully take part in the rotations are needed, they can be created using subroutine UGCTOL followed by a call to UG3PLN with the third coordinate supplied. To aid in this, subroutines UG3PMK and UG3PLN both recognize the options XCONST, YCONST, and ZCONST. These options mean that the indicated variable is constant and the resulting polymarker or polyline lies in a plane parallel to a coordinate axis. When XCONST is used, for example, then a single X value may be given instead of an array of X values.

There are two subroutines, UG3WRD and UG3TRN, which are used to control the manner in which the three-dimensional world coordinate system is projected onto the two-dimensional screen of the graphic device. One of the functions of UG3WRD is to define the three-dimensional view port where this image will appear. The three-dimensional view port is a subset of the drawing space that is independent of the two-dimensional view port. Subroutine UG3TRN defines an eye point and viewer orientation to produce the actual three-dimensions to two-dimensions transformation. On some graphic devices, subroutine UG3TRN may be used to read the current projection parameters, as modified by the console operator, back into the host computer. One use of this ability is to then produce a copy of the picture on the screen of the three-dimensional device on another graphic device. Thus, any other graphic device may act as a hard-copy unit for the three-dimensional device if the programmer so wishes.

A program that only uses two-dimensional primitives may, of course, be run on a graphic device with three-dimensional ability. The program will run the same as it would have on a two-dimensional device.

When a program that defines three-dimensional data runs on a two-dimensional device, the three-dimensional data is transformed to two-dimensions using the values given by UG3WRD and UG3TRN. The view of the three-dimensional data will be the initial view of the data that a user of a three-dimensional device would see. In this case, the console operator will not be able to transform the picture locally. This, however, does mean that a program that uses three-dimensional primitives can undergo initial checkout on a two-dimensional device.

When a program that defines three-dimensional data is run on a three-dimensional device, the two-dimensional data appears as it would on any device, while the three-dimensional data is initially displayed according to the parameters given in subroutines UG3WRD and UG3TRN. The console operator may then manipulate the image. Normally, this means that the three-dimensional image may be transformed by rotating about any of three axes, translating along any of three axes, and zooming the image to make it larger or smaller. On most devices, there will also be a simple way to return to the initial projection.

On some three-dimensional graphic devices, subroutine UG3TRN may be called after a picture is on the screen to change the viewing parameters. Thus, both the console operator and the application program can select the current view of the three-dimensional data. Calling UG3TRN on a two-dimensional device after a picture has been transmitted will have no effect.

SECTION 2: A DETAILED DESCRIPTION OF THE SUBROUTINES

This section gives a complete description of each of the subroutines provided by the Unified Graphics System. The names of these subroutines, and most other external names, start with the letters "UG", standing, of course, for Unified Graphics; it's not an approved opinion. If the user of these subroutines avoids external names beginning with "UG", naming conflicts between the user's names and names within the Unified Graphics System will be minimized.

The first argument in almost all subroutines is a character string, called 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; that is, 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; that is, the characters "0" and "1".

Thus, an example of an options list is:

```
FLAG1,FLAG2=3,FLAG3=2.15,FLAG4='ABC DEF',FLAG5=0101
```

Blanks may occur in the options list at its beginning or end, on either side of the commas separating items, and on either side of 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.

The Unified Graphics System contains a flexible scheme to report errors to the programmer. Any errors detected by the subroutines are classified into one of four severity levels. The error levels and their default actions are:

1. Minor Errors: The error indicators are set, but no error message is printed. The error indicators are values contained in a COMMON block. The program continues executing.
2. Errors: The error indicators are set, and an error message is printed. The program continues executing.
3. Severe Errors: An error message is printed, and the program terminates.
4. Terminal Errors: An error message is printed, and the program terminates with a memory dump and/or subroutine trace-back.

The error indicators, for non-terminal errors, may be checked by the program to determine what the error was, and possibly correct

the problem. The error message contains the name of the subroutine detecting the error, the severity level, the index of the error, and a description of the problem. The descriptions of each subroutine includes a list of all of the errors that the subroutine can detect. Both the index and the level number, in parenthesis, as well as a short description are given. Additional information on the handling of errors will be found in the section on Error Processing.

In the following descriptions of the subroutines, floating point or character string arguments are always described as such; if nothing is said about the data type of a parameter, it is fixed point. All arguments described as character strings must be character string literals or of type CHARACTER. Almost all arguments represent input to these subroutines; when an argument is an output variable, it will be explicitly described as such and will be underlined in the list of parameters following the calling sequence. In some cases, the underlined parameter may be used for both input and output.

SECTION 2.1: GRAPHIC SEGMENT GENERATION

The subroutines which are described in the following sub-sections may be used to pack picture description data into a graphic segment. A graphic segment is contained in an array of full word integers. An example of a statement allocating space for a graphic segment is:

```
      INTEGER*4     SEGMNT(500)
```

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

The options argument in the graphic segment generation subroutines usually specifies picture description parameters. The key words used for some of these items are:

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options. The Unified Graphics System supports five levels of intensity. VDIM is an abbreviation for "very dim" and VBRIGHT stands for "very bright". On some graphic devices, intensity level is simulated by line width.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options. The system supports the three additive primaries and their secondaries. The options items WHITE and BLACK are somewhat of a misnomer. WHITE really means the default color; for example, on a simple graphic device like a pen plotter, WHITE really means black. BLACK, on the other hand, really means the background color; on a pen plotter, this color would also probably result in a black display item.

The principal use of BLACK is for drawing on top of polygonal areas of color. On monochrome graphic devices, all colors, including both BLACK and WHITE, will be drawn in the default color.

BLINK and STEADY: Blink Options. Some graphic devices have a feature whereby parts of the picture may blink on and off.

When default values must be supplied by the Unified Graphics System, these defaults will usually be MEDIUM, WHITE, and STEADY. A subroutine is provided which allows the programmer to set the default values to something other than those given here. This system will utilize as many of these parameters as possible when the graphic segment 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 intensity level can sometimes be simulated by internal software in the Unified Graphics System. Thus, a user of the system may make free use of these picture description parameters; when the graphic segment is transmitted to the display device, as many parameters as possible will be utilized within the limitations of the actual display device.

SECTION 2.1.1: SUBROUTINE UGINIT

This subroutine may be used to clear and initialize a graphic segment. After this subroutine has been called, other subroutines may be called to add picture description data to the segment.

The calling sequence is:

```
CALL UGINIT(OPTIONS,SEGMENT,NSEGM)
```

The parameters in the calling sequence are:

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

- CLEAR The graphic segment is cleared and made ready to accept picture data. This is the normal use of this subroutine.
- RESET The graphic segment is not cleared but is made ready to accept more data. The only time this operation is normally necessary is after a graphic segment has been read from an external file and before new picture data is added to it.
- CONTINUE The graphic segment 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.

If none of the options items listed above is given, the default is CLEAR.

<u>SEGMENT</u> The graphic segment which is to be initialized.

NSEGM The length of the array SEGMENT in words.

The index and severity level of the errors detected by this subroutine are:

1(3): The length of the graphic segment is too small. The minimum size is 16, but it must be much larger than this to be useful.

SECTION 2.1.2: SUBROUTINE UGMARK

This subroutine may be used to add a single marker symbol to a graphic segment. A marker may consist of a single point or a more elaborate plotting symbol.

The calling sequence is:

```
CALL UGMARK(OPTIONS,XCOORD,YCOORD,SEGMENT)
```

The parameters in the calling sequence are:

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

MARK=<value> This item selects a special plotting symbol. The value may be any digit from 0 through 9. The meaning of the selected digit is:

- 0 Vertical Cross.
- 1 Diagonal Cross.
- 2 Diamond.
- 3 Square.
- 4 Fancy Diamond.
- 5 Fancy Square.
- 6 Fancy Vertical Cross.
- 7 Fancy Diagonal Cross.
- 8 Star Burst.
- 9 Octagon.

These symbols are shown in the section on the Extended Character Set. If the MARK item is not given, a single point is usually plotted.

SIZE=<value> The size of the marker symbol relative to the window in effect when the graphic segment is transmitted to the display device.

DSIZE=<value> The size of the marker symbol relative to the default drawing space. This item provides a means of specifying the size of the marker symbols that is independent of the current drawing space or window.

The SIZE and DSIZE values give the approximate vertical and horizontal size of the marker symbols. This size is measured relative to the horizontal axis. It is important to remember that the DSIZE value gives the size relative to the default square drawing space and not relative to the full physical drawing area. If MARK is given but neither SIZE nor DSIZE are given, then the default is usually DSIZE=0.015.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT:

Intensity Level Options.
WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.
BLINK and STEADY: Blink Options.
PICKID=<value> The pick identification of the marker symbol. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of the center of the marker.

YCOORD The floating point Y coordinate of the center of the marker.

SEGMENT The graphic segment which will have the marker added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.3: SUBROUTINE UGLINE

This subroutine may be used to add the end point of a single straight line to a graphic segment. The resulting 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. Curved lines are drawn by concatenating a series of short line segments. In addition to solid lines, this subroutine will also generate dashed, dotted, and dot-dashed line structure. For most graphic devices, line structure is produced by internal software in the Unified Graphics System. An exception is that line structure will not be produced on refresh display devices unless it can be done by the device itself. The reason is that generating dots and dashes instead of straight lines on such a device will quickly fill the refresh buffer. When line structure is generated by the Unified Graphics System, it is relatively independent of end point spacing along a curve. In a dashed line, for example, the line always begins with a dash and has dashes whose length is approximately one-third centimeter. A dashed line may, therefore, not always end with a dash. The dots in a dotted line are spaced about one-fourth of a centimeter apart, and a dot-dashed line also uses this spacing.

The calling sequence is:

```
CALL UGLINE(OPTIONS,XCOORD,YCOORD,BBIT,SEGMENT)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
SOLID, DASHED, DOTTED, and DOTDASH: Line Structure Options. The default value is usually SOLID.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.
WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.
BLINK and STEADY: Blink Options.
PICKID=<value> The pick identification of the line segment. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of an end point of a line.

YCOORD The floating point Y coordinate of an end point of a line.

BBIT The blanking bit. A zero indicates a blanked line, while a one indicates a drawn line.

SEGMENT The graphic segment which will have the end point of the line added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.4: SUBROUTINE UGPMRK

This subroutine may be used to add a "polymarker", that is, a number of marker symbols, to a graphic segment. A marker may consist of a single point or a more elaborate plotting symbol.

The calling sequence is:

```
CALL UGPMRK(OPTIONS,XARRAY,YARRAY,NCOORD,SEGMENT)
```

The parameters in the calling sequence are:

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

MARK=<value> This item selects a special plotting symbol. The value may be any digit from 0 through 9. The meaning of the selected digit is:

0 Vertical Cross.
1 Diagonal Cross.
2 Diamond.
3 Square.
4 Fancy Diamond.
5 Fancy Square.
6 Fancy Vertical Cross.
7 Fancy Diagonal Cross.
8 Star Burst.
9 Octagon.

These symbols are shown in the section on the Extended Character Set. If the MARK item is not given, a single point is usually plotted.

SIZE=<value> The size of the marker symbols relative to the window in effect when the graphic segment is transmitted to the display device.

DSIZE=<value> The size of the marker symbols relative to the default drawing space. This item provides a means of specifying the size of the marker symbols that is independent of the current drawing space or window.

The SIZE and DSIZE values give the approximate vertical and horizontal size of the marker symbols. This size is measured relative to the horizontal axis. It is important to remember that the DSIZE value gives the size relative to the default square drawing space and not relative to the full physical drawing area. If MARK is given but neither SIZE nor DSIZE are given, then the default is usually DSIZE=0.015.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the marker symbols. The default value for the pick identification is usually zero.

XARRAY A floating point array containing the X coordinates of the centers of the markers.

YARRAY A floating point array containing the Y coordinates of the centers of the markers.

NCOORD The number of coordinates in XARRAY and YARRAY.

SEGMENT The graphic segment which will have the markers added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.5: SUBROUTINE UGPLIN

This subroutine may be used to add a "polyline", that is, a number of end points of straight lines, to a graphic segment. The resulting vectors may either be blanked or drawn. The system always blanks to the first end point in the arrays of end point coordinates, but from there on, the blanking sequence is completely under the control of the programmer. In addition to solid lines, this subroutine will also generate dashed, dotted, and dot-dashed line structure.

The calling sequence is:

```
CALL UGPLIN(OPTIONS,XARRAY,YARRAY,NCOORD,BBITS,NBBITS,SEGMENT)
```

The parameters in the calling sequence are:

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

SOLID, DASHED, DOTTED, and DOTDASH: Line Structure Options. The default value is usually SOLID.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the line segments. The default value for the pick identification is usually zero.

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.

NCOORD The number of coordinates in XARRAY and YARRAY.

BBITS The blanking bits. This parameter is an integer array which may contain one bit per word or 32 bits per word. 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 result 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 unpacked bits are used, the first blanking bit is the low order bit of BBITS(1); when packed bits are used, the first blanking bit is the high order bit of BBITS(1).

SEGMENT The graphic segment which will have the end points of the lines added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.6: SUBROUTINE UGTEXT

This subroutine may be used to add text material to a graphic segment. The text will normally occupy a single line. The actual characters will be produced either by the hardware character generator on the device or by a programmed stroke generator. The only characters that are guaranteed to be produced correctly by this subroutine are those in the "basic"

character set. The basic character set consists of:

1. The upper case Roman alphabet.
2. The numerals.
3. The special characters: blank, plus sign, minus sign, asterisk, slash mark, equals sign, period, comma, left parenthesis, and right parenthesis.

These characters are exactly those common to the 8-bit EBCDIC, 7-bit ASCII, and 6-bit BCD character codes. If the hardware character generator on the graphic device produces additional characters that are representable on the host computer, then they will usually be correctly displayed. The programmed stroke generator will actually produce a few additional symbols beyond the basic character set: lower case Roman letters will be produced as upper case, and some additional special characters will be produced. If a character string is being drawn by the hardware character generator and this character generator cannot produce the requested character, then a non-blank character, usually an "at sign" (@), will be produced. If a program requests a character that the programmed stroke generator cannot produce, a special character consisting of seven horizontal lines is produced instead. The characters are positioned by giving the coordinates of the center of the first character, the center of the last character, or the center of the string. The characters are always equally spaced.

The calling sequence is:

```
CALL UGTEXT(OPTIONS,XCOORD,YCOORD,TEXT,SEGMENT)
```

The parameters in the calling sequence are:

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

SIZE=<value> The size of the characters relative to the window in effect when the graphic segment is transmitted to the display device.

DSIZE=<value> The size of the characters relative to the default drawing space. This item provides a means of specifying the size of the characters that is independent of the current drawing space or window.

The SIZE and DSIZE values give the spacing between the centers of consecutive characters and are also the approximate height of the characters. This size is measured relative to the horizontal axis. It is important to remember that the DSIZE value gives the size relative to the default square drawing space and not relative to the full physical drawing area. If neither SIZE nor DSIZE are given, then the default is usually DSIZE=0.015.

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 usually an angle of 0.0 degrees.

HARDGN, SOFTGN, and NORMGN: These items

describe the importance of using the hardware character generator when drawing characters. HARDGN means that the hardware character generator must be used even if SIZE or DSIZE cannot be matched very well. SOFTGN means that the programmed stroke generator must be used. NORMGN means that the hardware character generator will be used only if it matches SIZE or DSIZE reasonably well. The default is usually NORMGN.

LEFT, RIGHT, and CENTER: These items specify the part of the character string that is being positioned by the X and Y coordinates. The default is usually LEFT; that is, the X and Y coordinates specify the position of the center of the first character in the string.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the characters. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of the part of the character string being positioned. This may be the center of the first character, the center of the last character, or the center of the string.

YCOORD The floating point Y coordinate of the part of the character string being positioned.

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

<u>SEGMENT</u> The graphic segment which will have the text string added to it.

The index and severity level of the errors detected by this subroutine are:

1(2): The string length is invalid. The length must be at least 1 and at most 1024.

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

A problem 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 center while some actual devices use other parts of the character to locate its position. The result is that the positioning point in the Unified Graphics System may be within the drawing area, but the hardware-positioning point is off-screen. Because of these differences in 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 usually keep the character's hardware location point completely within the plotting area and no anomalies should occur.

SECTION 2.1.7: SUBROUTINE UGXTXT

This subroutine may be used to add extended text material to a graphic segment. The text will normally occupy a single line. The actual characters will be produced by a programmed stroke generator of great versatility. The stroke generator can generate upper and lower case Roman, Greek, and Cyrillic letters; a wide variety of special symbols; and has a versatile subscripting and superscripting facility. A character string is specified by giving two strings of equal length; a primary and a secondary string. The primary string gives an approximation to the actual character, while the secondary string gives a modifier character. For example, suppose the third character in a string is to be an upper case "A". In this case, the third character of the primary string should be an "A", while the third character of the secondary string should be a blank. If a lower case Roman "A" were desired, the corresponding characters are "A" and "L"; if a lower case Greek alpha were desired, the characters are "A" and "G", etc. The characters may be drawn in either of two fonts. The characters in the "simplex" font are made up of simple straight lines, while the "duplex" font contains characters with doubled strokes and serifs. A complete description of the available characters is given in the section on the Extended Character Set. If a program specifies an invalid primary-secondary character pair, then a special character consisting of seven horizontal lines is produced instead.

The calling sequence is:

```
CALL UGXTXT(OPTIONS,XCOORD,YCOORD,PRITXT,SECTXT,SEGMENT)
```

The parameters in the calling sequence are:

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

SIZE=<value> The size of the characters relative to the window in effect when the graphic segment is transmitted to the display device.

DSIZE=<value> The size of the characters relative to the default drawing space. This item provides a means of specifying the size of the characters that is independent of the current drawing space or window.

The SIZE and DSIZE values give the approximate spacing between the centers of consecutive characters and are also the approximate height of the characters. This size is measured relative to the horizontal axis. It

is important to remember that the DSIZE value gives the size relative to the default square drawing space and not relative to the full physical drawing area. Also, remember that actions like entering subscript or superscript mode change the actual size and spacing of the characters. If neither SIZE nor DSIZE are given, then the default is usually DSIZE=0.015.

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 usually an angle of 0.0 degrees.

LEFT, RIGHT, and CENTER: These items specify the part of the character string that is being positioned by the X and Y coordinates. The default is usually LEFT; that is, the X and Y coordinates specify the position of the center of the first character in the string.

FIXSIZE and NOFXSIZ: These items indicate whether or not the characters are to be proportionally spaced or are to have constant spacing as given by the SIZE or DSIZE items. The default is usually NOFXSIZ which results in proportionally spaced characters. The purpose of FIXSIZE is to provide a way to make small tables. It is difficult to use FIXSIZE with subscripts and superscripts, size changes, and horizontal or vertical movements.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the characters. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of the part of the character string being positioned. This may be the center of the first character, the center of the last character, or the center of the string.

YCOORD The floating point Y coordinate of the part of the character string being positioned.

PRITXT The string containing the primary characters to be added to the graphic segment.

SECTXT The string containing the secondary characters to be added to the graphic segment.

SEGMENT The graphic segment which will have the text strings added to it.

The index and severity level of the errors detected by this subroutine are:

1(2): The string lengths are invalid. The lengths must be at least 1 and at most 1024.

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.8: SUBROUTINE UGPFIL

This subroutine may be used to add a "polygon-fill" to a graphic segment. A polygon-fill is a polygon which will be filled with color on a graphic device. The polygon is defined by giving a sequence of vertices. If the first and last vertex are not identical, the Unified Graphics System will supply a closing vertex. The polygons supplied by this subroutine should be simple polygons; edges that cross each other are not valid. In contrast with most of the other primitives, this primitive is quite device-dependent in that it only works well on color graphic devices which support this function in hardware. If a device does not support this as a hardware primitive, the Unified Graphics System simulates it by outlining the polygon. In many cases this simulation is inadequate because the hardware polygon-fill erases everything within the polygon and the simulation does not. A more adequate simulation is possible but only at a severe increase in complexity for the simplest graphic devices. On some monochrome graphic devices that support polygon-fill, the Unified Graphics System will clear the interior of the polygon and then draw the border of the polygon.

The calling sequence is:

```
CALL UGPFIL(OPTIONS,XARRAY,YARRAY,NCOORD,SEGMENT)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.
- WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.
- BLINK and STEADY: Blink Options.
- PICKID=<value> The pick identification of the polygon-fill. The default value for the pick identification is usually zero.

XARRAY A floating point array containing the X coordinates of the vertices of the polygon.

YARRAY A floating point array containing the Y coordinates of the vertices of the polygon.

NCOORD The number of coordinates in XARRAY and YARRAY.

SEGMENT The graphic segment which will have the polygon-fill added to it.

The index and severity level of the errors detected by this subroutine are:

1(2): The polygon is invalid. The number of vertices must be at least 4 and at most 32 (including the closing

vertex).

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.9: SUBROUTINE UG3MRK

This subroutine may be used to add a single three-dimensional marker symbol to a graphic segment. A three-dimensional marker may only consist of a single point and not any of the more elaborate plotting symbol allowed for two-dimensional markers.

The calling sequence is:

```
CALL UG3MRK(OPTIONS,XCOORD,YCOORD,ZCOORD,SEGMENT)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.
- WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.
- BLINK and STEADY: Blink Options.
- PICKID=<value> The pick identification of the marker symbol. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of the marker.

YCOORD The floating point Y coordinate of the marker.

ZCOORD The floating point Z coordinate of the marker.

<u>SEGMENT</u> The graphic segment which will have the three-dimensional marker added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.10: SUBROUTINE UG3LIN

This subroutine may be used to add the end point of a single three-dimensional straight line to a graphic segment. The resulting 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. Curved lines are drawn by concatenating a series of short line segments. Three-dimensional lines may only be solid lines and not any of the other line structures allowed for two-dimensional lines.

The calling sequence is:

```
CALL UG3LIN(OPTIONS,XCOORD,YCOORD,ZCOORD,BBIT,SEGMENT)
```

The parameters in the calling sequence are:

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

- VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.
- WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.
- BLINK and STEADY: Blink Options.
- PICKID=<value> The pick identification of the line segment. The default value for the pick identification is usually zero.

XCOORD The floating point X coordinate of an end point of a line.

YCOORD The floating point Y coordinate of an end point of a line.

ZCOORD The floating point Z coordinate of an end point of a line.

BBIT The blanking bit. A zero indicates a blanked line while a one indicates a drawn line.

<u>SEGMENT</u> The graphic segment which will have the end point of the three-dimensional line added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.11: SUBROUTINE UG3PMK

This subroutine may be used to add a three-dimensional polymarker to a graphic segment. A three-dimensional marker may only consist of a single point and not any of the more elaborate plotting symbol allowed for two-dimensional markers.

The calling sequence is:

```
CALL UG3PMK(OPTIONS,XARRAY,YARRAY,ZARRAY,NCOORD,SEGMENT)
```

The parameters in the calling sequence are:

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

- NOCONST, XCONST, YCONST, and ZCONST: Constant coordinate flag. This item can be convenient to use when the graphic data lies in a plane parallel to one of the coordinate planes. If XCONST is given, the X coordinate for all of the points is obtained from XARRAY(1). YCONST and ZCONST operate similarly. The default value is usually

NOCONST which means that all NCOORD entries in the arrays are used.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the marker symbols. The default value for the pick identification is usually zero.

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

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

ZARRAY A floating point array containing the Z coordinates of the markers.

NCOORD The number of coordinates in XARRAY, YARRAY, and ZARRAY.

SEGMENT The graphic segment which will have the three-dimensional markers added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.12: SUBROUTINE UG3PLN

This subroutine may be used to add a three-dimensional polyline to a graphic segment. The resulting vectors may either be blanked or drawn. The system always blanks to the first end point in the arrays of end point coordinates, but from there on, the blanking sequence is completely under the control of the programmer. Three-dimensional lines may only be solid lines and not any of the other line structures allowed for two-dimensional lines.

The calling sequence is:

```
CALL UG3PLN(OPTIONS,XARRAY,YARRAY,ZARRAY,NCOORD,
            BBITS,NBBITS,SEGMENT)
```

The parameters in the calling sequence are:

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

NOCONST, XCONST, YCONST, and ZCONST: Constant coordinate flag. This item can be convenient to use when the graphic data lies in a plane parallel to one of the coordinate planes. If XCONST is given, the X coordinate for all of the points is obtained from XARRAY(1). YCONST and ZCONST operate

similarly. The default value is usually NOCONST which means that all NCOORD entries in the arrays are used.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the line segments. The default value for the pick identification is usually zero.

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.

ZARRAY A floating point array containing the Z coordinates of the end points of the line segments.

NCOORD The number of coordinates in XARRAY, YARRAY, and ZARRAY.

BBITS The blanking bits. This parameter is an integer array which may contain one bit per word or 32 bits per word. 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 an 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 unpacked bits are used, the first blanking bit is the low order bit of BBITS(1); when packed bits are used, the first blanking bit is the high order bit of BBITS(1).

SEGMENT The graphic segment which will have the end points of the three-dimensional lines added to it.

The index and severity level of the errors detected by this subroutine are:

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.13: SUBROUTINE UG3TXT

This subroutine may be used to add three-dimensional text material to a graphic segment. The text will normally occupy a single line. On a three-dimensional graphic device, the text will always be produced by the hardware character generator. On a two-dimensional device it may be produced either by the hardware character generator or by a programmed stroke generator.

The character strings drawn with this subroutine will always be displayed as if they were in a plane perpendicular to the viewing direction. That is, the characters themselves do not rotate, only the position of the characters rotates. The only characters that are guaranteed to be produced correctly by this subroutine are those in the basic character set. If the hardware character generator on the graphic device produces additional characters that are representable on the host computer, then they will usually be correctly displayed. If a character string is being drawn by the hardware character generator and this character generator cannot produce the requested character, then a non-blank character, usually an "at sign" (@), will be produced. If a program requests a character that the programmed stroke generator cannot produce, a special character consisting of seven horizontal lines is produced instead. The characters are positioned by giving the three-dimensional coordinates of the center of the first character, the center of the last character, or the center of the string. The characters are always equally spaced.

The calling sequence is:

```
CALL UG3TXT(OPTIONS,XCOORD,YCOORD,ZCOORD,TEXT,SEGMENT)
```

The parameters in the calling sequence are:

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

DSIZE=<value> The size of the characters relative to the default drawing space.

The DSIZE value gives the spacing between the centers of consecutive characters and is also the approximate height of the characters. This size is measured relative to the horizontal axis. It is important to remember that the DSIZE value gives the size relative to the default square drawing space and not relative to the full physical drawing area. If DSIZE is not given, then the default is usually DSIZE=0.015.

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 usually an angle of 0.0 degrees.

LEFT, RIGHT, and CENTER: These items specify the part of the character string that is being positioned by the X, Y, and Z coordinates. The default is usually LEFT, that is, the X, Y, and Z coordinates specify the position of the center of the first character in the string.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the characters. The default value for the pick

identification is usually zero.

XCOORD The floating point X coordinate of the part of the character string being positioned. This may be the center of the first character, the center of the last character, or the center of the string.

YCOORD The floating point Y coordinate of the part of the character string being positioned.

ZCOORD The floating point Z coordinate of the part of the character string being positioned.

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

SEGMENT The graphic segment which will have the three-dimensional text string added to it.

The index and severity level of the errors detected by this subroutine are:

1(2): The string length is invalid. The length must be at least 1 and at most 1024.

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.14: SUBROUTINE UGDDAT

This subroutine may be used to add device-dependent data to a graphic segment. This data is transmitted to the graphic device in unmodified form. Only a few graphic devices will accept this data; most devices will simply ignore it. For those graphic devices which will accept this data, it is an unenforced suggestion that the first eight characters consist of the device identification, as supplied to UGOPEN, and a colon, followed by the actual device-dependent data. This subroutine has a very specialized purpose and its successful utilization cannot be accomplished with the information in this document.

The calling sequence is:

```
CALL UGDDAT(OPTIONS,XCOORD,YCOORD,DATA,SEGMENT)
```

The parameters in the calling sequence are:

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

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the characters. The default value for the pick identification is usually zero.

XCOORD A floating point X coordinate associated with the device-dependent data.

YCOORD A floating point Y coordinate associated with the

device-dependent data.

DATA A character string containing the device-dependent data to be added to the graphic segment.

SEGMENT The graphic segment which will have the device-dependent data added to it.

The index and severity level of the errors detected by this subroutine are:

1(2): The string length is invalid. The length must be at least 1 and at most 1024.

11(2): Not enough room is available in the graphic segment to contain the new data. In this case, nothing has been added to the graphic segment, and it remains unchanged.

SECTION 2.1.15: SUBROUTINE UGDEFL

This subroutine may be used to change the default values for many of the options items in the previously described subroutines. When one of these earlier subroutines is called, its OPTIONS parameter may override any of the values set by this subroutine. The advantage of using this subroutine is that it can eliminate the repeated scanning of long options lists. It is more efficient, for example, to use this subroutine once followed by repeated calls to UGLINE with a null options list than to include a long options list with each call to UGLINE. The disadvantage involved in using this subroutine is that programs can become more difficult to read and understand.

The calling sequence is:

```
CALL UGDEFL(OPTIONS,IARRAY,XARRAY)
```

The parameter in the calling sequence is:

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

RESET This item causes the options items to be reset to their default values.

GET This item causes the options items to be saved in the IARRAY and XARRAY arrays. IARRAY(1) is the intensity level value; 1 for VDIM, 2 for DIM, 3 for MEDIUM, 4 for BRIGHT, and 5 for VBRIGHT. IARRAY(2) is the color value; 1 for WHITE, 2 for RED, 3 for GREEN, 4 for BLUE, 5 for YELLOW, 6 for MAGENTA, 7 for CYAN, and 8 for BLACK. IARRAY(3) is the blink value; 1 for STEADY and 2 for BLINK. IARRAY(4) is the MARK value; -1 means a simple point. IARRAY(5) is the line structure value; 1 for SOLID, 2 for DASHED, 3 for DOTTED, and 4 for DOTDASH. IARRAY(6) is the justification value; 1 for LEFT, 2 for RIGHT, and 3 for CENTER. IARRAY(7) is the character priority; 1 for

NORMGN, 2 for HARDGN, and 3 for SOFTGN. IARRAY(8) is the fixed size value; 1 for NOFXSIZ and 2 for FIXSIZE. IARRAY(9) is the constant coordinate flag; 0 for NOCONST, 1 for XCONST, 2 for YCONST, and 3 for ZCONST. IARRAY(10) is the pick identification value. XARRAY(1) is the SIZE value; 0.0 means DSIZE is the default. XARRAY(2) is the DSIZE value; 0.0 means SIZE is the default. Finally, XARRAY(3) is the ANGLE value.

PUT This item causes the options items to be set from the IARRAY and XARRAY arrays.

At most, one of the above items should be given. If one of the above items is used in conjunction with some of the following items, the RESET-GET-PUT action will occur first, even if that item is not first in the option list.

VDIM, DIM, MEDIUM, BRIGHT, and VBRIGHT: Intensity Level Options.

WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, and BLACK: Color Options.

BLINK and STEADY: Blink Options.

PICKID=<value> The pick identification of the display item.

MARK=<value> This item selects a special plotting symbol. The value may be any digit from -1 through 9. The meaning of the selected digit is:

- -1 A Single Point.
- 0 Vertical Cross.
- 1 Diagonal Cross.
- 2 Diamond.
- 3 Square.
- 4 Fancy Diamond.
- 5 Fancy Square.
- 6 Fancy Vertical Cross.
- 7 Fancy Diagonal Cross.
- 8 Star Burst.
- 9 Octagon.

SOLID, DASHED, DOTTED, and DOTDASH: Line Structure Options.

SIZE=<value> The size of the marker symbols or characters relative to the window in effect when the graphic segment is transmitted to the display device.

DSIZE=<value> The size of the marker symbols or characters relative to the default drawing space. This item provides a means of specifying the size of the marker symbols or characters that is independent of the current drawing space or window.

ANGLE=<value> The angle that the characters make with the horizontal. The angle is measured, in degrees, in the counter-clockwise direction.

HARDGN, SOFTGN, and NORMGN: These items describe how important it is that the hardware character generator be used when the characters are drawn. HARDGN means that the hardware character generator must be used even if SIZE or DSIZE cannot be matched very well. SOFTGN means that the programmed stroke generator must be used. NORMGN means that the hardware character generator will be used only if it matches SIZE or DSIZE reasonably well.

LEFT, RIGHT, and CENTER: These items specify the part of the character string that is being positioned by the X and Y coordinates.

FIXSIZE and NOFXSIZ: These items indicate whether or not the characters are to be proportionally spaced or are to have constant spacing as given by the SIZE or DSIZE items.

NOCONST, XCONST, YCONST, and ZCONST: Constant coordinate flag for three-dimensional polymarkers and polylines.

IARRAY A fixed point array for the GET and PUT options.

XARRAY A floating point array for the GET and PUT options.

No error messages are produced by this subroutine.

There are two problems associated with the PUT option in this subroutine. First, an application program can become very difficult to read and understand because the properties of a graphic item are not set at the time that the graphic item is generated. The second problem is more fundamental; if the Unified Graphics System is ever extended and more picture description options are made available, existing code may no longer work because new items are not correctly initialized in the arrays. This latter problem can be overcome by careful use of the GET option. First, the programmer should define IARRAY and XARRAY to be larger than necessary and use the statement:

```
CALL UGDEFL('GET',IARRAY,XARRAY)
```

to initialize IARRAY and XARRAY. Then the options items may be changed and the changes communicated to the Unified Graphics System by calling UGDEFL with the PUT option.

SECTION 2.2: GRAPHIC DEVICE CONTROL

The subroutines in this section perform basic operations on the graphic device. Subroutine UGOPEN is used to initialize a graphic device and prepare it for use while UGCLOS is used to terminate the use of a graphic device.

UGSLCT is used to select an open graphic device and make it the "active" device. Most versions of the Unified Graphics System permit a number of graphic devices to be open at once. However, the subroutines in the Unified Graphics System that communicate with the graphic device only talk to the active graphic device. At most one graphic device may be active at one time and this device may be selected dynamically with subroutine UGSLCT.

Other subroutines, UGINFO and UGMCTL, are provided to obtain information about a graphic device or perform certain control operations.

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 sequence is:

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

The parameters in the calling sequence are:

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

- SDDXCOW The Console on Wheels (COW) of the SLC Project. These units are always used in the slave-display mode.
- SEQGIGI Display Files for the DEC GIGI Color Graphics Terminal are saved in a sequential file.
- SDDGIGI The DEC GIGI Color Graphics Terminal as a slave-display device.
- DECGIGI The DEC GIGI Color Graphics Terminal as an interactive display device.
- SDDVST2 The DEC VAXSTATION II/GPX or VAXSTATION 2000 as a slave-display graphic device.
- DECVST2 The DEC VAXSTATION II/GPX or VAXSTATION 2000 as an interactive graphic device.
- SDDGRIN The GRINNELL GMR-27 Display System. These units are always used in the slave-display mode.
- IBM3179 The IBM 3179 G Color Graphics Display Station. These units are always used in the interactive mode.
- IBM5080 The IBM 5080 Graphics System. These units are always used in the interactive mode.

IMGN300 The Model 8/300 Laser Printer/Plotter made by the IMAGEN Corporation. This is a non-interactive graphic device.

IMGNIBM This options item produces files for the IMAGEN Model 8/300 Laser Printer/Plotter on a VAX that must later be sent to the IBM computer for plotting.

SDDMETH The Omega 300 Display Controller made by the Metheus Corporation. This is a slave-display graphic device.

POSTSCR Display Files are created using the PostScript Language. Either monochrome or color pictures may be produced. These devices are always used in the non-interactive mode.

PRNTRNX The PRINTRONIX (Model MVP) impact printer/plotter. This is a non-interactive graphic device.

SDDSKRS The Seiko GR-1105 Color Graphic Terminal as a slave-display device in the raster-scan mode.

SDDSKRF The Seiko GR-1105 Color Graphic Terminal as a slave-display device in the refresh mode.

SEIKORS The Seiko GR-1105 Color Graphic Terminal as an interactive device in the raster-scan mode.

SEIKORF The Seiko GR-1105 Color Graphic Terminal as an interactive device in the refresh mode.

SDDSXSS The SLAC Experimental Slave Scope. These units are always used in the slave-display mode.

TALARIS The Laser Printer/Plotters made by Talaris Systems. This is a non-interactive graphic device.

SEQ4010 Display Files for the TEKTRONIX 4010 series terminals are saved in a sequential file.

SDD4010 The TEKTRONIX 4010 series terminal as a slave-display device.

TEK4010 The TEKTRONIX 4010 series terminal as an interactive display device.

SEQTKEM Display Files for TEKTRONIX 4010/4014 Emulators are saved in a sequential file.

SDDTKEM TEKTRONIX 4010/4014 Emulators as a slave-display device.

TEKEMUL TEKTRONIX 4010/4014 Emulators as an interactive display device.

SEQ4027 Display Files for the TEKTRONIX 4027 Color Graphics Terminals are saved in a sequential file.

SDD4027 The TEKTRONIX 4027 Color Graphics Terminal as a slave-display device.

TEK4027 The TEKTRONIX 4027 Color Graphics

Terminal as an interactive display device.

SEQ4105 Display Files for the TEKTRONIX 4105 Computer Display Terminal are saved in a sequential file.

SDD4105 The TEKTRONIX 4105 Computer Display Terminal as a slave-display device.

TEK4105 The TEKTRONIX 4105 Computer Display Terminal as an interactive display device.

SDD4207 The TEKTRONIX 4207 Computer Display Terminal as a slave-display device.

TEK4207 The TEKTRONIX 4207 Computer Display Terminal as an interactive display device.

TEK4510 The TEKTRONIX 4510 Color Graphics Rasterizer.

VEP12FF The VERSATEC Electrostatic Plotter with fan-fold paper. This is a non-interactive graphic device.

VEP12CR The VERSATEC Electrostatic Plotter with continuous-roll paper. This is a non-interactive graphic device.

SDDXWDO The X-Windows protocol in a slave-display mode.

XWINDOW The X-Windows protocol in an interactive mode.

PDEVUGS A graphic pseudo-device which causes the graphic data to be made available as unprocessed graphic segments.

PDEVLIN A graphic pseudo-device which causes the graphic data to be made available after it has been transformed for a line drawing graphic device.

PDEVSVR A graphic pseudo-device which causes the graphic data to be made available after it has been transformed for a raster graphic device. A single value, one bit, is made available for each raster point on the graphic device.

GENWKST Device-dependent code for a generic workstation. This may be used to drive any programmable workstation from the host computer as long as a companion program has been prepared for the workstation.

Exactly one of the above items should be given in the options list. There are many additional items which may appear in the options list but most of these are device-dependent. They are described in the later sections devoted to the specific graphic devices.

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 nonzero value. However, if multiple graphic devices are being used, each must have a distinct identification.

The index and severity level of the errors detected by this subroutine are:

1(3): A valid graphic device was not specified. Check your options parameter.

2(3): Invalid identification. The identification parameter must have a nonzero value.

3(3): Duplicate identification. If you are trying to open more than one graphic device, then each device must have a unique identification.

4(3): Too many graphic devices. For some versions of the Unified Graphics System, only one graphic device may be open at once. For other versions, a larger number (usually 32) may be open at once.

5(3): The device-dependent code for the selected device is not available. The executable load module will have to be recreated.

6(3): The graphic device cannot be opened. Check your specification statements for the output file.

7(2): You have opened more than one fully interactive graphic device. This may not work satisfactorily.

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 reopened. 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 sequence is:

```
CALL UGCLOS(OPTIONS)
```

The parameter in the calling sequence 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.

NOCLEAR This item indicates that the display device is not to be cleared on termination. This item only applies to slave-display and interactive devices. The default is to clear the display device on termination.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

SECTION 2.2.3: SUBROUTINE UGSLCT

This subroutine may be used to select any open graphic device and make it the active device. All of the subroutines in the Unified Graphics System that communicate with a graphic device, such as UGPICT or UGWRIT, only talk to the active graphic device. Only one graphic device may be active at once. The programmer will not have to use this subroutine unless more than one graphic device is being used by the program.

The calling sequence is:

```
CALL UGSLCT(OPTIONS,IDENT)
```

The parameters in the calling sequence 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 index and severity level of the errors detected by this subroutine are:

- 1(3): The given identification is invalid; no such graphic device is open at present.

SECTION 2.2.4: SUBROUTINE UGINFO

This subroutine may be used to inquire about the status of the Unified Graphics System, and to determine some of the properties and capabilities of the active graphic device. With each call to this subroutine, the programmer may ask a single question or a small number of questions if the answers do not conflict. It is the responsibility of the user to assure that the output parameters have enough space to hold the answers.

The calling sequence is:

```
CALL UGINFO(OPTIONS,STRING,IARRAY,XARRAY)
```

The parameters in the calling sequence are:

- OPTIONS — A character string which may contain any of the following items:
  - OPENDEV — This item causes IARRAY(1) to be set to the number of open graphic devices. The entries starting with IARRAY(2) are set to the identifications of the open devices.
  - ACTDEV — IARRAY(1) is set to the identification of the active graphic device. A zero value means no device is active.

  The options described above do not require that a graphic device be active. To answer the following questions, a graphic device must be active.
  - DEVTYPE — This item causes STRING to be set to the device type as listed under UGOPEN. The seven character name is terminated by a

blank so that eight characters are returned.

ILEVEL IARRAY(1) is set to the interaction level of the active device: 1 means non-interactive, 2 means slave-display, and 3 means interactive.

DMEDIUM IARRAY(2) is set to the drawing medium properties: 1 means a non-erasable medium, 2 means a raster-scan device, and 3 means a refresh display device.

DIMENSION IARRAY(3) is set to the dimension of the graphic device: 2 means it is a two-dimensional device, and 3 means it is a three-dimensional device.

CONTROLS IARRAY(1) through IARRAY(8) are set to indicate the interactive controls (real or simulated) that are available on the active device. A zero means the control is not available while a positive number means it is available. IARRAY(1) is the keyboard indicator. IARRAY(2) is the pick indicator. IARRAY(3) is the number of buttons. IARRAY(4) is the stroke indicator. IARRAY(5) is the locator indicator. IARRAY(6) is the number of valuators. Finally, IARRAY(7) and IARRAY(8) are set to zero.

ECONTROLS IARRAY(1) through IARRAY(8) are set to indicate the interactive controls that have been enabled. A zero means the control is disabled.

EXTENSIONS IARRAY(1) through IARRAY(4) are set to indicate any indefinite extensions that are possible in the drawing medium. A zero means no extension is possible while a one means the extension is possible. IARRAY(1) is for the low X direction, IARRAY(2) is for the low Y direction, IARRAY(3) is for high X, and IARRAY(4) is the high Y value. A drum plotter, for example, would return a value of one in IARRAY(3) while most terminals return four zeros.

DSPCSIZE The number of centimeters per unit in the drawing space will be provided. XARRAY(1) is the X direction value, and XARRAY(2) is the Y direction value.

WDOWSIZE The number of centimeters per unit in the current window will be provided. XARRAY(1) is the X direction value and XARRAY(2) is the Y direction value.

STRING A character string to hold the output values.
IARRAY A fixed point array to hold the output values.
XARRAY A floating point array to hold the output values.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

SECTION 2.2.5: SUBROUTINE UGMCTL

This subroutine may be used to perform a number of miscellaneous control operations. Some control functions apply to the Unified Graphics System itself while others apply to the active graphic device. Multiple operations may be performed in one call to this subroutine as long as the operations are not contradictory.

The calling sequence is:

```
CALL UGMCTL(OPTIONS,STRING,IARRAY,XARRAY)
```

The parameters in the calling sequence are:

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

ERRUNIT=<value> This item specifies the FORTRAN unit number to which the error messages will be written. The default value is 6.

ERRMAX=<value> A count is kept for each error message, and the message is suppressed when this count reaches the value specified by this item. The default value is 8.

The options items described above do not require that a graphic device be active. The following items require that a device be active.

BEEP Sound the audible alarm on the graphic device. This item applies only to slave-display and interactive devices.

STRING A character string which may be used for both input and output.

IARRAY A fixed point array which may be used for both input and output.

XARRAY A floating point array which may be used for both input and output.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active graphic device.

SECTION 2.3: PICTURE CONTROL

This section describes those subroutines that are directly related to controlling the picture. Subroutine UGWRIT will transmit a graphic segment to a graphic device, and subroutine UGPICT gives the programmer control over the picture and parts of the picture.

On refresh display devices, the programmer may assign a numeric identification to a graphic segment when it is transmitted. This identification may then be used to manipulate the segment at a later time. For instance, a segment may have its pick accessibility status changed, or an individual segment 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 segments than from a large number of small segments. This is especially true on refresh display devices. When individual segments are deleted from a picture, it is usually 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 refresh displays also allow an individual segment to be temporarily turned off. Such a segment is said to be in the "omit" state (as opposed to the "include" state).

SECTION 2.3.1: SUBROUTINE UGWRIT

This subroutine may be used to transmit a previously constructed graphic segment to a display device. When a graphic segment is transmitted to a graphic device, it is first translated to device-dependent orders, and then added to the display file of the graphic device. After a graphic segment has been transmitted to a display device, the programmer may reuse the array containing the graphic segment. It is when UGWRIT is called that the Unified Graphics System makes use of the windowing parameters and any of the character stroke generators.

The calling sequence is:

```
CALL UGWRIT(OPTIONS,IDENT,SEGMENT)
```

The parameters in the calling sequence are:

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

- PICK Make the segment accessible to a pick device. This item only applies to refresh display devices. The default is to put the graphic segment into the NOPICK mode.
- OMIT Leave the segment in the omit state. This item only applies to refresh display devices. The default is to put the graphic segment into the INCLUDE state.
- ERASE Causes a segment that has previously been transmitted to be erased by tracing over the image. This item only applies to raster-scan display devices. It is the programmers responsibility to either save the graphic segment or regenerate it exactly. The

programmer must also assure that the same windowing parameters are present.

IDENT A numeric value which may be used to identify the segment. A zero value means that the segment is unidentified. On refresh display devices, transmitting two segments with the same nonzero identification will cause the second to replace the first.

SEGMENT The graphic segment which is to be transmitted to the graphic device.

The index and severity level of the errors detected by this subroutine are:

1(2): The display file is not large enough to hold all of the graphic segments. You will have to simplify the picture.

2(2): The graphic segment is too large for this graphic device. Try using smaller graphic segments.

3(4): The graphic segment contains invalid data.

4(2): An internal array has overflown while trying to scissor a polygon-fill area to the current window. Use simpler polygons.

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active graphic device.

14(3): The extended character module is not available. The executable load module will have to be recreated.

This subroutine sometimes generates underflows. This is usually caused by a user programming error; a subroutine such as UGLINE 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 graphic device. On refresh display devices, individual segments may be manipulated or deleted. Multiple operations may be performed in one call to this subroutine as long as the operations are not contradictory.

The calling sequence is:

```
CALL UGPICT(OPTIONS,IDENT)
```

The parameters in the calling sequence are:

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

CLEAR If IDENT is zero, the entire display is cleared, otherwise the specific graphic segment with an identification of IDENT is deleted. Deleting specific graphic segments only applies to refresh display devices.

WINDOW If this item is used with CLEAR, it means that the current window and view port

is to be cleared. Other parts of the picture remain unchanged. This operation only applies to some raster-scan display devices.

PICK and NOPICK: Change the pick accessibility of a segment. This item only applies to refresh display devices.

OMIT and INCLUDE: Change omit-include status of a graphic segment. This item only applies to refresh display devices.

ON and OFF: Turn the display device 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. This name is used in addition to the name constructed from the PICTID and PICTSQ options given in UGOPEN.

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

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active graphic device.

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

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

performs this operation. On a device like a drum plotter, this call causes the pen to move over to a clean piece of paper; on a refresh display device, the screen is cleared by reinitializing the display buffer. Any window or shields that are in effect are deleted. The drawing space limits that were established by the most recent call to UGDSPC are preserved and a default window and view port relative to those values are established. The three-dimensional parameters supplied by subroutine UG3TRN are also replaced by the default values relative to the last call to subroutine UG3WRD. 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. However, calling UGPICT before the first picture in this manner will not generate a blank picture.

Clearing a window causes all existing shields to be deleted but does not affect the window itself.

SECTION 2.4: GRAPHIC DEVICE INPUT CONTROL

This section describes subroutines which allow the programmer to obtain information from the interactive control units on an interactive graphic device. It is by means of these subroutines that the programmer synchronizes the execution of the program with the actions of the operator of the interactive device.

The Unified Graphics System supports six different types of interactive control units. These control unit types are:

1. Keyboards: A keyboard may be used to enter character strings. The operator of the interactive graphic device signals the end of a character string by hitting a special key, usually the "carriage return" or "enter" key. Almost all interactive graphic devices will have a keyboard.
2. Pick Control Units: A pick device may be used to select a displayed item. The Unified Graphics System returns the identification of the graphic segment that was selected and the pick identification of the graphic item within the graphic segment. An example of a pick device is a light pen. This type of control unit is only found on refresh display devices or on devices that simulate refresh display devices.
3. Button Control Units: A button device consists of a group of buttons which may be pushed. The Unified Graphics System returns the index of the pushed button.
4. Stroke Control Units: A stroke device may be used to generate a sequence of positions. The Unified Graphics System returns arrays of X and Y coordinates of the positions. Examples of a stroke device are a mouse, some joysticks, and some tracking balls.
5. Locator Control Units: A locator is a device that produces an X and Y coordinate on the display device. Examples of locator devices are touch panels, a mouse, some joysticks, and some tracking balls.
6. Valuator Control Units: A valuator returns a floating point value between 0.0 and 1.0. Valuators usually come in groups of a number of valuators. An example of a valuator is a control dial.

The Unified Graphics System will support all interactive control units that are available on a graphic device. In addition, the Unified Graphics System may sometimes try to simulate control units but this is usually not very successful and the programmer should not rely heavily on these simulations.

The interactive control units that have been described above are of two types. These types are:

1. Event Causing Control Units: These devices cause an "event" to occur when the graphic device operator performs some action. Records of the events are put on a queue and the programmer may obtain this information. When a programmer needs an event from the device operator, a wait of indefinite duration or a timed wait may be performed. The keyboard, pick, button, and

stroke control units are event causing devices.

2. Sampled Control Units: These devices do not cause events but may be read at any time. When a sampled control unit is read, the system will wait for an action by the operator to signal that the data is available. The locator and valuator control units are sampled devices.

SECTION 2.4.1: SUBROUTINE UGENAB

This subroutine may be used to enable an interactive control unit on a graphic device. A control unit must be enabled before any use can be made of it.

The calling sequence is:

```
CALL UGENAB(OPTIONS)
```

The parameter in the calling sequence is:

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

- KEYBOARD Indicates that the keyboard is to be enabled.
- PICK Indicates that the pick control unit is to be enabled.
- BUTTON Indicates that the button control unit is to be enabled.
- STROKE Indicates that the stroke control unit is to be enabled.
- LOCATOR Indicates that the locator control unit is to be enabled.
- VALUATOR Indicates that the valuator control unit is to be enabled.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active graphic device.

SECTION 2.4.2: SUBROUTINE UGDSAB

This subroutine may be used to disable an interactive control unit on a graphic device. Once a control unit is disabled, no use may be made of it until it is re-enabled.

The calling sequence is:

```
CALL UGDSAB(OPTIONS)
```

The parameter in the calling sequence is:

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

KEYBOARD Indicates that the keyboard is to be disabled.
PICK Indicates that the pick control unit is to be disabled.
BUTTON Indicates that the button control unit is to be disabled.
STROKE Indicates that the stroke control unit is to be disabled.
LOCATOR Indicates that the locator control unit is to be disabled.
VALUATOR Indicates that the valuator control unit is to be disabled.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.
13(2): An invalid operation was requested for the active graphic device.

SECTION 2.4.3: SUBROUTINE UGEVNT

This subroutine may be used to retrieve an event record from the event queue of the active graphic device. When this subroutine is called and the event queue is empty, there are a number of possible options for the programmer to select. These options are: (1) this subroutine may return immediately with an indication that no event is ready to be reported, (2) it may enter a wait state that will only be broken when an event occurs, or (3) it may enter a timed wait state that will be broken when an event occurs or when the time interval expires.

On some versions of the Unified Graphics System, the full flexibility of this subroutine may not be available. Possible restrictions are: (1) events may not be queued and only the last occurring event is available, (2) events will only be accepted while UGEVNT is active, and (3) timed waits may not be possible. The user should refer to the section on the Supported Host Computers for specific information.

The calling sequence is:

```
CALL UGEVNT(OPTIONS,TIME,STRING,IARRAY,XARRAY,YARRAY)
```

The parameters in the calling sequence are:

OPTIONS This parameter is present for consistency with the other calling sequences; no items will be recognized.
TIME A floating point value giving the length of time, in seconds, that the subroutine will remain in the wait state. If TIME has a negative value, the subroutine will wait until an event occurs. A zero value of TIME will cause the subroutine to check the event queue and return immediately.
STRING A character string which may contain output values.
IARRAY A fixed point array which may contain output values.

XARRAY A floating point array which may contain output values.

YARRAY A floating point array which may contain output values.

The data which will be returned for each of the event causing control units is:

1. Keyboard events: IARRAY(1) is set to 1, IARRAY(2) contains the length of the character string, and the first IARRAY(2) characters of STRING contain the input character string. The maximum length of the string that will be returned may be set with the KEYBOARD option in subroutine UGECTL. The character string will usually have all lower case characters that were entered at the keyboard translated to upper case, but this action may also be modified by subroutine UGECTL.
2. Pick events: IARRAY(1) is set to 2, IARRAY(2) contains the identification of the selected graphic segment, and IARRAY(3) contains the pick identification of the graphic item within the graphic segment.
3. Button events: IARRAY(1) is set to 3 and IARRAY(2) contains the index of the button that was pushed.
4. Stroke events: IARRAY(1) is set to 4, IARRAY(2) contains the number of stroke end points that are being returned, and the X and Y coordinates are in XARRAY and YARRAY. The maximum number of coordinates that will be returned may be set with the STROKE option in subroutine UGECTL. The stroke end points are in the coordinate system of the window in effect when UGEVNT is called.

If no event is being reported, IARRAY(1) will be set to zero.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active graphic device.

SECTION 2.4.4: SUBROUTINE UGECTL

This subroutine may be used to perform a number of special operations related to event reporting on a graphic device. In particular, sampled interactive controls may be read. When sampled controls are being read, this subroutine will wait for the graphic device operator to signal that the item is ready to be read. Multiple operations may be performed in one call to this subroutine as long as the operations are not contradictory.

The calling sequence is:

```
CALL UGECTL(OPTIONS,STRING,IARRAY,XARRAY,YARRAY)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the

following items:

KEYBOARD This item indicates that initial values are being specified for the keyboard input buffer. STRING gives the initial buffer content, and (XARRAY(1),YARRAY(1)) give the coordinates of the center of the first character in the current window. The default is a string of 32 blanks near the bottom of the display area. The maximum string length that can be given is 128.

LOCASE This option may be used with the KEYBOARD option. The default action in subroutine UGEVNT is to translate lower case characters to upper case. When this option is specified, the upper and lower case characters are transmitted exactly as entered at the keyboard.

BUTTON=<value> This item specifies the setting of the lights on the button control unit. The value should be a string of bits (zero's and one's) with a one indicating that the light is to be turned on. The default is for all of the lights to be off.

STROKE This item specifies parameters for the stroke control unit. IARRAY(1) gives the maximum number of strokes that can be reported. The default value is 32 and the maximum value is 128. XARRAY(1) gives the size of the minimum stroke size as a ratio of the full horizontal deflection. The default value is 1/128. YARRAY(1) gives the minimum time, in seconds, between the samplings of the stroke control unit. The default is 1/4 seconds.

LOCATOR This item causes the locator control unit to be read. (XARRAY(1),YARRAY(1)) give the coordinates of the locator in terms of the window in effect when UGECTL is called.

VALUATOR=<value> This item causes the specified valuator to be read. The result is put in XARRAY(1) and is a value between 0.0 and 1.0.

STRING A character string which may be used for both input and output.

IARRAY A fixed point array which may be used for both input and output.

XARRAY A floating point array which may be used for both input and output.

YARRAY A floating point array which may be used for both input and output.

The index and severity level of the errors detected by this subroutine are:

12(3): No graphic device is active at present.

13(2): An invalid operation was requested for the active

graphic device.

15(2): An attempt was made to read a disabled control unit on the active graphic device.

SECTION 2.5: CONTROLLING THE VIEWING WINDOW

The programmer describes most pictures by giving the two-dimensional coordinates of markers, lines, and character strings, etc. in the "world coordinate system". This data is mapped onto the "drawing space" of the graphic device. This section describes the means that are available to map a part of the world coordinate system onto a part of the drawing space.

The default drawing space available to the programmer is a square area with coordinates of (0.0, 0.0) at its lower left corner and (1.0, 1.0) at its upper right corner. Subroutine UGDSPC may be called to redefine the drawing space so that it has a different aspect ratio. After UGDSPC has been called, a window is available with (0.0, 0.0) at its lower left corner and programmer supplied values at its upper right corner. A unit distance in the X direction of the drawing space usually represents the same physical distance as a unit distance in the Y direction.

A second subroutine, UGWDOW, is provided to map a rectangular subset of the world coordinate system (the "window") onto a rectangular subset of the drawing space (the "view port"). When markers, lines and character strings are defined that extend outside the current window, these picture items are clipped at the current window boundary. These items are illustrated in Figure 2.5.1.

A third subroutine, UGSHLD, may be used to define up to four "shields". A shield is a rectangular subset of a window with the property that any part of a line within a shield is blanked out. The effect of a shield is also illustrated in Figure 2.5.1.

Once a picture is started, the drawing space is fixed until the next picture. However, the window, view port, and shields, may be redefined many times within a picture. It is also very important that the programmer remember that the current window parameters are not utilized until subroutine UGWRIT is called, and that picture items put into a graphic segment are not clipped until then.

When three-dimensional primitives are part of a picture, the programmer will normally have to specify how this data is to be projected onto the drawing space. On a two-dimensional graphic device, this projection cannot be changed after the picture has been drawn, but on a three-dimensional graphic device, the operator may control this view. This two-dimensions to three-dimensions projection is controlled by two subroutines, UG3WRD

and UG3TRN, and is independent of all of the two-dimensional control that has been described earlier except the drawing space.

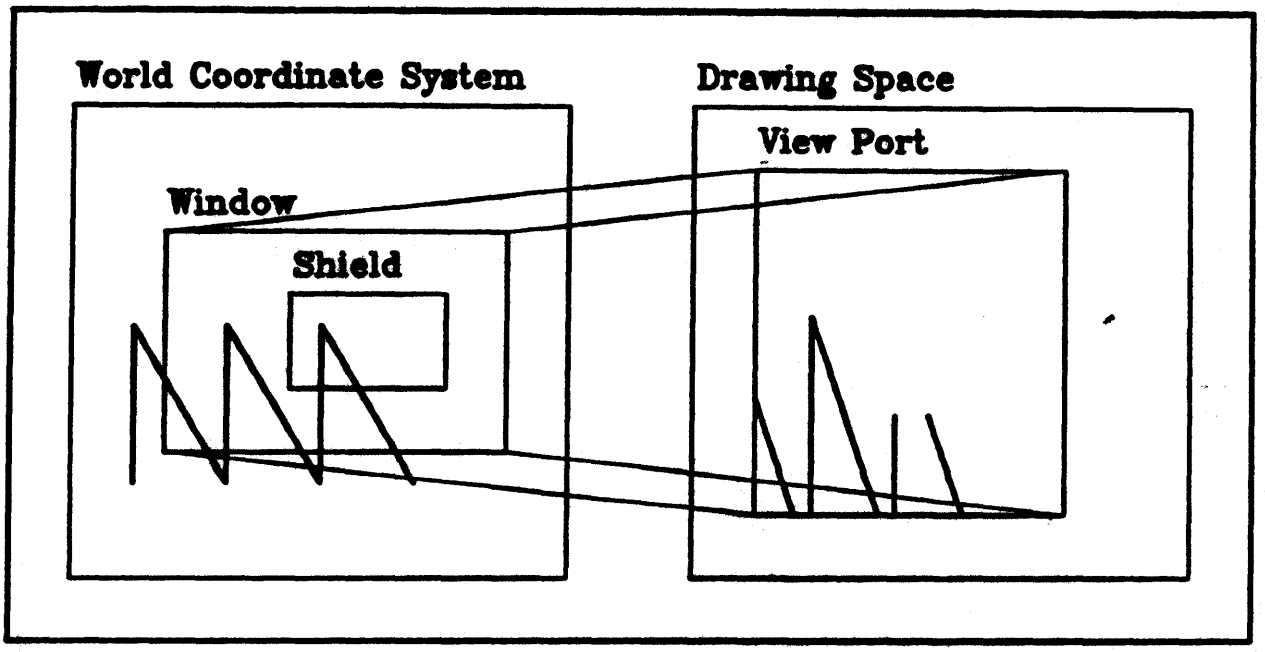


Figure 2.5.1: The Relation Between the World Coordinate System, its Window, the Drawing Space, and its View Port.

The X, Y, and Z coordinates that the user specifies are given in the "three-dimensional world coordinate system". Subroutine UG3WRD is used to define the "world volume" and "three-dimensional view port". The world volume is a rectangular parallelepiped which defines a subset of the three-dimensional world coordinate system and is similar in purpose to the two-dimensional world coordinate system. Unlike the two-dimensional world coordinate system, the world volume is not infinite in extent but represents a limited volume of space. For most three-dimensional graphic devices, the three-dimensional data is clipped at the world volume boundaries; any data outside these limits will not be displayed on a three-dimensional device. The three-dimensional view port is similar to the view port used for two-dimensional graphic primitives but is completely independent of it. The two and three-dimensional view ports are independent because it is difficult to assign a consistent meaning to multiple three-dimensional view ports and most of these assigned meanings complicate things considerably; while multiple two-dimensional view ports within a picture are simple and very useful.

Subroutine UG3TRN is used to define the "object volume", "eye point", "upward direction", and also to select a parallel projection or a point projection. The three-dimensional data that is to be viewed does not normally have to be within the object volume. The object volume is simply that part of the three-dimensional data that is to be initially viewed. However, the user should remember that on some low performance graphic devices, the three-dimensional data will be clipped at the object

volume boundaries. The object volume and eye point must be within the world volume.

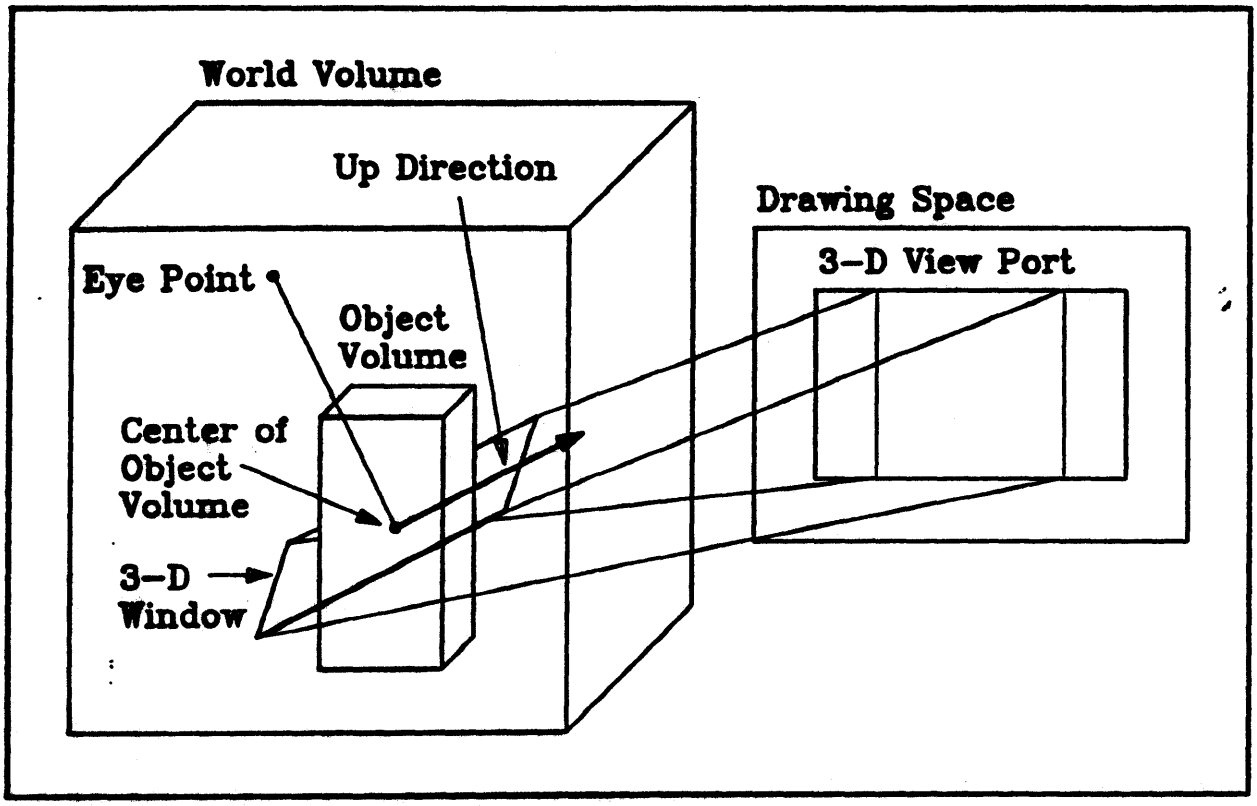


Figure 2.5.2: The Three-Dimensions to Two-Dimensions Projection.

Figure 2.5.2 illustrates how the projection is constructed. First, consider the world volume. The scaling along the coordinate axes of the three-dimensional space do not have to measure comparable quantities. For example, the X axis could measure energy, the Y axis time, and the Z axis interaction counts. To normalize these quantities, the first transformation applied by the Unified Graphics System is to map the world volume into a cube. If the coordinate axis do have consistent scaling, centimeters or inches for example, then it is the users responsibility to assure that the given world volume is a cube. Now consider a viewer at the eye point looking at the center of the object volume. This direction defines the "direction of projection". At the center of the object volume, a square "three-dimensional window" is constructed perpendicular to this direction. A parallel projection means that a point on the object is projected onto the three-dimensional window by moving in the direction of projection. A point projection means that the image of a point on the object is the intersection of a line through that point and the eye point with the three-dimensional window. The upward direction is projected into the plane of the three-dimensional window and the window is oriented so that one side of the square is parallel to this vector. The side of the square is just large enough so that the entire object volume is contained within the square under a parallel projection. The square three-dimensional window is then mapped onto the largest

central square area of the three-dimensional view port. Scissoring of the three-dimensional image takes place at the boundary of the three-dimensional view port.

For a point projection, there is another concept to be defined. The "near scissoring plane" is a plane parallel to the three-dimensional window and is situated between the eye point and the center of object. All marks, lines and text on the eye point side of this plane are deleted. If this were not done, parts of the object that lie behind the observer would be projected into the three-dimensional window causing great confusion in the picture.

On two-dimensional graphic devices, a three-dimensional picture element cannot be manipulated, but on a true three-dimensional graphic device, the console operator can manipulate these three-dimensional elements. The basic operations that the console operator may usually perform are rotate, translate, and zoom. These operations can be understood by referring to Figure 2.5.2.

There are three possible rotational operations. A side to side rotation is accomplished by rotating the eye point about the center of object while preserving the upward direction and the distance from the eye point to the center of object. A top to bottom rotation of the image is performed by rotating the eye point about the center of object in the plane defined by the eye point, center of object, and upward direction. In this case, the upward direction vector also takes part in the rotation. The third rotation operation consists of rotating the upward direction about the center of object in the plane of the three-dimensional window.

There are also three translation operations. Translations are performed by simultaneously moving the eye point and the object volume. A side to side translation is performed by moving in a direction perpendicular to the direction of projection and the upward direction. A top to bottom translation is done by moving in a direction parallel to the upward direction. The third translation is an in or out motion where the motion is parallel to the direction of projection. All three translations are needed to position the center of object so that rotations may be performed about any point in the world volume.

The zooming operation consists of holding the center of object fixed while the object volume and distance from the eye point to the center of object are simultaneously increased or decreased. The upward direction remains unchanged.

On some three-dimensional graphic devices, the transformation variables may be read from the device. This can be useful if it is necessary to create a copy of a transformed picture on some other graphic device.

There is one sense in which Figure 2.5.2 is very misleading. The object volume should never be as large as shown. If it were made

that large, it would be impossible to rotate the eye point very far without bumping into the boundaries of the world volume. For most uses, the sides of the world volume should be at least ten times the corresponding sides of the object volume. The eye point should be about two to four times the length of the object volume away from the object volume.

Another problem can be with the size of the three-dimensional image that is produced. Sometimes the image may appear smaller than a user expects. This is due to the way the size of the three-dimensional window is determined. The size of the three-dimensional window must be large enough to hold the projection of the object volume under all possible rotations. This means that the three-dimensional window must be big enough to contain the diagonal of the object volume, and not just large enough to contain a side.

SECTION 2.5.1: SUBROUTINE UGDSPC

This subroutine may be used to initialize the drawing space and redefine the aspect ratio of the picture. This operation must be performed before the first graphic segment of a picture is transmitted to the graphic device. That is, UGDSPC can be called immediately after calling UGOPEN, or between a call to UGPICT to clear the screen and the first call to UGWRIT. Any window or shields that are in effect when this subroutine is called to initialize the drawing space are deleted. After this subroutine has been called, a default view port exists which coincides with the drawing space, and a default window exists which has the same limits as the view port. The parameters supplied by subroutines UG3WRD and UG3TRN are also deleted by this subroutine when it is used to supply a new drawing space; those parameters are replaced by their default values. The drawing space parameters remain in effect, from picture to picture, until they are changed. This subroutine may also be used to retrieve the parameters of the current drawing space.

The calling sequence is:

```
CALL UGDSPC(OPTIONS,XSIZE,YSIZE,AFF)
```

The parameters in the calling sequence are:

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

- PUT Indicates that the drawing space parameters are being supplied to the Unified Graphics System.
- GET Indicates that the drawing space parameters are to be retrieved from the system.

If neither of the options items listed above is given, no operation is performed.

XSIZE The size of the picture in the X direction in arbitrary units. This argument is a floating point value and, for the PUT option, must be a positive

number.

YSIZE The size of the picture in the Y direction in arbitrary units. This argument is a floating point value and, for the PUT option, must be a positive number.

AFF The affinity value for XSIZE and YSIZE. This argument is a floating point value and, for PUT, must be a positive number between zero and one. A one means that XSIZE and YSIZE must be used exactly as given while a zero means that the Unified Graphics System is free to do the best it can to fill the plotting area. Values between zero and one result in XSIZE and YSIZE being honored to a lesser or greater extent. Some examples of the use of this parameter are given in the examples following the description of subroutine UGWDOW.

The index and severity level of the errors detected by this subroutine are:

1(3): The subroutine was called at an incorrect time. It must be called before the first graphic segment in a picture is transmitted to the graphic device.

2(3): The picture sizes are incorrect. They must both be positive.

3(3): The affinity parameter is incorrect. It must be greater than or equal to zero and less than or equal to one.

12(3): No graphic device is active at present.

If this subroutine is not called, the effect is as if the statement:

```
      CALL UGDSPC('PUT',1.0,1.0,1.0)
```

had been executed.

SECTION 2.5.2: SUBROUTINE UGWDOW

This subroutine will take an arbitrary window on the world coordinate system and map it onto a view port in the drawing space. Any shields that are in effect when this subroutine is called to define a window are deleted. The window remains in effect until it is changed by this subroutine or until a new picture is started. If a marker, line, character string, or polygon-fill extends outside a window, the part that is outside will be clipped and will not appear in the picture. This subroutine can also retrieve the parameters of the current window with respect to the drawing space.

The calling sequence is:

```
  CALL UGWDOW(OPTIONS,VIEWPRT,WINDOW)
```

The parameters in the calling sequence are:

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

PUT Indicates that the window parameters are being supplied to the Unified Graphics System.

GET Indicates that the window parameters are to be retrieved from the system.

If neither of the options items listed above is given, no operation is performed.

WINDOW This item indicates that the VEIWPRT parameter is given relative to the current window instead of the drawing space.

VIEWPRT A floating point array of dimension (2,2) which is used to define a rectangle in the drawing space or in the current view port if the option WINDOW is given. VIEWPRT(1,1) gives the low X value of the view port, VIEWPRT(2,1) gives the low Y value, VIEWPRT(1,2) gives the high X value, and VIEWPRT(2,2) gives the high Y value.

WINDOW A floating point array of dimension (2,2) which is used to define a rectangle within the world coordinate system. WINDOW(1,1) gives the low X value of the window, WINDOW(2,1) gives the low Y value, WINDOW(1,2) gives the high X value, and WINDOW(2,2) gives the high Y value.

The index and severity level of the errors detected by this subroutine are:

1(3): The view port parameter is incorrect. The low X and Y values must be less than the high values.

2(3): An incorrect overlay of the view port onto the drawing space has been specified. The view port must lie entirely within the drawing space.

3(3): The window parameter is incorrect. The low X and Y values must be less than the high values.

12(3): No graphic device is active at present.

A few examples may clarify the way subroutines UGDSPC and UGWDOW may be used. Consider the following situations:

Example 1: A program is to be written that is to generate pictures on a drum plotter that are five times as long in the X direction as they are in the Y direction. It is also required that the program produce pictures on an interactive terminal for quick verification of the pictures. When the pictures are viewed on the terminal, it is desired that the pictures fill the entire available screen. Thus, if a circle is drawn, it will appear as a circle on the drum plotter, but will be quite elliptical on the terminal.

This example is relatively simple and can be solved by the statement:

```
      CALL UGDSPC('PUT',5.0,1.0,0.0)
```

After this call, the current window on the world coordinate system runs from 0.0 to 5.0 in X and from 0.0 to 1.0 in Y. The

affinity value of 0.0 allows the Unified Graphics System to do the best it can on the graphic device being used. The requested aspect ratio can easily be met on the drum plotter, so, on that device, the picture is five physical units long in the X direction and one physical unit long in the Y direction. On the terminal, the only way the given aspect ratio could be met is to utilize only that part of the screen in a narrow horizontal band across the screen. Since the affinity value is zero, the programmer has stated that the preservation of the aspect ratio is not very important, so the Unified Graphics System uses the entire available screen.

There is a problem with the plotting of character strings produced by the programmed stroke generator that is illustrated by this example. The aspect ratio of the characters is preserved under all conditions, and the size of the characters is determined from the horizontal direction of the drawing space (for DSIZE) or from the current window (for SIZE). This means that the characters will be much larger on the drum plotter than they are on the terminal. Thus, in this example, characters that look correct on the drum plotter may be very small on the terminal. On the other hand, a title that looks correct on the terminal may become so tall on the drum plotter that part of it is clipped off.

<u>Example 2</u>: A program is to produce graphs and other similar plots, and the overall size is always to be 13 units in X by 10 units in Y. The graphs themselves are to be confined to the area from 2.6 units to 11.7 units in X and from 2.0 units to 9.0 units in Y. The area outside this inner area will be used for labels and titles. When the graphs are plotted, the data points are to be supplied in their own coordinate system.

The first step is to define the aspect ratio of the pictures. This can be done by:

```
      CALL UGDSPC('PUT',13.0,10.0,1.0)
```

Since the affinity value is 1.0, the picture will always be 13 units by 10 units in size. After subroutine UGDSPC has been called, the program may generate graphic segments with the labels, axes, and titles in it and call subroutine UGWRIT to transmit the segments to the graphic device. Now suppose the data for the graphs has been scanned and the minimum and maximum values of X and Y are XMIN, XMAX, YMIN, and YMAX. The code to define the window for the graphs is then:

```
      REAL          VPRT(2,2), WDOW(2,2)
      DATA          VPRT/2.6,2.0,11.7,9.0/
        ...
      WDOW(1,1)=XMIN
      WDOW(2,1)=YMIN
      WDOW(1,2)=XMAX
      WDOW(2,2)=YMAX
      CALL UGWDOW('PUT',VPRT,WDOW)
```

Now the program may generate graphic segments containing the data to be plotted, and UGWRIT may then be used to transmit the segments to the graphic device.

SECTION 2.5.3: SUBROUTINE UGSHLD

This subroutine may be used to define as many as four shielded areas. The shields are rectangular subsets of the current window. When markers, lines, or text produced by the programmed stroke generator extend within a shielded area, that part of the graphic data within the shield is eliminated. Text produced by the hardware character generator or polygon-fills are not shielded at present. In addition, three-dimensional data is not shielded. The four shielded areas may overlap or extend out of the current window. This subroutine can also retrieve the parameters of a shield or delete a shield. All existing shields are deleted when a new window or drawing space is defined.

The calling sequence is:

```
CALL UGSHLD(OPTIONS,SHIELD)
```

The parameters in the calling sequence are:

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

- PUT Indicates that the parameters of a shield are being supplied to the Unified Graphics System.
- GET Indicates that the parameters of a shield are to be retrieved from the system. If the indicated shield has not been defined, the returned values in SHIELD will all be zero.
- DELETE Indicates that a shield is to be deleted.

If none of the options items listed above is given, no operation is performed.

- SHIELD=<value> Selects the shield to be manipulated. Any value between one and four is permitted. If this item is not given, the default value is 1.

<u>SHIELD</u> A floating point array of dimension (2,2) which is used to define the extent of the shield relative to the current window. SHIELD(1,1) gives the low X value of the shield, SHIELD(2,1) gives the low Y value, SHIELD(1,2) gives the high X value, and SHIELD(2,2) gives the high Y value.

The index and severity level of the errors detected by this subroutine are:

1(3): The shield parameter is incorrect. The low X and Y values must be less than the high values.

2(3): The index of the selected shield is incorrect. It must be between one and four.

12(3): No graphic device is active at present.

SECTION 2.5.4: SUBROUTINE UG3WRD

This subroutine may be used to define the three-dimensional view port and the world volume. This operation must be performed before the first graphic segment of a picture is transmitted to the graphic device. That is, UG3WRD can be called immediately after calling UGOPEN, or between a call to UGPICT to clear the screen and the first call to UGWRIT. The three-dimensional view port is a rectangular subset of the drawing space. When the projection of three-dimensional data extends outside this three-dimensional view port, the data will be clipped at the boundary of this view port. The world volume is a rectangular parallelepiped in space which will contain all of the three-dimensional data. Calling subroutine UGDSPC causes the three-dimensional view port and world volume to revert to their default values. The three-dimensional transformation data in effect when this subroutine is called are deleted and replaced by their default values. The parameters supplied by this subroutine remain in effect, from picture to picture, until they are changed by this subroutine. This subroutine may also be used to retrieve the current values of the three-dimensional view port and the world volume.

The calling sequence is:

```
CALL UG3WRD(OPTIONS,VPRT3D,WRLDVOL)
```

The parameters in the calling sequence are:

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

- PUT Indicates that the parameters are being supplied to the Unified Graphics System.
- GET Indicates that the parameters are to be retrieved from the system.

If none of the options items listed above is given, no operation is performed.

<u>VPRT3D</u> A floating point array of dimension (2,2) which is used to define the extent of the three-dimensional view port with respect to the drawing space. VPRT3D(1,1) gives the low X value of the view port, VPRT3D(2,1) gives the low Y value, VPRT3D(1,2) gives the high X value, and VPRT3D(2,2) gives the high Y value.

<u>WRLDVOL</u> A floating point array of dimension (3,2) which is used to define the extent of the world volume. WRLDVOL(1,1) gives the low X value of the world volume, WRLDVOL(2,1) gives the low Y value, WRLDVOL(3,1) gives the low Z value, WRLDVOL(1,2) gives the high X value, WRLDVOL(2,2) gives the high Y value, and WRLDVOL(3,2) gives the high Z value.

The index and severity level of the errors detected by this subroutine are:

1(3): The subroutine was called at an incorrect time. It must be called before the first graphic segment in a picture is transmitted to the graphic device.

2(3): The view port parameter is incorrect. The low X and Y values must be less than the high values.
3(3): An incorrect overlay of the view port onto the drawing space has been specified. The view port must lie entirely within the drawing space.
4(3): The world volume parameter is incorrect. The low X, Y, and Z values must be less than the high values.
12(3): No graphic device is active at present.

If this subroutine is not called, a default three-dimensional view port and world volume is supplied. The default three-dimensional view port coincides with the drawing space. The default world volume extends from -4.5 to 5.5 in X, Y, and Z; that is, the world volume is a cube of 10 units on a side centered at (0.5, 0.5, 0.5).

SECTION 2.5.5: SUBROUTINE UG3TRN

This subroutine may be used to define the three-dimensions to two-dimensions transformation. It can supply the object volume, eye point, upward direction, and projection flag. The object volume is a rectangular parallelepiped which is a subset of the world volume. The eye point must also lie within the world volume. This transformation data supplied by this subroutine reverts to its default value when a new picture is begun. This subroutine can also retrieve the transformation data from the host computer, or, in some cases, from the graphic device itself.

The calling sequence is:

```
CALL UG3TRN(OPTIONS,OBJVOL,EYEPT,UPDIR,PFLAG)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:
- PUT Indicates that the parameters are being supplied to the Unified Graphics System.
- GET Indicates that the parameters are to be retrieved from the system.

If none of the options items listed above is given, no operation is performed.
- REMOTE This item may be used with the GET option. Without this option the transformation data is obtained from the host computer, with this option the data is obtained from the graphic device.

OBJVOL A floating point array of dimension (3,2) which is used to define the extent of the object volume. OBJVOL(1,1) gives the low X value of the object volume, OBJVOL(2,1) gives the low Y value, OBJVOL(3,1) gives the low Z value, OBJVOL(1,2) gives the high X value, OBJVOL(2,2) gives the high Y value, and OBJVOL(3,2) gives the high Z value.

EYEPT A floating point array of dimension 3 which specifies

the eye point.

UPDIR A floating point array of dimension 3 which specifies the upward direction.

PFLAG A floating point value which specifies the projection flag. A zero means that a parallel projection is to be used while a positive value means that a point projection is to be used. The value of the parameter for a point projection specifies the position of the near scissoring plane along the line from the eye point to the center of volume. The value must be between zero and one. A value near zero puts the near scissoring plane near the eye point while a value near one puts the plane near the center of object. For a point projection, a value of 0.01 is suggested.

The index and severity level of the errors detected by this subroutine are:

1(3): The object volume parameter is incorrect. The low X, Y, and Z values must be less than the high values.

2(3): The object volume parameter is incorrect. It must lie completely within the world volume.

3(3): The eye point parameter is incorrect. It must be inside the world volume and outside the object volume.

4(3): The up direction parameter for the three-dimensional view is invalid.

5(3): The projection flag parameter is incorrect. It must be zero or greater and less than one.

6(2): The active graphic device cannot return the current three-dimensional view information. Instead, the information available on the host computer has been returned.

12(3): No graphic device is active at present.

If this subroutine is not called, default values are provided. These defaults are defined by the statements:

```
WVDELX       = WRLDVOL(1,2)-WRLDVOL(1,1)
WVDELY       = WRLDVOL(2,2)-WRLDVOL(2,1)
WVDELZ       = WRLDVOL(3,2)-WRLDVOL(3,1)
OBJVOL(1,1)  = WRLDVOL(1,1)+0.45*WVDELX
OBJVOL(2,1)  = WRLDVOL(2,1)+0.45*WVDELY
OBJVOL(3,1)  = WRLDVOL(3,1)+0.45*WVDELZ
OBJVOL(1,2)  = WRLDVOL(1,2)-0.45*WVDELX
OBJVOL(2,2)  = WRLDVOL(2,2)-0.45*WVDELY
OBJVOL(3,2)  = WRLDVOL(3,2)-0.45*WVDELZ
EYEPT(1)     = 0.5*(WRLDVOL(1,2)+WRLDVOL(1,1))
EYEPT(2)     = 0.5*(WRLDVOL(2,2)+WRLDVOL(2,1))
EYEPT(3)     = 0.5*(WRLDVOL(3,2)+WRLDVOL(3,1))+0.25*WVDELZ
UPDIR(1)     = 0.0
UPDIR(2)     = 1.0
UPDIR(3)     = 0.0
PFLAG        = 0.01
```

These statements can best be understood by considering them in conjunction with the defaults for subroutine UG3WRD. In that case, the object volume is a cube of one unit on a side centered at (0.5, 0.5, 0.5), and the eye point is at (0.5, 0.5, 3.0).

Thus, if neither UG3WRD and UG3TRN are called, the default view is a point projection of the unit cube from a point two units above the cube. The cube is oriented so that the X and Y axes point in their usual directions and the z axis points at the observer.

There is a hierarchy to the subroutines in this section that is shown in Figure 2.5.3. The direction of the arrows in the figure shows which subroutines destroy data supplied by other subroutines. For example, starting a new picture causes the data supplied by previous calls to UGWDOW (and therefore UGSHLD) and UG3TRN to revert to their default values; it does not, however, change the data supplied by UGDSPC or UG3WRD. This hierarchy is not an arbitrary construct but is absolutely necessary. The shields must be deleted, for example, when a new window is defined because the shields were defined relative to the old window. The other parts of the hierarchy result from similar constraints.

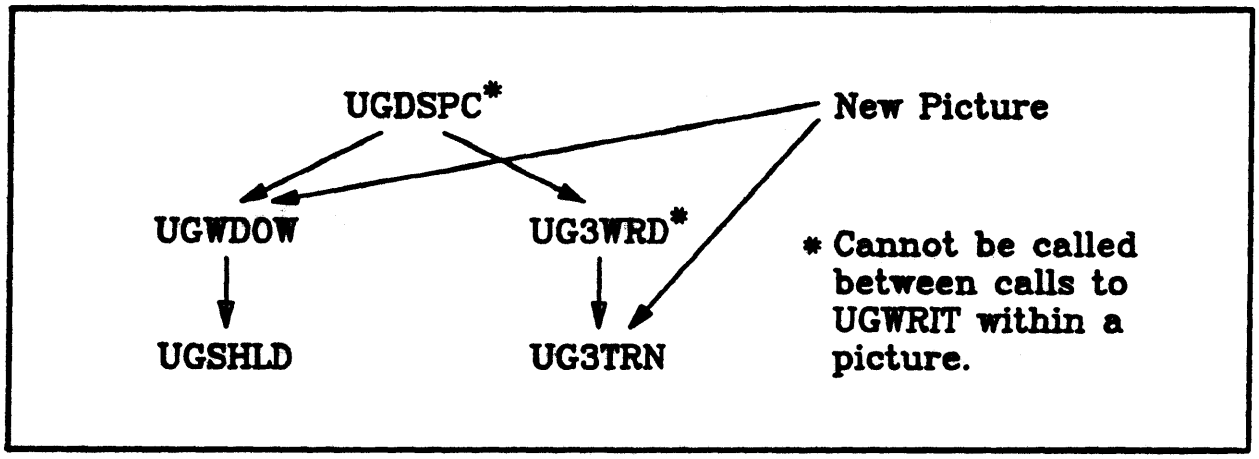


Figure 2.5.3: The Hierarchy of the Subroutines.

SECTION 2.6: THE EXTENDED CHARACTER SET

The characters from the extended character set may be added to a graphic segment by subroutine UGXTXT. The extended character set consists of the upper and lower case Roman, Greek, and Cyrillic alphabets, the numerals, and a great variety of special characters. In addition a flexible position and size control scheme, including subscripting and superscripting, is provided.

The extended character set may be produced in either of two fonts. The "simplex" font minimizes the number of strokes in the character while the "duplex" font produces characters that approximate the appearance of typeset characters. The full extended character set is described in the following table. The

table gives the primary and secondary character followed by its description. The symbol "x" stands for a blank. Notice that the primary and secondary characters are all selected from the basic character set.

The Upper Case Roman Alphabet:

Ax	Upper Case Roman A.
Bx	Upper Case Roman B.
Cx	Upper Case Roman C.
Dx	Upper Case Roman D.
Ex	Upper Case Roman E.
Fx	Upper Case Roman F.
Gx	Upper Case Roman G.
Hx	Upper Case Roman H.
Ix	Upper Case Roman I.
Jx	Upper Case Roman J.
Kx	Upper Case Roman K.
Lx	Upper Case Roman L.
Mx	Upper Case Roman M.
Nx	Upper Case Roman N.
Ox	Upper Case Roman O.
Px	Upper Case Roman P.
Qx	Upper Case Roman Q.
Rx	Upper Case Roman R.
Sx	Upper Case Roman S.
Tx	Upper Case Roman T.
Ux	Upper Case Roman U.
Vx	Upper Case Roman V.
Wx	Upper Case Roman W.
Xx	Upper Case Roman X.
Yx	Upper Case Roman Y.
Zx	Upper Case Roman Z.

The Lower Case Roman Alphabet:

AL	Lower Case Roman A.
BL	Lower Case Roman B.
CL	Lower Case Roman C.
DL	Lower Case Roman D.
EL	Lower Case Roman E.
FL	Lower Case Roman F.
GL	Lower Case Roman G.
HL	Lower Case Roman H.
IL	Lower Case Roman I.
JL	Lower Case Roman J.
KL	Lower Case Roman K.
LL	Lower Case Roman L.
ML	Lower Case Roman M.
NL	Lower Case Roman N.
OL	Lower Case Roman O.
PL	Lower Case Roman P.
QL	Lower Case Roman Q.
RL	Lower Case Roman R.
SL	Lower Case Roman S.
TL	Lower Case Roman T.
UL	Lower Case Roman U.

VL Lower Case Roman V.
WL Lower Case Roman W.
XL Lower Case Roman X.
YL Lower Case Roman Y.
ZL Lower Case Roman Z.

The Upper Case Greek Alphabet:
AF Upper Case Greek Alpha.
BF Upper Case Greek Beta.
GF Upper Case Greek Gamma.
DF Upper Case Greek Delta.
EF Upper Case Greek Epsilon.
ZF Upper Case Greek Zeta.
HF Upper Case Greek Eta.
QF Upper Case Greek Theta.
IF Upper Case Greek Iota.
KF Upper Case Greek Kappa.
LF Upper Case Greek Lambda.
MF Upper Case Greek Mu.
NF Upper Case Greek Nu.
XF Upper Case Greek Xi.
OF Upper Case Greek Omicron.
PF Upper Case Greek Pi.
RF Upper Case Greek Rho.
SF Upper Case Greek Sigma.
TF Upper Case Greek Tau.
UF Upper Case Greek Upsilon.
FF Upper Case Greek Phi.
CF Upper Case Greek Chi.
YF Upper Case Greek Psi.
WF Upper Case Greek Omega.

The Lower Case Greek Alphabet:
AG Lower Case Greek Alpha.
BG Lower Case Greek Beta.
GG Lower Case Greek Gamma.
DG Lower Case Greek Delta.
EG Lower Case Greek Epsilon.
ZG Lower Case Greek Zeta.
HG Lower Case Greek Eta.
QG Lower Case Greek Theta.
IG Lower Case Greek Iota.
KG Lower Case Greek Kappa.
LG Lower Case Greek Lambda.
MG Lower Case Greek Mu.
NG Lower Case Greek Nu.
XG Lower Case Greek Xi.
OG Lower Case Greek Omicron.
PG Lower Case Greek Pi.
RG Lower Case Greek Rho.
SG Lower Case Greek Sigma.
TG Lower Case Greek Tau.
UG Lower Case Greek Upsilon.
FG Lower Case Greek Phi.
CG Lower Case Greek Chi.

YG Lower Case Greek Psi.
WG Lower Case Greek Omega.

The Upper Case Cyrillic Alphabet:

AB Upper Case Cyrillic Ah.
BB Upper Case Cyrillic Beh.
VB Upper Case Cyrillic Veh.
GB Upper Case Cyrillic Geh.
DB Upper Case Cyrillic Deh.
EB Upper Case Cyrillic Yeh.
XB Upper Case Cyrillic Zheh.
ZB Upper Case Cyrillic Zeh.
IB Upper Case Cyrillic Ee.
1B Upper Case Cyrillic Ee S Kratkoy.
KB Upper Case Cyrillic Kah.
LB Upper Case Cyrillic El.
MB Upper Case Cyrillic Em.
NB Upper Case Cyrillic En.
OB Upper Case Cyrillic Oh.
PB Upper Case Cyrillic Peh.
RB Upper Case Cyrillic Err.
SB Upper Case Cyrillic Ess.
TB Upper Case Cyrillic Teh.
UB Upper Case Cyrillic Ooh.
FB Upper Case Cyrillic Ef.
HB Upper Case Cyrillic Kha.
CB Upper Case Cyrillic Tseh.
2B Upper Case Cyrillic Cheh.
3B Upper Case Cyrillic Shah.
4B Upper Case Cyrillic Shchah.
QB Upper Case Cyrillic Tvyordy Znak.
YB Upper Case Cyrillic Yery.
5B Upper Case Cyrillic Myakhki Znak.
6B Upper Case Cyrillic Eh Oborotnoye.
WB Upper Case Cyrillic Yoo.
JB Upper Case Cyrillic Yah.

The Lower Case Cyrillic Alphabet:

AC Lower Case Cyrillic Ah.
BC Lower Case Cyrillic Beh.
VC Lower Case Cyrillic Veh.
GC Lower Case Cyrillic Geh.
DC Lower Case Cyrillic Deh.
EC Lower Case Cyrillic Yeh.
XC Lower Case Cyrillic Zheh.
ZC Lower Case Cyrillic Zeh.
IC Lower Case Cyrillic Ee.
1C Lower Case Cyrillic Ee S Kratkoy.
KC Lower Case Cyrillic Kah.
LC Lower Case Cyrillic El.
MC Lower Case Cyrillic Em.
NC Lower Case Cyrillic En.
OC Lower Case Cyrillic Oh.
PC Lower Case Cyrillic Peh.
RC Lower Case Cyrillic Err.

SC Lower Case Cyrillic Ess.
TC Lower Case Cyrillic Teh.
UC Lower Case Cyrillic Ooh.
FC Lower Case Cyrillic Ef.
HC Lower Case Cyrillic Kha.
CC Lower Case Cyrillic Tsah.
2C Lower Case Cyrillic Cheh.
3C Lower Case Cyrillic Shah.
4C Lower Case Cyrillic Shchah.
QC Lower Case Cyrillic Tvyordy Znak.
YC Lower Case Cyrillic Yery.
5C Lower Case Cyrillic Myakhki Znak.
6C Lower Case Cyrillic Eh Oborotnoye.
WC Lower Case Cyrillic Yoo.
JC Lower Case Cyrillic Yah.

The Numerals:
0# Numeral 0.
1# Numeral 1.
2# Numeral 2.
3# Numeral 3.
4# Numeral 4.
5# Numeral 5.
6# Numeral 6.
7# Numeral 7.
8# Numeral 8.
9# Numeral 9.

Special Symbols from the Basic Character Set:
Blank.
+# Plus Sign.
-# Minus Sign.
*# Asterisk.
/# Slash Mark.
=# Equal Sign.
.# Period.
,# Comma.
(# Left Parenthesis.
)# Right Parenthesis.

Special Symbols for Punctuation:
.P Colon.
,P Semi-Colon.
EP Exclamation Mark.
UP Question Mark.
IP Interrobang.
AP Apostrophe.
QP Quotation Marks.
PP New Paragraph.
DP Dagger.
FP Double Dagger.

Additional Special Symbols:
DS Dollar Sign.
CS Cent Sign.

+S Ampersand.
PS Pound Sign.
AS At Sign.
%S Percent Sign.
VS Vertical Line.
WS Double Vertical Line.
US Underline.
NS Not Sign.
/S Backwards Slash.
(S Left Bracket.
)S Right Bracket.
LS Left Brace.
RS Right Brace.
BS Left Angle Bracket.
ES Right Angle Bracket.
XS Accent Mark.
TS Caret Mark.

Mathematical Special Symbols:
.M Dot Product.
XM Cross Product.
/M Division Sign.
PM Group Plus.
*M Group Multiply.
+M Plus or Minus.
-M Minus or Plus.
LM Less Than.
GM Greater Than.
MM Less Than or Equal.
HM Greater Than or Equal.
NM Not Equal.
=M Identically Equal.
AM Approximately Equal.
CM Congruent To.
SM Similar To.
RM Proportional To.
TM Perpendicular To.
2M Square Root.
DM Degrees.
IM Integral Sign.
JM Line Integral.
YM Partial Derivative.
ZM Del.
(M Left Floor Bracket.
)M Right Floor Bracket.
BM Left Ceiling Bracket.
EM Right Ceiling Bracket.
8M Infinity.

Set Theoretic Special Symbols:
ET Existential Quantifier.
AT Universal Quantifier.
MT Membership Symbol.
NT Membership Negation.
IT Intersection.

UT Union.
LT Contained In.
GT Contains.
KT Contained In or Equals.
FT Contains or Equals.

Arrows and Pointers:
UW Up Arrow.
DW Down Arrow.
LW Left Arrow.
RW Right Arrow.
BW Left/Right Arrow.

Physics Special Symbols:
HK H-Bar.
LK Lambda-Bar.

Astronomical Special Symbols:
HA Sun.
MA Mercury.
VA Venus.
EA Earth.
WA Mars.
JA Jupiter.
SA Saturn.
UA Uranus.
NA Neptune.
PA Pluto.
OA Moon.
CA Comet.
*A Star.
XA Ascending Node.
YA Descending Node.
KA Conjunction.
QA Quadrature.
TA Opposition.
0A Aries.
1A Taurus.
2A Gemini.
3A Cancer.
4A Leo.
5A Virgo.
6A Libra.
7A Scorpius.
8A Sagittarius.
9A Capricornus.
AA Aquarius.
BA Pisces.

Plotting Symbols:
00 Vertical Cross.
10 Diagonal Cross.
20 Diamond.
30 Square.
40 Fancy Diamond.

50 Fancy Square.
60 Fancy Vertical Cross.
70 Fancy Diagonal Cross.
80 Star Burst.
90 Octagon.

Special Drawing Symbols:
UD Underscore.
OD Overscore.

Blank and Vertical Movement Control:
#U Null.
0U Backwards Blank.
1U Half Blank.
2U Half Backwards Blank.
3U Third Blank.
4U Third Backwards Blank.
5U Sixth Blank.
6U Sixth Backwards Blank.
1V Half Up Movement.
2V Half Down Movement.
3V Third Up Movement.
4V Third Down Movement.
5V Sixth Up Movement.
6V Sixth Down Movement.

Subscript and Superscript Control:
0X Enter Subscript Mode.
1X Leave Subscript Mode.
2X Enter Superscript Mode.
3X Leave Superscript Mode.

Character Size Control:
0Y Increase Size by One-Half.
1Y Decrease Size by One-Third.

Position Control:
0Z Put Current State in First Save Area.
1Z Restore State from First Save Area.
2Z Put Current State in Second Save Area.
3Z Restore State from Second Save Area.
4Z Put Current State in Third Save Area.
5Z Restore State from Third Save Area.
6Z Put Current State in Fourth Save Area.
7Z Restore State from Fourth Save Area.

In addition to the primary and secondary character pairs shown above, most of the characters representable on the host computer will be produced with a secondary character of blank. Thus, if the primary character is a lower case roman letter and the secondary character is a blank, then the proper character will be produced.

The underscore and overscore characters in the above table have some special properties. The purpose of these characters is to allow the programmer to draw lines under or over a line of text. Two consecutive underscore 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 overscore will also join properly with the square root character to form a full radical sign.

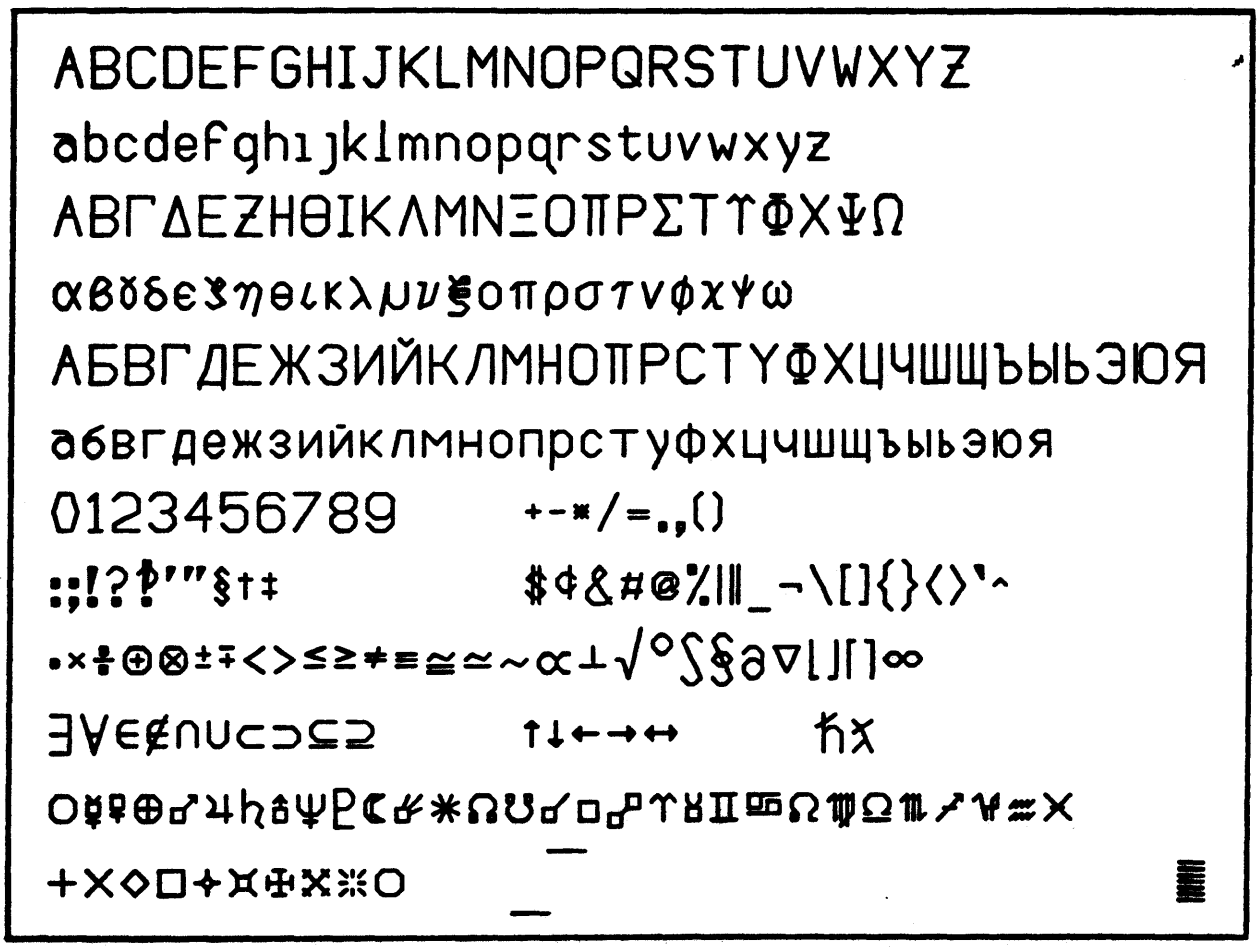


Figure 2.6.1: The Extended/Simplex Character Set.

The extended character generators produce characters of differing widths and heights; thus the upper case letter "M" is about twice as wide as the upper case "I", and most lower case letters are about 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 group of position control characters have 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 four independent save-restore control character pairs available. The scope of these save-restore pairs is a single string supplied by UGXTXT; that is, you cannot save a position in one call to UGXTXT 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.

The extended/simplex character set is shown in Figure 2.6.1, and the extended/duplex character set is shown in Figure 2.6.2. The order of the characters in the figures is the same as in the preceding table. The character in the lower right of Figure 2.6.1 and Figure 2.6.2 is produced when an invalid character pair is specified. The average number of strokes per character in the simplex font is 6.3, and the maximum number is 17. The average number of strokes per character in the duplex font is 16.9, and the maximum number is 63.

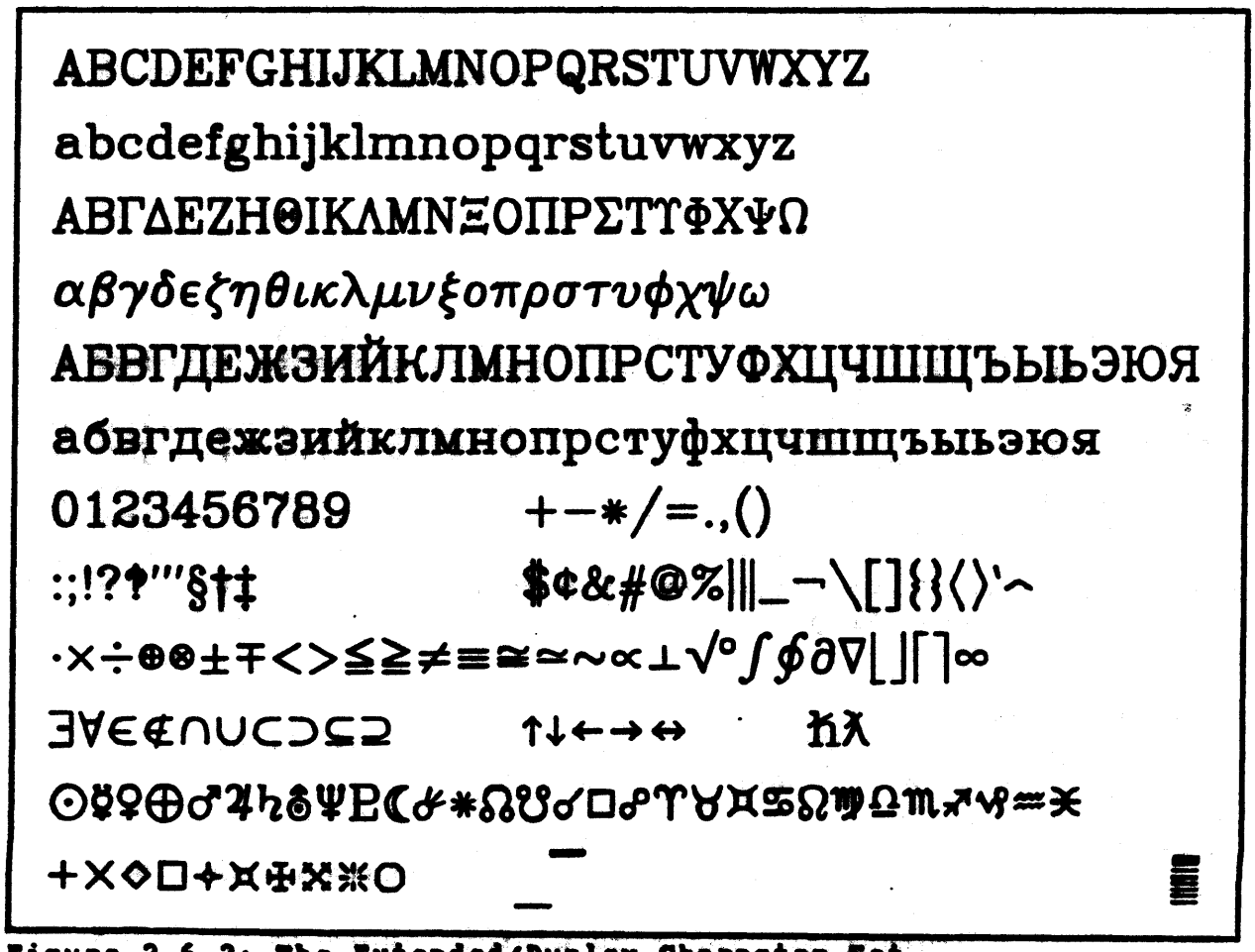


Figure 2.6.2: The Extended/Duplex Character Set.

SECTION 2.6.1: SUBROUTINE UGFONT

On most versions of the Unified Graphics System, the font of the extended character set may be selected dynamically. For those systems, this subroutine will accomplish that function. The user should remember that the extended character generator is not actually used until subroutine UGWRIT or UGCTOL is called.

The calling sequence is:

```
CALL UGFONT(OPTIONS)
```

The parameter in the calling sequence is:

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

- SIMPLEX Selects the simplex font of the extended character generator.
- DUPLEX Selects the duplex font of the extended character generator.

The index and severity level of the errors detected by this subroutine are:

1(2): For some versions of the Unified Graphics System, the font of the extended character set is not dynamically selectable.

SECTION 2.6.2: SUBROUTINE UGCTOL

This subroutine will accept a primary and secondary character string as input, invoke the current extended character generator, and supply the resulting line segments and blanking bits as output. The line segments and blanking bits are made available in a form that is the same as that required by subroutine UGPLIN. Before passing these line segments on to UGPLIN, the programmer may transform them in any manner. One simple thing a programmer could do is create slanted lettering. Another use of this subroutine could be the creation of perspective views of a letter in 3-Space. No clipping is performed on the generated line segments.

The calling sequence is:

```
CALL UGCTOL(OPTIONS,XCOORD,YCOORD,PRITXT,SECTXT,NSIZE,
            XARRAY,YARRAY,NCOORD,BBITS)
```

The parameters in the calling sequence are:

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

- SIZE=<value> The size of the characters. The default value is 0.015.
- 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.

LEFT, RIGHT, and CENTER: These items specify the part of the character string that is being positioned by the X and Y coordinates. The default is LEFT, that is, the X and Y coordinates specify the position of the center of the first character in the string.

FIXSIZE and NOFXSIZ: These items indicate whether or not the characters are to be proportionally spaced or are to have constant spacing as given by the SIZE item. The default is NOFXSIZ which results in proportionally spaced characters.

LAST and NEXT: Either of these items will suppress the generation of the strokes. Instead XARRAY(1) and YARRAY(1) will contain the coordinates of the center of the last character in the string or the coordinates of what would be the center of the next character. XARRAY(2) will contain the size change factor. The size change factor will usually be 1.0 but terminating the string in subscript mode will, for example, result in a value of two-thirds. LAST can be useful in pre-scanning titles so that they can be drawn in constrained areas. NEXT can be useful in concatenating longer strings of characters.

XCOORD The floating point X coordinate of the part of the character string being positioned. This may be the center of the first character, the center of the last character, or the center of the string.

YCOORD The floating point Y coordinate of the part of the character string being positioned.

PRITXT The string containing the primary characters.

SECTXT The string containing the secondary characters.

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

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

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

<u>NCOORD</u> The number of coordinates that were stored in XARRAY and YARRAY.

<u>BBITS</u> A bit string where the blanking bits will be stored.

The index and severity level of the errors detected by this subroutine are:

1(2): The string lengths are invalid. The lengths must be at least 1 and at most 1024.

2(2): There is not enough space available in the output arrays to contain the generated line segment data.

14(3): The extended character module is not available. The executable load module will have to be recreated.

SECTION 2.7: 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 a FORTRAN COMMON block. The program continues executing.
2. Errors: The error information is saved in a COMMON block, 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. The program continues executing.
3. Severe Errors: The error message is printed, and the program is terminated without a memory dump or subroutine trace-back.
4. Terminal Errors: The error message is printed, and the program is terminated with a memory dump and/or subroutine trace-back.

The error message is usually written to FORTRAN unit 6. Each individual error message is printed a maximum of 8 times. After 8 times, the error message is not printed but all other actions remain the same. Both the FORTRAN unit number and the maximum print count may be changed by using subroutine UGMCTL.

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 in the COMMON block after a subroutine has been called. The COMMON block is declared by:

```
SAVE          /UGERRD/
COMMON        /UGERRD/UGELV,UGENM,UGEIX
INTEGER       UGELV
CHARACTER*8   UGENM
INTEGER       UGEIX
```

When a Minor Error or Error has occurred, UGELV will contain the severity level, UGENM will contain the subroutine name (padded on the right with blanks), and UGEIX will contain the index. If no error has occurred, UGELV and UGEIX will contain zero and UGENM 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 is:

```
SUBROUTINE     UGXERR(LEVEL,SNAME,INDEX)
INTEGER        LEVEL
CHARACTER*8    SNAME
INTEGER        INDEX
  ...
END
```

This error processing subroutine can do almost anything to try to recover from the error, including calling other subroutines in this system. However, it is the responsibility of the user to assure that no subroutine is called recursively. A serious problem can arise if one of the subroutines called by the user's error processing subroutine detects an error because this implies recursive use of the error processor. If an attempt is made to call the error processor recursively, the message for the first error will be printed followed by a message with a subroutine name of UGXERR whose index, severity level, and description is:

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

When the user's 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 nontrivial example of the use of 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 segment is cleared by calling procedure UGINIT with the CLEAR option.
2. The graphic segment generators are used to create the graphic segment.
3. The graphic segment is transmitted to the graphic device.

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

1. Put the graphic segment into a COMMON block.
2. Prepare a UGXERR subroutine which recognizes a graphic segment overflow error as signaled by the graphic segment generators (these are distinguished by an index number of 11). In this case, UGXERR should transmit the graphic segment to the device by calling UGWRIT and then clear the graphic segment by calling UGINIT with the CONTINUE option. The CONTINUE option is needed here because of the way UGLINE 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 segment generators. When they find that all of the new data will not fit, they report the condition to the error processor. If the error processor returns, the segment generators again try to put the data into the graphic segment before returning.

Error indexes of 1 through 9 have a meaning which varies for each subroutine. To identify one of these errors at execution time, you must check both the subroutine name and the error index. Error indexes 11 through 19 have a common meaning within a group of subroutines. Thus, an index of 11 always means graphic segment overflow. There are also a few cases in the Unified Graphics System where an "impossible" error may be detected. These are characterized by a direct call to subroutine UGZ001 (which terminates execution) without any error message being produced. If such an error occurs, you should seek expert help.

SECTION 2.7.1: SUBROUTINE UGRERR

This subroutine gives the programmer the ability to call the error processor. Any error reported in this way will be handled in the same manner as described above except that a full description of the error will not be printed.

The calling sequence is:

```
CALL UGRERR(LEVEL,SNAME,INDEX)
```

The parameters in the calling sequence are:

LEVEL	The level number of the error.
SNAME	A character string of 8 characters giving the name of the subroutine identifying the error.
INDEX	The index of the error.

SECTION 2.8: 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 options lists to obtain the information from them. The programmer may make use of this subroutine.

SECTION 2.8.1: 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. These limitations are dictated by the fact that this subroutine is called repeatedly and must be as efficient as possible.

The calling sequence is:

```
CALL UGOPTN(OPTIONS,INDATA,EXDATA)
```

The parameters in the calling sequence are:

OPTIONS The options list which will be scanned. The maximum number of items that an options list may contain is 32.

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 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 index 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 a multiple of 4 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 output structure if the index (the third item in the descriptor) is positive. If the index is negative, the absolute value of the given index is used and the type dependent value is OR-ed into the output structure.
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.
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.
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 and stored in the output structure.
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 and stored in the output structure.

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 character other than zero or one in the bit field of a type 5 item, (3) a length of a string in a type 4 or 5 item which is longer than provided for in the output structure, or (4) more than the permitted number of items in the character string being scanned. In addition, misspelled items will not be recognized and will be ignored.

The following 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 integers for the output. Therefore:

```
      INTEGER        EXDATA(2),EXFLAG,EXINTG
      EQUIVALENCE    (EXFLAG,EXDATA(1)),(EXINTG,EXDATA(2))
```

Now that the output structure is known, we can specify the input structure as follows:

```
      INTEGER        INDATA(11)
      DATA           INDATA/2,1,4,1,5,'FLAG',
     X                      2,3,2,0,'INT '/
```

With these structures 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
      EXINTG=0
      CALL USOPTN(OPTNS,INDATA,EXDATA)
```

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

SECTION 3: THE SUPPORTED GRAPHIC DEVICES

This section contains a description of each of the graphic devices supported by the Unified Graphics System. The description of each device includes any special hardware capabilities; whether the device may be used in the non-interactive, slave-display, or interactive modes; whether the device is a storage, raster-scan, or refresh device; and other device-dependent information.

SECTION 3.1: THE CONSOLE ON WHEELS (COW) OF THE SLC PROJECT

This device was designed and constructed at SLAC for use as a portable monitor on the Stanford Linear Collider (SLC) Project. A COW is a slave-display refresh graphic device which includes a color display. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDDXCOW

A COW consists of an INTEL 8086 single board computer and a pair of video frame buffers attached to display monitors. One display is a color graphic monitor while the other is a monochrome monitor with a touch panel. The software drives the display monitors as if it had a resolution of 1024 by 1024 but it is normal to use a 512 by 512 monitor. This rescaling is handled internally and should not concern the user. The COW is connected to the VAX computer over the SLC-Network. Lines, points, and hardware-generated characters may be drawn. All of the colors supported by the Unified Graphics System will be produced on the color monitor. The character size is equivalent to:

DSIZE=0.01369

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

COW#=<value> The identification of the COW to be used. No default value is supplied for this parameter; it must be given when UGOPEN is called.

SCREEN=<value> The identification of the screen to be used. The default value is 1 which writes to the color monitor.

XMAX=<value> The maximum X raster value. The default value is 1023.

YMAX=<value> The maximum Y raster value. The default value is 1023.

XSIZ=<value> The size of the monitor in centimeters in the horizontal direction. The default value is 25.4

YSIZ=<value> The size of the monitor in centimeters in the vertical direction. The default value is 25.4

At present the module SDDXCOW is not in the normal Unified Graphics System directory on the VAX; instead it is maintained in a separate directory.

SECTION 3.2: THE DEC GIGI COLOR GRAPHICS TERMINAL

This graphic device consists of a keyboard with a color television monitor attached to it. GIGI is an acronym for General Imaging Generator and Interpreter. The terminal is made by Digital Equipment Corporation. It is a raster-scan graphic device and may be used in the non-interactive, slave-display, or interactive modes. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SEQGIGI, SDDGIGI, DECGIGI

The GIGI color graphics terminal has a keyboard, an audible alarm, and a LOCATOR control unit. The CRT has a resolution of 768 rasters horizontally and 240 vertically. These units can display points, lines, and hardware-generated characters in a wide variety of sizes. The smallest and largest sizes are equivalent to:

DSIZE=0.01879 (smallest size)
DSIZE=0.30063 (largest size)

Graphic primitives may be of any color and can be put in the blinking mode. The terminal contains a microprocessor with a BASIC language interpreter and can be used as an independent computer system.

When this device is used in the non-interactive mode, a job is run which generates a file containing the graphic orders to draw the pictures. Each picture consists of: (1) a short title, (2) a clear screen order, (3) the graphic orders themselves, and (4) commands to sound the audible alarm. The pictures are separated by IEBUPDTE control cards; that is, a card:

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

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

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

and the last card in the file is:

```
./        ENDUP
```

A program is supplied by the Unified Graphics System to allow a user to display the pictures on a terminal. On the VAX computers the program is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ RUN UGSYSTEM:DPICGIGI
```

In the interactive mode, this graphic device supports the KEYBOARD and LOCATOR control units. When the LOCATOR is being read, a cross-hair appears on the screen. The console operator can move this indicator around with the arrow keys, or with the arrow keys in conjunction with the shift key for faster movement.

When the indicator is properly positioned, the console operator should strike one of the keys for a normal printing character or the carriage return to send the screen coordinates to the host computer.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file for non-interactive use of this device. The default value is UGDEVICE.DAT on the VAX computers.

PICTID=<value> This item gives the first four characters for the picture name in the non-interactive mode. The default value is PICT.

PICTSQ=<value> This item supplies a numeric value which is incremented for each picture. It is converted to three digits and concatenated to the end of the PICTID value to form the complete picture name. The default value is 1.

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default is TT.

There are a few difficulties associated with this device. First, the vertical resolution of 240 rasters can be inadequate for many pictures. Second, when two graphic primitives of different colors are very close, the colors tend to "bleed" from the last drawn to the earlier drawn item. Finally, the terminal has an immense number of modes and states that it can be put into; if certain important states are incorrectly set, the terminal will not function in the desired manner.

SECTION 3.3: THE DEC VAXSTATIONS

The VAXSTATION II/GPX and VAXSTATION 2000 are made by Digital Equipment Corporation. They are very versatile devices, and the Unified Graphics System supports them as a refresh display device in the slave-display or interactive modes using DEC's UIS protocol. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers SDDVST2, DECVST2

The basic VAXSTATION consists of a MicroVAX II, a monochrome or color CRT, an audible alarm, a keyboard, and a mouse. A user normally runs with a number of "windows", that is, independent, rectangular, possibly overlapping, displays on the screen, and the Unified Graphics System writes to one of these windows. Note that this use of the word "window" is different than its use in most graphic packages. It is also possible, by using multiple graphic devices, to have the Unified Graphics System write to

more than one window in the slave-display mode. When used in the interactive mode, the KEYBOARD, PICK, BUTTON, and LOCATOR control units are available. To use the keyboard, it is the users responsibility to attach the physical keyboard to the window that the Unified Graphics System is using. The 36 buttons that the Unified Graphics System supplies are activated by the keys on the numeric keypad. The pattern was chosen to be consistent with the ASCII terminal emulation on the IBM computer. Buttons 1 through 12 are activated by the PF1, PF2, PF3, 7, 8, 9, 4, 5, 6, 1, 2, and 3 keys respectively. Striking the 0 key immediately before one of these keys adds 12 to its value, and striking the comma key adds 24 to its value. When the PICK is active, the graphic cursor's shape will be that of a square with lines radiating outward at the corners. The square indicates the aperture of the pick's view. When the LOCATOR is active, the cursor's shape is that of a simple cross. Any key on the mouse will signal the PICK or LOCATOR control units. The full resolution of the screen is 1024 pixels horizontally by 864 vertically, although a window usually uses less than the full screen. The unit can draw points, lines, filled polygons, and horizontal hardware-generated characters in a number of fonts. The Unified Graphics System supports three fonts whose character sizes are equivalent to:

DSIZE=0.00983 (small size)
DSIZE=0.01311 (medium size)
DSIZE=0.01967 (large size)

when the full width of the screen is used. When less than the full width of the screen is used, the corresponding value of DSIZE is proportionally larger. Graphic primitives may be of any color.

On color graphic devices, all of the colors supported by the Unified Graphics System may be used and filled polygons may be of any color. On monochrome devices, all colors, including BLACK, are drawn in the normal color. For these monochrome devices, the interior of filled polygons will be cleared and the border will then be drawn.

Since this is a refresh graphic display, individual graphic segments may be identified and individually deleted. In addition, individual graphic segments may take part in the OMIT/INCLUDE operations of subroutine UGPICT provided the VDID and WDID parameters that are described below are not used.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

GENIL This item indicates that the intensity level data is to be simulated. Normally, all lines are one pixel wide. When GENIL is used, a MEDIUM line is two pixels wide, a VDIM or DIM line is one pixel wide, and a BRIGHT or VBRIGHT line is three pixels wide.

CHANNEL=<value> This specifies an internal symbol for the output device. The default value is SYS$WORKSTATION and this value should not normally be changed.

XSIZ=<value> The required horizontal size of the display-

window in centimeters. The default value is 25.0.

YSIZ=<value> The required vertical size of the display-window in centimeters. The default value is 20.0.

TITLE=<value> The title of the window to be created. This will appear in a band at the top of the window. The default value is "Unified Graphics System". If no title is wanted, the user can eliminate this title by supplying the item TITLE=' '.

VDID=<value> The virtual-display value returned by subroutine UIS$CREATE_DISPLAY. This item is only available in the slave-display mode and should not normally be used.

WDID=<value> The display-window/display-viewport value returned by subroutine UIS$CREATE_WINDOW. This item is only available in the slave-display mode and should not normally be used.

XMIN=<value> The low X limit of the display-window. The default value is 0.0. This item is only available in the slave-display mode and should not normally be used.

YMIN=<value> The low Y limit of the display-window. The default value is 0.0. This item is only available in the slave-display mode and should not normally be used.

XMAX=<value> The high X limit of the display-window. The default value is 1.0. This item is only available in the slave-display mode and should not normally be used.

YMAX=<value> The high Y limit of the display-window. The default value is 1.0. This item is only available in the slave-display mode and should not normally be used.

This graphic device uses the concepts of "virtual-display" and "display-window/display-viewport". These are defined by calling the system subroutines UIS$CREATE_DISPLAY and UIS$CREATE_WINDOW. Normally, the Unified Graphics System will call these subroutines. However, in the slave-display mode, the capability exists for a user to call these subroutines and supply the results to the Unified Graphics System. In this latter case, additional manipulations of the window, such as panning and zooming, may be performed in the calling program. When the Unified Graphics System generates these items, the user may supply the XSIZ, YSIZ, and TITLE items. If the virtual-display and display-window/display-viewport are being supplied to the Unified Graphics System, then the items VDID, WDID, XMIN, YMIN, XMAX, and YMAX may be supplied. In this second case, both VDID and WDID must be given. If only one of them is specified, it will be ignored and the Unified Graphics System will create its own virtual-display and display-window/display-viewport. When VDID and WDID are given, it is also the users responsibility to set up a color map defining the color indices 0 through 7. The color indices should be the same as those defined in subroutine USDEFL except that BLACK is index 0. On monochrome devices, only color indices of 0 (BLACK) and 1 (WHITE) will be used. It is also important to realize that the values of XMIN, YMIN, XMAX, and YMAX have nothing to do with the actual size of the picture or its aspect ratio.

There is a problem with the use of the PICK control unit in the interactive mode. A graphic primitive will be detected if the pick's aperture intersects the smallest rectangle with vertical and horizontal sides that surrounds the graphic primitive. This works fine if you are trying to obtain hits on character strings, points, or compact groups of lines. It works very poorly on a simple diagonal line because a hit will be detected anywhere in the rectangle that encloses the line.

SECTION 3.4: THE GRINNELL GMR-27 DISPLAY SYSTEM

This graphic device consists of a controller which can have a number of television monitors connected to it. The device is a raster-scan graphic device and may be used in the slave-display mode only. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDDGRIN

The GRINNELL Display System consists of a controller with a number of television monitors connected to it. The controllers can be purchased in a wide variety of configurations. The Unified Graphics System treats each monitor of differing capability as a distinct type. Multiple monitors of the same type and on the same controller can be handled as a single graphic device; they will always display the same picture. The things that distinguish one type of monitor from another are: (1) whether the monitor is black and white or color, (2) the number of memory planes allocated to the monitor, (3) the meaning of the bits in a memory plane, and (4) the presence or absence of a look-up table. The controllers can also differ from each other in a number of other ways. A large number of options are provided in UGOPEN so that the user may fully describe the unit to be used.

In addition to being able to display lines, points, and text, the color devices will also support the polygon-fill operation to a limited degree. If the polygon is a rectangle with vertical and horizontal sides, then it can be filled with color, otherwise, the default action of simply drawing the border occurs.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

CHANNEL=<value> The identification of the I/O controller. The default value is GRA0.

UNITS=<value> This item selects the television monitors on the controller which are to be used. The value of this item is an integer which is the sum of the display units to be used where the units are numbered 1, 2, 4, 8,... Thus, if the first and third television monitors are to be used, this parameter should have a value of

5. The default value is 1.

BWTYPE=<value> This item specifies that black and white monitors are being used and selects the type. Valid types are: (1) one memory plane with no look-up table, (2) two memory planes giving three intensity levels with no look-up table, (3) two memory planes with a look-up table giving a mixture of intensity levels and blink modes.

COLTYPE=<value> This item specifies that color monitors are being used and selects the type. Valid types are: (1) four memory planes specifying seven colors at two intensity levels, (2) four memory planes giving seven colors and a blink mode for each color, (3) three memory planes specifying a mixture of color, intensity level, and blink modes, and (4) similar to type 2 except that the look-up table must be loaded differently.

XMAX=<value> The maximum X raster value. The default value is 511.

YMAX=<value> The maximum Y raster value. The default value is 511.

CSIZ=<value> The spacing of the characters produced by the hardware character generator. The default value is 7 which is the correct value for a 5 by 7 character generator.

EXTCHR Indicates that the controller can produce the extended character set. This extended character set has 128 characters instead of the default 64.

REPCHR This item indicates that character data written to the screen by the hardware character generator is to replace existing data on the screen instead of simply being added to the screen. REPCHR is for certain special uses only and should not normally be used. The problem is that the use of this item causes the GRINNELL to operate differently than most other graphic devices. For example, if you use this option and draw a label too close to an axis, part of the axis will disappear.

XSIZ=<value> The size of the monitor in centimeters in the horizontal direction. The default value is 25.4

YSIZ=<value> The size of the monitor in centimeters in the vertical direction. The default value is 25.4

If neither BWTYPE nor COLTYPE is given, the default is BWTYPE=1.

It is frequently desirable on these devices to have a means of blocking and buffering the output to the controller. The problem occurs when UGPICT is called to clear the screen or window and then some graphic segments are sent to the device. On a heavily loaded system, the result can be a screen that contains incomplete pictures for appreciable lengths of time. On a device that could be turned on and off, the following code would prevent this problem:

```
      CALL UGPICT('OFF,CLEAR',0)
        ...
      CALL UGWRIT(...)
        ...
      CALL UGWRIT(...)
      CALL UGPICT('ON',0)
```

This code will also prevent the display of partial pictures on this device because the "OFF" option of UGPICT causes the following output operations, including the clear, to be held in an internal buffer until the "ON" signal is given in UGPICT. As long as the internal buffer does not overflow, the entire group of changes will occur at the call to UGPICT with the "ON" option.

SECTION 3.5: THE IBM 3179 G COLOR GRAPHICS DISPLAY STATION

This is a raster-scan display device and may be used in the interactive mode only. It may be selected on the appropriate computer with the following UGOPEN parameter:

IBM computers IBM3179

The device consists of a keyboard, an audible alarm, and a color television monitor. The keyboard contains 24 program function keys that may be used as a BUTTON control unit and another set of arrow keys that can operate the LOCATOR control unit. The resolution of the screen is 720 pixels horizontally by 384 vertically. The unit can draw points, lines, filled polygons, and horizontal hardware-generated characters whose size is equivalent to:

DSIZE=0.01694

Graphic primitives may be of any color.

Since this is a raster-scan graphic device, individual graphic items may be erased by drawing over them with the ERASE option of subroutine UGWRIT. The WINDOW option of subroutine UGPICT may be used to erase rectangular areas. The KEYBOARD input string may be initialized to any arbitrary characters. It will be read when the ENTER key is hit. When this happens, the characters from the start of the string to the position just before the alphanumeric cursor will be obtained. If the cursor is moved out of the input buffer and the ENTER key is hit, then the program will obtain either a null string or the full string depending on whether the cursor was positioned before or after the input buffer. The position of the LOCATOR is signaled by hitting either the ENTER key or any of the program function keys while the graphic cursor is on the screen.

There are no additional parameters that will be recognized by UGOPEN.

There are a number of problems that can arise when using this terminal. Some of the biggest problems occur when something

other than the Unified Graphics System tries to write to the screen while a graphic application program is running. Examples of this are messages from the operator and FORTRAN writes from the application program. When this happens, the picture on the screen will be erased and lost. Depending on what happened, the terminal operator may continue executing the program by hitting either the CLEAR or ENTER key. After this recovery, the application program will be in the same state that it would have been if the interruption had not occurred except that the picture will be gone. Because of this problem, the application program should assure that it is always possible for the terminal operator to regenerate the important aspects of the picture. Messages from other users will not normally get through but will be queued until the application program terminates.

SECTION 3.6: THE IBM 5080 GRAPHICS SYSTEM

This is a refresh display device and may be used in the interactive mode only. It may be selected on the appropriate computer with the following UGOPEN parameter:

IBM computers IBM5080

The basic IBM 5080 Graphics System consists of a color monitor with a resolution of 1024 by 1024, an audible alarm, and a keyboard. Optional control units include a tablet with a drawing cursor, a set of dials, and a panel with 32 buttons. These units can display the two-dimensional primitives of points, lines, four sizes of hardware-generated characters, and filled polygons in all of the colors and intensity levels supported by the Unified Graphics System. All graphic primitives except the polygon-fill can be put into the blink mode. It can also display the three-dimensional primitives of point, line, and character. The three-dimensional viewing parameters can be modified by manipulating the control unit with the dials. The current viewing parameters can be read by subroutine UG3TRN. The character sizes are equivalent to:

DSIZE=0.00977 (very small size)
DSIZE=0.01172 (small size)
DSIZE=0.01758 (large size)
DSIZE=0.02051 (very large size)

Characters can be drawn at an angle of zero or 90 degrees.

This device supports the KEYBOARD, PICK, and BUTTON control units. The PICK operation is controlled by the tablet. When the drawing cursor is within the active area of the tablet, a small cross appears on the screen. When the cursor is used to position the cross near a graphic segment in the PICK mode, then that part of the segment with a common PICKID will brighten on the screen. When button number one on the cursor is pushed, the event is sent to the host computer. The lights under the 32 buttons can be turned on and off with the BUTTON option of subroutine UGECTL.

To make the IBM 5080 available to your program, it must first be attached to your virtual machine, usually at address 201. You must also issue a FILEDEF command such as:

```
FILEDEF UGDEVICE GRAF 201
```

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the "file" in the FILEDEF command. The default value is UGDEVICE on the IBM computers.

EXTKEYS This item specifies that the 24 pfkeys on the keyboard are to be reported as buttons 33 through 56. The default value is for these pfkeys to be reported as buttons 1 through 24. If the graphics system does not have the panel with 32 buttons, then this item is ignored and the default applies.

BWONLY This item specifies that color is to be ignored and the picture is to consist of black and white display items only. This can be useful to find out how a program might run on a simpler graphic device.

VIEWCTL=<value> The default for this item is NONE which means that the graphic device is to act as a two-dimensional device with no three-dimensional viewing control. To get three-dimensional viewing control the value should be 3DDIAL. This value causes the dial control unit to act as a three-dimensional control unit. In that case, dials 1 through 3 perform rotations, dial 4 does a zoom, dials 5 through 7 do translations, dial 8 modifies the position of the near and far clipping plane, button 3 on the tablet cursor acts as a toggle for a point or parallel projection, and button 4 on the cursor resets the view to its original value.

DSHADE=<value> This item indicates the number of intensity levels to be used to try to indicate depth in three-dimensional primitives. Its value may be 1, 2, 3, or 4 with 1 being the default. When the higher values are used, some of the intensity levels defined in the program will not be honored. The higher values also significantly increase regeneration time.

There are a number of problems associated with this device. The polygon-fill primitive should be used with care because it can severely slow down regeneration time. It should not be used when three-dimensional manipulation is to take place. Also the polygon-fill primitives do not blink. Three-dimensional character strings cannot be positioned as described in this document. The RIGHT and CENTER options are always ignored and the position with LEFT is only approximate.

There are a number of additional things that can be done with this device through the Unified Graphics System. Some of these are functions that are similar to the normal Unified Graphics System operations but work differently on this device. There are

also additional UGOPEN options that are very device-dependent and make it very difficult to switch a program from this device to another device. These additional facilities will not be described here.

SECTION 3.7: THE IMAGEN LASER PLOTTERS

The Unified Graphics System supports the Model 8/300 Printer/Plotter and a number of other printer/plotters made by the IMAGEN Corporation. These graphic devices may be used in the non-interactive mode only. They may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... IMGN300, IMGNIBM
IBM computers IMGN300

The IMGN300 option generates files that may be sent to an IMAGEN connected to the type of computer that generated the file. IMGNIBM may be used on the VAX computer to produce files that must be sent to the IBM computer for plotting.

The devices consist of a Canon copier coupled to a microprocessor. The plotter can produce pictures on many sizes of paper depending on the type of paper cassette installed in the unit. The normal paper size is "letter" size or 8.5 by 11.0 inches and this is the usual size available at SLAC although the "legal" size or 8.5 by 14.0 inches is available on some units. Raster units are spaced 300 to the inch on the Model 8/300 and most of the other models. The long dimension of the paper is usually the horizontal direction of the pictures. The intensity level options can be fully supported on this graphic device; higher intensity lines may be drawn thicker than lower intensity lines. The character generator within the laser plotter is not used by the Unified Graphics System. A limited polygon-fill is available on these devices; normally the interior of filled polygons is cleared and then the border is drawn. The laser plotters 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 Canon copier.

On the VAX computer, a file may be sent to a printer/plotter on the IBM computer by a command similar to:

```
$ IBMSERVE <file-name> <unit-identification> /CLASS=G
```

If the Kellerman & Smith support software is installed on the VAX, a file may be sent to a printer/plotter on the VAX by a command similar to:

```
$ IMPRINT/TRANSPARENT <file-name>
```

If the type of the file is .IMP, then a command similar to:

```
$ PRINT <name>.IMP
```

may be used. If the full Kellerman & Smith software is not available on the VAX, other means of sending the file to the IMAGEN may have to be used.

On the IBM computer, A FILEDEF command must be issued for the output file before your program is executed. A typical command is:

```
FILEDEF UGDEVICE DISK fn ft fm
```

The default DCB parameters are RECFM=VB, LRECL=137, and BLKSIZE=2048. The LRECL and BLKSIZE parameters may be changed to any reasonable value, and the RECFM value may be V or VB. On the IBM computer, the command to send the output file to the printer/plotter is:

```
IMPSEND fn ft fm ( <unit-identification>
```

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

PAPER=<value> This item is used to select the paper size for which the pictures are to be produced. The currently supported sizes are LETTER (8.5 by 11.0 inches) or LEGAL (8.5 by 14.0 inches). The default value is LETTER. The laser plotter will signal an error if there is a mismatch between the assumed paper size and the cassette mounted when the pictures are produced.

GENIL This item indicates that the intensity level data is to be simulated. Normally, all lines are composed of a single string of dots. When GENIL is used, a MEDIUM intensity line, for example, is three dots wide and other intensity levels are similarly generated. The use of GENIL on this device does not significantly increase the size of the data set being created; it may slow the plotting speed on the laser plotter but not by very much.

ROTAXIS The normal orientation of a picture on a sheet of paper has the X axis as the longest axis. This option will rotate the picture so that the Y axis is the longest axis.

SLDFILL This item changes the normal action for polygon-fill. When this item is used, all polygon-fill primitives will be solid black. The default action is best for most applications.

BIGMAR The normal plotting area used by the Unified Graphics System leaves one-half inch margins on all four sides of the paper. When this option is used, the margins are one and one-fourth inches.

NOMAR This is a special item that was put in to aid in the merging of text and graphics on this device. It causes the maximum drawing space to have an origin of (0,0) while retaining its normal size. This item should not normally be used.

It is very important to understand that the files prepared for an IMAGEN connected to a VAX are different from the files prepared for an IMAGEN connected to the IBM computer. Do not get them

mixed up.

SECTION 3.8: THE METHEUS OMEGA 300 DISPLAY CONTROLLER

This graphic device is a raster-scan device which is used in the slave-display mode. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDBMETH

This device can support CRT's with a resolution of 640 pixels horizontally by 480 pixels vertically, 736 by 552, 1024 by 768, or 1024 by 1024 pixels. The units can display points, lines, filled polygons, and hardware-generated characters of a number of sizes. When the CRT displays 1024 pixels horizontally, the sizes of the hardware-generated characters supported by the Unified Graphics System are equivalent to:

DSIZE=0.00781 (very small size)
DSIZE=0.01563 (small size)
DSIZE=0.02344 (large size)
DSIZE=0.03125 (very large size)

For units with lesser resolution, the DSIZE values will be proportionally larger. The hardware-generated characters may be drawn at angles of 0, 90, 180, or 270 degrees. Graphic primitives may be of any color, and any primitive may be put into the blink mode. The Unified Graphics System only supports a single intensity level.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

CHANNEL=<value> The identification of the I/O controller. The default value is XAA0.
XMAX=<value> The maximum X pixel value. The default value is 1023.
YMAX=<value> The maximum Y pixel value. The default value is 1023.
XSIZ=<value> The size of the monitor in centimeters in the horizontal direction. The default value is 25.4
YSIZ=<value> The size of the monitor in centimeters in the vertical direction. The default value is 25.4

It is frequently desirable on these devices to have a means of blocking and buffering the output to the controller. The problem occurs when UGPICT is called to clear the screen or window and then some graphic segments are sent to the device. On a heavily loaded system, the result can be a screen that contains incomplete pictures for appreciable lengths of time. On a device that could be turned on and off, the following code would prevent this problem:

```
      CALL UGPICT('OFF,CLEAR',0)
        ...
      CALL UGWRIT(...)
        ...
      CALL UGWRIT(...)
      CALL UGPICT('ON',0)
```

This code will also prevent the display of partial pictures on this device because the "OFF" option of UGPICT causes the following output operations, including the clear, to be held in an internal buffer until the "ON" signal is given in UGPICT. As long as the internal buffer does not overflow, the entire group of changes will occur at the call to UGPICT with the "ON" option.

For this device, subroutine UGDDAT may be used to send device-dependent data to the display. The character string supplied to that subroutine in the DATA argument must have "SDDMETH:" as its first eight characters. These characters will be stripped off, and the remaining characters will be sent to the device without modification. When the data is sent to the device, the unit will be in the state specified by the OPTIONS argument. In addition, the P1 register of the device will be positioned at (XCOORD,YCOORD). These coordinates must be within the current window; if they are not, the scissoring module will eliminate the item.

SECTION 3.9: THE POSTSCRIPT LANGUAGE

This section describes device-dependent code that produces files that contain picture descriptions in the PostScript Language. These files can then be transmitted to any device that supports the PostScript Language. A number of UGOPEN options items are provided so that a user can tailor the output of this device-dependent code to match non-standard device and/or operating system requirements. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... POSTSCR
IBM computers ..... POSTSCR

The resolution depends on the actual device doing the plotting. The device-dependent code can draw monochrome or color points, lines, filled polygons, and hardware-generated characters of any size. The hardware-generated characters may be drawn at any angle. The intensity level options can be used to cause lines to be drawn in various widths.

When this device is used on the IBM computer, a FILEDEF command must be issued for the output file before the program is executed. A typical command is:

FILEDEF UGDEVICE DISK fn ft fm

The default DCB parameters are RECFM=VB, LRECL=136, and BLKSIZE=2048. The LRECL and BLKSIZE parameters may be changed to

any reasonable value and the RECFM parameter may be changed to V.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

XMIN=<value> This item specifies the smallest addressable X value to be used. The X direction is always represented by the long dimension of an 8.5 by 11 inch piece of paper. The default value is 150.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 3150.

YMIN=<value> This item specifies the smallest addressable Y value to be used. The default value is 150.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 2400.

RUCMX=<value> This item specifies the number of raster units per centimeter in the X direction. The default value is 118.11024.

RUCMY=<value> This item specifies the number of raster units per centimeter in the y direction. The default value is 118.11024.

GENIL This item indicates that the intensity level data is to be simulated by drawing thicker lines for the higher intensity levels. The use of GENIL on this device does not significantly increase the size of the data set being created; it may slow the speed on the plotter but usually not by very much.

GENCR This item indicates that a file is to be prepared for a color PostScript device. The default is to prepare a file for a monochrome device.

ROTAXIS The normal orientation of a picture on a sheet of paper has the X axis as the longest axis. This option will rotate the picture so that the Y axis is the longest axis.

SLDFILL This item changes the normal action for polygon-fill. When this item is used, all polygon-fill primitives will be drawn in solid color instead of the default action of outlining the polygon. The default action is best for monochrome applications but this option is very useful with color.

ASCII The graphic records are generated in the character encoding on the host computer. This item translates all characters in all records to ASCII. It has no effect on the VAX computer but forces an EBCDIC to ASCII translation on the IBM computer.

BEGFIL=<value> This item defines an initialization string of hexadecimal characters. It is put at the very beginning of the file. The default value is a null string.

ENDFIL=<value> This item defines a termination string of hexadecimal characters. It is put at the very end of the file. The default value is a null string.

BEGPIC=<value> This item defines a string of hexadecimal characters that is put at the beginning of each picture. The default value is a null string.

ENDPIC=<value> This item defines a string of hexadecimal characters that is put at the end of each picture. The default value is a null string.

BEGREC=<value> This item defines a string of hexadecimal characters that is put at the beginning of every record. The default value is a null string.

ENDREC=<value> This item defines a string of hexadecimal characters that is put at the end of every record. The default value is a null string.

The defaults for the XMIN,...,RUCMY options are correct for 8.5 by 11 inch letter size paper at 300 dots per inch and half inch margins on all sides. A PostScript file is really independent of these values; however if they are set to match the actual device, full advantage will be taken of the device and no extra information will be sent to the device. It is very important that the user be aware of the properties of the PostScript printer to be used. The defaults are generally right for a monochrome LaserWriter but the options XMIN=392 and XMAX=3088 are necessary for a QMS ColorScript 100, Model 10p printer because it cannot use the full 11 inch length on the page; if these options are not given, part of the picture may be lost.

The options items described as hexadecimal character strings must consist of an even number of the hexadecimal digits: 0,...,9,A,...,F. The hexadecimal characters are given in the code of the host computer: ASCII on the VAX, and EBCDIC on the IBM. The maximum length of these strings is 32 characters or 64 hexadecimal digits. If the given string does not follow these restrictions, it is set to the null string. The defaults for the BEGPGM,...,ENDREC options cause no extra non-PostScript data to be added to the display file.

If a monochrome file is sent to a color device, everything usually works correctly. If a color file is sent to a monochrome device, strange things, including the disappearance of lines can occur.

There is a problem with the length of the files that are produced; they tend to be very long with respect to other similar devices.

SECTION 3.10: THE PRINTRONIX (MODEL MVP) PLOTTER

This graphic device may be used in the non-interactive mode only. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... PRNTRNX

The PRINTRONIX, Model MVP, Printer/Plotter generates pictures when it is supplied with data in raster form. 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 Unified Graphics System generates pictures for the Model MVP when it is in the Medium Resolution Mode (Mode 002) and is using 14 by 11 inch paper. The hardware character generator on the unit can only be used in the print mode and is not used by the Unified Graphics System. The resolution is 792 raster units horizontally by 720 raster units vertically.

On the VAX computers, the command to send the output file to the printer/plotter is:

```
$ PRINT/NOFEED <file-name>
```

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers.

ROTAXIS The normal orientation of a picture on a fan-fold sheet of paper has the X axis as the longest axis. This option will rotate the picture so that the Y axis is the longest axis.

MAXNSA=<value> MAXNSA gives the size, in words, of the blocks of memory where the line segments that constitute the picture will be accumulated. The default value is 1024.

The default value for MAXNSA should not normally be changed.

The user should remember that the scan lines are saved on disk in an uncompressed form. As a result, each picture will require nearly 100,000 bytes of data.

SECTION 3.11: THE SEIKO GR-1105 COLOR GRAPHICS TERMINAL

This graphic device can be used as either a raster-scan or refresh device. It is potentially usable as a non-interactive, slave-display or interactive device. At present, the Unified Graphics System supports it as a raster-scan or refresh device in the slave-display or interactive modes. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDDSKRS, SDDSKRF, SEIKORS, SEIKORF

The device consists of a keyboard, an audible alarm, and a color television monitor. The keyboard contains 16 program function keys that may be used as a BUTTON control unit in the interactive modes, and another set of arrow keys that can operate the LOCATOR control unit, also in the interactive modes. The resolution of the screen is 1024 pixels horizontally by 780 vertically. The

unit can draw points, lines, filled polygons, and hardware-generated characters of a number of sizes. The sizes supported by the Unified Graphics System are equivalent to:

DSIZE=0.01524 (small size)
DSIZE=0.03048 (large size)

The hardware-generated characters may be drawn at angles of 0, 90, 180, or 270 degrees. Graphic primitives may be of any color but only a single intensity level is supported.

When used as a raster-scan graphic device, individual graphic items may be erased by drawing over them with the ERASE option of subroutine UGWRIT. The WINDOW option of subroutine UGPICT may be used to erase rectangular areas.

When used as a refresh graphic display, individual graphic segments may be identified and individually deleted. In addition, individual graphic segments may take part in the OMIT/INCLUDE operations of subroutine UGPICT.

In the slave-display modes, the character display is really under the control of the user. The only thing the Unified Graphics System does is to clear it when the graphic picture is cleared.

In the interactive modes, the Unified Graphics System will control the character plane, and the KEYBOARD input string may be initialized to any arbitrary characters. It will be read when the RETURN key is hit. The position of the LOCATOR is signaled by hitting either the RETURN key or any of the normal character keys while the graphic cursor is on the screen.

When this device is used as a re-fresh device in the interactive mode, an additional facility is available. If the console operator types "$UGS" or "$ugs", and only these characters, followed by the RETURN key, the display will be cleared and regenerated from local memory. This can be useful if the deleting or omitting of other graphics segments has partially erased some of the remaining segments. The KEYBOARD control unit does not have to be enabled for this to work.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default value is TT.

BASIC, VT100 These items are used to indicate if the terminal is a basic GR-1105 or a unit that emulates a VT100. The default value is VT100.

NOCLTXT This item causes the Unified Graphics System to refrain from clearing the text display when running in the slave-display modes. The text display will be cleared when the device is opened, but after that, it is completely under the control of the user.

When the BUTTON control unit is to be used, the Unified Graphics System has to change the character strings that are loaded into the function key registers. These registers are only changed when the BUTTON control unit is enabled. The original contents of the registers are restored when the BUTTON unit is disabled or when the device is closed.

There are a few difficulties associated with using this device as a refresh display device. In the first place, the unit must have enough segment memory to hold the picture. A second, and more disconcerting, problem is that nothing appears on the screen while data is being written to segment memory. When the entire graphic segment has been transmitted, it is then quickly drawn on the screen. This means that a user can spend substantial periods of time staring at an apparently dead terminal.

SECTION 3.12: THE SLAC EXPERIMENTAL SLAVE SCOPE

This is a refresh display device and may be used in the slave-display mode. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDDSXSS

This device was designed and constructed at SLAC and is usually attached to its host computer through a CAMAC controller. The CRT has a resolution of 1024 rasters horizontally and 1024 vertically. These units can display points, lines, and four sizes of hardware-generated characters at a single intensity level. Any of the displayable items may be put into the blink mode. Graphic segments may be fully manipulated on this device; individual segments may be put into the include or omit state, and individual segments may be deleted. The hardware is capable of supporting a color CRT although no color CRT has ever been used on any of these units. The character sizes are equivalent to:

DSIZE=0.00781 (very small size)
DSIZE=0.01563 (small size)
DSIZE=0.02344 (large size)
DSIZE=0.03125 (very large size)

The controller contains a 1024 word buffer which is used to hold the current picture. A program using this device will write display orders to the display buffer. The picture is regenerated from these display orders 60 times a second. This limit of 1024 words can represent a serious problem when writing programs for this device; the programmer must always make a strong effort to keep the pictures short and simple. The Unified Graphics subroutines transmit their data to the graphic devices by calling subroutines CAMDIO and CAMIOP. It is the responsibility of the user to assure that the correct versions of these subroutines are incorporated into the executable module.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

JORWAY This item indicates that the display scopes are connected through a Jorway CAMAC controller. The default is to assume that the SLAC designed CAMAC controller is being used.

UNIT=<value> This gives the address bits required to access a display unit through the CAMAC controller. The value must be given as a string of bits (zeros and ones). For the SLAC designed CAMAC controller, the form of the value is BBBBBBBBCCCMMMMM and for the Jorway controller the form is BBBBCCCCCCCMMMMM where the B's represent the low order bits of the branch address, the C's give the crate address, and the M's give the module address. No default value is supplied for this parameter; it must be given when UGOPEN is called.

GENLS This item indicates that line structure is to be simulated. Normally, all lines are solid on this device. GENLS should be used sparingly; much more data must be put into the display controller when line structure is generated.

GENCR If a color CRT ever becomes available, this item will instruct the Unified Graphics System to generate data for that device.

SECTION 3.13: THE TALARIS LASER PLOTTERS

The Unified Graphics System supports the Talaris Printer/Plotters made by Talaris Systems through their EXCL language. These graphic devices may be used in the non-interactive mode only. They may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... TALARIS

The plotter can produce pictures on a number of different sizes of paper. The normal paper size is "letter" size or 8.5 by 11.0 inches. Raster units are spaced 300 to the inch. The long dimension of the paper is usually the horizontal direction of the pictures. The intensity level options can be fully supported on this graphic device; higher intensity lines may be drawn thicker than lower intensity lines. The character generator within the laser plotter is not used by the Unified Graphics System. A limited polygon-fill is available on these devices; normally the interior of filled polygons is cleared and the border is drawn.

On the VAX computers, the command to send the output file to the printer/plotter is:

```
$ PRINT <file-name>
```

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers.

GENIL This item indicates that the intensity level data is to be simulated. Normally, all lines are composed of a string of dots two rasters wide. When GENIL is used, a MEDIUM intensity line, for example, is four rasters wide and other intensity levels are similarly generated. The use of GENIL on this device does not significantly increase the size of the data set being created; it may slow the plotting speed on the laser plotter but not by very much.

GENLS This item indicates that the line structure is to be generated by software in the Unified Graphics System. Normally, the line structure is produced by the hardware. Much less data must be sent to the device when the line structure is generated by hardware. However, the software generator usually does a much better job, especially if a curve is being defined by many short vectors.

ROTAXIS The normal orientation of a picture on a sheet of paper has the X axis as the longest axis. This option will rotate the picture so that the Y axis is the longest axis.

SLDFILL This item changes the normal action for polygon-fill. When this item is used, all polygon-fill primitives will be solid black. The default action is best for most applications.

XMIN=<value> This item specifies the smallest addressable X value, that is the value along the long dimension the paper, to be used. The default value is 150.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 3150.

YMIN=<value> This item specifies the smallest addressable Y value, that is the value along the short dimension of the paper, to be used. The default value is 150.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 2400.

XSIZE=<value> This item specifies the size of the paper in the long dimension. The default value is 3300.

YSIZE=<value> This item specifies the size of the paper in the short dimension. The default value is 2550.

PTRAY=<value> This item selects which tray the paper is to be obtained from. The default value is 2, which selects the top tray.

PSIZE=<value> This item specifies the paper size. The default value is 3, which selects letter size paper.

For this device, subroutine UGDDAT may be used to send device-dependent data to the plotter. The character string supplied to that subroutine in the DATA argument must have "TALARIS:" as its first eight characters. These characters will be stripped off,

and the remaining characters will be sent to the device without modification. When the data is sent to the device, the unit will be in the state specified by the OPTIONS argument. In addition, the graphics cursor will be positioned at (XCOORD,YCOORD). These coordinates must be within the current window; if they are not, the scissoring module will eliminate the item. The data string supplied by subroutine UGDDAT cannot be split between records. If it cannot fit, it will be ignored. This problem can be somewhat severe because the record length is only 133 bytes.

The options items XMIN, XMAX, YMIN, YMAX, XSIZE, YSIZE, and PSIZE all have default values which are correct for letter size paper with one-half inch borders. If other size paper is to be used, most of these values will have to be changed. The values of XMIN, XMAX, YMIN, and YMAX are independent of ROTAXIS; they do not have to be changed when ROTAXIS is specified. The options items PTRAY and PSIZE are used as arguments in the TALFCTL command of the EXCL language. Their values must be appropriate for that command.

SECTION 3.14: THE TEKTRONIX 4010 SERIES TERMINALS

This graphic device contains a storage display screen and may be used in the non-interactive, slave-display, or interactive modes. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers SEQ4010, SDD4010, TEK4010
IBM computers SEQ4010, SDD4010, TEK4010

On the IBM computers, this device is supported as a line mode terminal.

The TEKTRONIX 4010 series terminals have a display area that is about eight inches wide by six inches high, a keyboard, an audible alarm, and some have a pair of thumb wheels that can be used as a LOCATOR control unit. The CRT has a resolution of 1024 rasters horizontally and 780 vertically. These units can display points, lines, and a single size of hardware-generated characters at a single intensity level. The character size is equivalent to:

DSIZE=0.01797

When this device is used in the non-interactive mode, a job is run which generates a file containing the graphic orders to draw the pictures. Each picture consists of: (1) a short title, (2) a clear screen order, (3) null characters to permit the clear screen order to take effect, (4) the graphic orders themselves, and (5) commands to sound the audible alarm. The pictures are separated by IEBUPDTE control cards; that is, a card:

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

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

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

and the last card in the file is:

```
./        ENDUP
```

A program is supplied by the Unified Graphics System to allow a user to display the pictures on a terminal. On the VAX computers the program is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ RUN UGSYSTEM:DPIC4010
```

On the IBM computers, the program is invoked by the commands:

```
GIME UGS77
DPIC4010
```

In the interactive mode, this graphic device supports the KEYBOARD and LOCATOR control units. When the LOCATOR is being read, a cross-hair appears on the screen. The console operator can move this indicator around with the thumb wheels on the terminal. When the indicator is properly positioned, the console operator should strike one of the keys for a normal printing character to send the screen coordinates to the host computer. There is, however, a problem when using this locator control. When the terminal is in one mode, the event is signaled by striking the key as described above. In another mode, the event is signaled by doing the above and then striking the carriage-return key. This mode is controlled by something TEKTRONIX calls "strappable options". Strappable options are controlled by switches and jumper connectors on some of the circuit boards. Most TEKTRONIX emulators also have ways to select this mode.

When this device is used in the non-interactive mode on the IBM computers, a FILEDEF command must be issued for the output file before your program is executed. A typical command is:

```
FILEDEF UGDEVICE DISK fn ft fm
```

The default DCB parameters are RECFM=FB, LRECL=80, and BLKSIZE=800. The BLKSIZE parameter may be changed to any reasonable value and the RECFM parameter may be changed to F. For slave-display or interactive use of this terminal under VM, it is vital that the "line size" of the terminal be set to its maximum value of 255. This may be done with the command:

```
CP TERMINAL LINESIZE 255
```

If this is not done, VM will break the transmitted records up into smaller records by inserting carriage-returns into the record. The result will be missing line segments and strange character strings on the screen.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file for non-interactive use of this device. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

PICTID=<value> This item gives the first four characters for the picture name in the non-interactive mode. The default value is PICT.

PICTSQ=<value> This item supplies a numeric value which is

incremented for each picture. It is converted to three digits and concatenated to the end of the PICTID value to form the complete picture name. The default value is 1.

BAUDRATE=<value> This item is only utilized when using this device in the non-interactive mode. It controls the number of null records following the clear screen order. The default value is 2400.

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default is TT. This parameter is not used on the IBM computers.

SECTION 3.15: TEKTRONIX 4010/4014 EMULATORS

Many graphic device claim to emulate a TEKTRONIX 4010 or TEKTRONIX 4014. That statement is always false. Emulators often neglect to support all of the TEKTRONIX functions, usually have non-TEKTRONIX extensions, and almost always have modes which are not present on a real TEKTRONIX 4010/4014. Indeed, it must have multiple modes to be usable on the IBM computer because a real TEKTRONIX 4010/4014 cannot be used in the full screen mode. The device-dependent code described in this section is an attempt to support a large number of these devices. The purpose of the large number of UGOPEN options items associated with this device-dependent code is to make it possible for a user to utilize one of these emulators without having to constantly modify the setup controls, perform special actions to switch modes, or toggle the text and graphics screen on or off. Unfortunately, the wide variation in TEKTRONIX 4010/4014 emulators makes this code difficult to use and requires that the user have very detailed information about the device they are working with. The graphic device is supported as a storage-display device and can be used in the non-interactive, slave-display, or interactive mode. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SEQTKEM, SDDTKEM, TEKEMUL
IBM computers ..... SEQTKEM, SDDTKEM, TEKEMUL

On the IBM computers, this device is supported as a full-screen terminal.

The device-dependent code described here only uses the normal writing mode of the TEKTRONIX 4010/4014; it does not use the Defocused or Write-Thru modes of the TEKTRONIX 4014. It also does not support any of the many extensions to the basic TEKTRONIX 4010/4014. The resolution of a real TEKTRONIX 4010 is 1024 raster units horizontally and 780 vertically. On a TEKTRONIX 4014, it is four times as big or 4096 by 3120. Emulators usually require that they be addressed at one of these resolutions although the real resolution on the CRT is often

less. The four hardware line structures and the four hardware character sizes of the TEKTRONIX 4014 are supported. The default character sizes supported by the Unified Graphics System are equivalent to:

```
DSIZE=0.01795    (normal size)
DSIZE=0.01635    (small size)
DSIZE=0.01090    (smaller size)
DSIZE=0.00994    (smallest size)
```

Alternate sizes may be specified by the user. In the interactive mode, the KEYBOARD and LOCATOR control units are supported. In addition, the BUTTON control unit is supported on the IBM computer. The BUTTON control unit cannot be supported on the VAX because there is no common interface to that control unit.

When this device is used in the non-interactive mode, a job is run which generates a file containing the graphic orders to draw the pictures. Each picture usually consists of: (1) a short title, (2) orders to clear the graphics screen, (3) the graphic orders themselves, and (4) commands to sound the audible alarm. The pictures are normally separated by IEBUPDTE control cards; that is, a card:

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

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

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

and the last card in the file is:

```
./        ENDUP
```

A program is supplied by the Unified Graphics System to allow a user to display the pictures on a terminal. On the VAX computers the program is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ RUN UGSYSTEM:DPICTKEM
```

On the IBM computers, the program is invoked by the commands:

```
GIME UGS77
DPICTKEM
```

When this device is used on the IBM computer in the non-interactive mode, a FILEDEF command must be issued for the output file before the program is executed. A typical command is:

```
FILEDEF UGDEVICE DISK fn ft fm
```

The default DCB parameters are RECFM=FB, LRECL=80, and BLKSIZE=800. The BLKSIZE parameter may be changed to any reasonable value and the RECFM parameter may be changed to F.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

XMIN=<value> This item specifies the smallest addressable X value to be used. The default value is 0.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 4095.

YMIN=<value> This item specifies the smallest addressable Y value to be used. The default value is 0.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 3119.

XSIZ=<value> This item specifies the width of the display screen in centimeters. This value serves two functions; first, it lets subroutine UGINFO return the correct values for the drawing space and window size, and second, if line structure must be generated by software, it allows the Unified Graphics System to generate the proper spacing of the dots and dashes. The default value is 24.0.

YSIZ=<value> This item specifies the height of the display screen in centimeters. The default value is 18.0.

LORES This item causes the Unified Graphics System to produce pictures with the lower resolution of the TEKTRONIX 4010 instead of the higher resolution of the TEKTRONIX 4014. In the TEKTRONIX 4010 resolution a maximum of 4 bytes are transmitted for each line end point, while a maximum of 5 bytes are transmitted when using TEKTRONIX 4014 resolution.

NOOPT This item causes the Unified Graphics System to suppress the optimization of the X-Y coordinates. Normally, only the necessary bytes are transmitted to the device. When this option is used, all bytes are transmitted. Some devices may require that this option be used, especially at higher baud rates.

GENLS This item causes the Unified Graphics System to generate the line structure with software. If this options item is not used, the graphic device must respond to the line structure orders of the TEKTRONIX 4014. If this options item is given, no line structure orders will be sent to the graphic device.

CSIZ1=<value> This item defines the size of the largest hardware generated character on the device. The default value is 56.

CSIZ2=<value> This item defines the size of the second largest hardware generated character on the device. The default value is 51.

CSIZ3=<value> This item defines the size of the third largest hardware generated character on the device. The default value is 34.

CSIZ4=<value> This item defines the size of the smallest hardware generated character on the device. The default value is 31.

NCSIZ=<value> This item gives the number of hardware generated character sizes available on the graphic device. If this value is bigger than 1, the device must respond to the character size selection orders of the TEKTRONIX 4014. If a value of 1 is used, a necessity for a device that emulates a TEKTRONIX 4010, no orders to select character size will be sent to the device. The default value is 4.

BEGPGM=<value> This item defines an initialization string of hexadecimal characters. In the non-interactive mode, a non-null string is put in a special "picture" at the beginning of the file. In the slave-display and interactive modes, it is transmitted when UGOPEN is called. The default value is a null string.

ENDPGM=<value> This item defines a termination string of hexadecimal characters. In the non-interactive mode, a non-null string is put in a special "picture" at the very end of the file. In the slave-display and interactive modes, it is transmitted when UGCLOS is called. The default value is a null string.

BEGREC=<value> This item defines a string of hexadecimal characters that is put at the beginning of every record of graphic data sent to the device. The default value is a null string.

ENDREC=<value> This item defines a string of hexadecimal characters that is put at the end of every record of graphic data sent to the device. The default value is a null string.

CLEAR=<value> This item defines a string of hexadecimal characters that is used to clear the graphics screen. The default value is 1D1B0C1F.

IDLE=<value> This item defines a single hexadecimal character that is to be replicated and follow the CLEAR string. Its purpose is to delay the start of graphic data until the screen has had time to clear. The default value is 1F.

NIDLE=<value> The number of IDLE characters to follow the CLEAR string. The value of this item can be dependent on the baud rate being used. The default value is 0.

BELL=<value> This item defines a string of hexadecimal characters that is used to ring the bell on the device. The default value is 1D07070707070707070707070707071F.

DDNAME=<value> This item selects the name of the output file for non-interactive use of this device. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

PICTID=<value> This item gives the first four characters for the picture name in the non-interactive mode. The default value is PICT.

PICTSQ=<value> This item supplies a numeric value which is incremented for each picture. It is converted to three digits and concatenated to the end of the PICTID value to form the complete picture name. The default value is 1.

ENDCLEAR In the non-interactive mode, this item causes a special "picture" to be created that contains only the CLEAR and BELL strings. It is put at the end of all normal pictures. The purpose of the picture is to clear the graphics screen on devices that make this difficult. The default is to not generate this picture.

TEKONLY In the non-interactive mode, this item causes the IEBUPDTE records and the title record in a picture to be suppressed. The default is to generate these records. This options item must not be used if the program DPICTKEM is to be used to view the pictures.

ASCII The graphic records are always generated in ASCII. In the non-interactive mode on the IBM computer, they

are usually then translated to EBCDIC. This options item suppresses that translation and must not be used if the program DPICTKEM is to be used to view the pictures.

NSEQ=<value> In the non-interactive mode, this item specifies the number of characters in the "sequence number field" of the record. The default value is 8.

PADBER=<value> This item defines a single hexadecimal character that is used as padding before the ENDREC string in the non-interactive mode. The effect of using this item is to cause the ENDREC string to be shifted to the end of the usable part of the record. The default value is a null string.

PADAER=<value> This item defines a single hexadecimal character that is used as padding after the ENDREC string and in the sequence number field in the non-interactive mode. The default value is 20, that is, an ASCII blank.

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default value is TT. On the IBM computer, the default value is UGDEVICE.

BEGLOC=<value> In the interactive mode, this item defines a string of hexadecimal characters that is used to initiate the use of the LOCATOR control unit. The default value is 1D1B1A.

ENDLOC=<value> In the interactive mode, this item defines a string of hexadecimal characters that is used to terminate the use of the LOCATOR control unit. The default value is 1F.

NOLOC In the interactive mode, this item specifies that the locator control unit is not available on the device. If the device does not have a LOCATOR control unit, the use of this options item will allow UGINFO to return correct information about the control units, and the Unified Graphics System will be able to flag any attempt to use the LOCATOR control unit as an error. The default value is to support the LOCATOR control unit with the BEGLOC and ENDLOC strings.

DLINES=<value> This item is used in the interactive mode on the IBM computer to tell the Unified Graphics System how many lines the text display actually has in the non-TEKTRONIX mode. This value should count all of the lines, even if they are not being used in the current mode. The default value is 32.

TLINE=<value> This item is used in the interactive mode on the IBM computer to tell the Unified Graphics System which line of text is at the top of the usable portion of the screen. The default value is 1.

BLINE=<value> This item is used in the interactive mode on the IBM computer to tell the Unified Graphics System which line of text is at the bottom of the usable portion of the screen. The default value is the same as the DLINES value.

DCHARS=<value> This item is used in the interactive mode on the IBM computer to tell the Unified Graphics System how many characters there are in a line of text. The default value is 80.

BUTTON=<value> This item is used in the interactive mode on the IBM computer to tell the Unified Graphics System how many buttons are available in the BUTTON control unit. If the BUTTON control unit is not available on the device, this value should be set to zero. The default value is 24 on the IBM computer.

The defaults for these options items do not match any device in particular. They were chosen to try to minimize the changes that are necessary for a large number of graphic devices. Unfortunately, that means that it will always be necessary to supply at least a few of them.

The values given for XMIN,...,YMAX and for CSIZ1,...,CSIZ4 should be given in the coordinate space of the TEKTRONIX 4014, even if the actual device is really emulating a TEKTRONIX 4010. The use of the LORES options item for a TEKTRONIX 4010 emulator effectively divides all coordinates by four. Actually, the LORES options item is seldom absolutely necessary because the extra byte generated for the TEKTRONIX 4014 resolution is usually ignored by TEKTRONIX 4010 emulators. The use of LORES will, however, always reduce the amount of data that is sent to the graphic device to draw the picture.

The options items described as hexadecimal character strings must consist of an even number of the hexadecimal digits: 0,...,9,A,...,F. The hexadecimal characters are always given as ASCII characters, even on the IBM computer. The maximum length of these strings is 32 characters or 64 hexadecimal digits. Items described as a single hexadecimal character must consist of exactly two hexadecimal digits. If the given string does not follow these restrictions, it is set to the null string.

The purpose of the BEGPGM, ENDPGM, BEGREC, and ENDREC strings is to provide a means of switching between the native mode of the graphic device and the TEKTRONIX mode. Typically, on the VAX computers, it is only necessary to put the device into TEKTRONIX mode at the beginning of a picture or program and get it back into the native mode at the end of a picture or program. Additions to the CLEAR and BELL strings may be used to do this function before and after each picture in the non-interactive mode while the BEGPGM and ENDPGM can be used in the slave-display and interactive modes to perform the switching at the beginning an end of a program. However, on the IBM computer, it is usually necessary to get the graphic device back to its native mode after each record. In this case, BEGREC and ENDREC may be used. The BEGPGM, ENDPGM, CLEAR and IDLE, BELL strings, and the graphic records themselves, are all preceded by BEGREC and followed by ENDREC when they are sent to the graphic device. Each record of graphic data will always begin with a hexadecimal character of 1D and will always end with a 1F. BEGLOC is preceded by BEGREC and ENDLOC is followed by ENDREC. In the slave-display and

interactive modes, the length of the given hexadecimal strings does not pose any problems because the length of the actual records can be quite large. However, in the non-interactive mode, problems can arise. For example, the BEGREC, BELL, and ENDREC strings must all fit into a single record whose default length is only 72 characters. Another problem with the use of the BEGPGM, ENDPGM, BEGREC, and ENDREC strings occurs when a program terminates abnormally. In that case, the final sequences may mot be transmitted and a graphic terminal can be left in the TEKTRONIX 4010/4014 mode while the computer is trying to address it in its native mode. This usually does not cause to much trouble on the VAX computers, but it can be disastrous on the IBM computer.

For this device, subroutine UGDDAT may be used to send device-dependent data to the display. The character string supplied to that subroutine in the DATA argument must have "TEKEMUL:" as its first eight characters. These characters will be stripped off, and the remaining characters will be sent to the device without modification. When the data is sent to the device, the unit will be in the state specified by the OPTIONS argument. In addition, the graphics cursor will be positioned at (XCOORD,YCOORD). These coordinates must be within the current window; if they are not, the scissoring module will eliminate the item. The data string supplied by subroutine UGDDAT cannot be split between records. If it cannot fit, it will be ignored. This does not pose much problem in the slave-display or interactive modes, but the small record size in the non-interactive mode can be troublesome, especially if long BEGREC or ENDREC strings are used.

The following paragraphs will supply a number of examples of UGOPEN options items for various actual graphic devices. The examples shown here only represent one possibility for the options items, other combinations are often possible. The examples for the terminals all try to keep both the text and graphics screens, if they are different, visible at all times. This is the way a real TEKTRONIX 4010/4014 works, and successful use of an emulator usually requires that this be the case. It is also very important to remember that the options items are extremely sensitive to the setup parameters on the device; the examples shown here will all work when the suggested settings at SLAC are used.

A real TEKTRONIX 4010 on the VAX:

```
SEQTKEM,XSIZ=20.0,YSIZ=15.2,LORES,NOOPT,GENLS,NCSIZ=1,NIDLE=240
SDDTKEM,XSIZ=20.0,YSIZ=15.2,LORES,NOOPT,GENLS,NCSIZ=1,NIDLE=240
TEKEMUL,XSIZ=20.0,YSIZ=15.2,LORES,NOOPT,GENLS,NCSIZ=1,NIDLE=240
```

The only item needing extra explanation is the NIDLE value; it is shown for a device connected at 2400 baud. The file produced by SEQTKEM will be functionally equivalent, but not identical, to a similar file produced by SEQ4010. A real TEKTRONIX 4010 cannot be supported on the IBM computer in the full-screen mode.

A Modgraph GX-2000 on the VAX:

```
SEQTKEM,XSIZ=25.5,YSIZ=20.2,LORES,
  CLEAR=1B5E31333661D1B0C1F,BELL=1D071F181B5E31343366
SDDTKEM,XSIZ=25.5,YSIZ=20.2,LORES,ENDPGM=18
TEKEMUL,XSIZ=25.5,YSIZ=20.2,LORES,ENDPGM=18
```

The CLEAR and BELL strings are more complicated in the non-interactive use because they have more to do. CLEAR must clear the text screen on the device in addition to the graphics screen. BELL must also get out of graphics mode and restore the text screen so that any keyboard input is visible on the screen.

A Modgraph GX-2000 on the IBM computer:

```
SEQTKEM,YMIN=475,XSIZ=25.5,YSIZ=20.2,LORES,
  ENDREC=181B5E31343366
SDDTKEM,YMIN=475,XSIZ=25.5,YSIZ=20.2,LORES,
  ENDREC=181B5E31343366
TEKEMUL,XSIZ=25.5,YSIZ=20.2,LORES,DLINES=48,BLINE=43,
  ENDREC=181B5E31343366
```

The YMIN value in the non-interactive and slave-display modes keeps the pictures from overlapping the VM/CMS command line at the bottom of the screen. The ENDREC string takes the terminal out of TEKTRONIX mode and assures that the graphics screen is still visible.

The IMAGEN on the IBM computer: This example prepares a file that may be sent to the IMAGEN, using the IMPSEND command, without modification. It produces the pictures by using the TEKTRONIX emulation mode of the IMAGEN. This is not a practical use of this device dependent code because the IMAGEN is fully supported in its native mode, but is included here to show an unusual application.

```
SEQTKEM,XSIZ=21.5,YSIZ=16.7,LORES,GENLS,
  BEGREC=40,ENDREC=E7,BEGPGM=
  40444F43554D454E54284C414E475541474520544 54B54524F4E495829,
  BELL=X,TEKONLY,ASCII,NSEQ=0,PADBER=1F
```

The BEGREC and ENDREC strings represent characters that will be stripped off by some of the VM/CMS software. The BEGPGM string consists of the ASCII characters @DOCUMENT(LANGUAGE TEKTRONIX) and puts the IMAGEN into TEKTRONIX emulation mode. The BELL value is invalid and sets that string to the null string. The NSEQ and PADBER values assure that the ENDREC string is the final character of each record.

SECTION 3.16: THE TEKTRONIX 4027 COLOR GRAPHICS TERMINAL

This graphic device is a raster-scan graphic device and may be used in the non-interactive, slave-display, or interactive modes. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers SEQ4027, SDD4027, TEK4027

The 4027 color graphics terminal has a keyboard, an audible alarm, and a LOCATOR control unit. The CRT has a resolution of 640 rasters horizontally and 350 vertically when the default DLINES value is used. These units can display points, lines, a single size of hardware-generated characters, and filled polygons at a single intensity level. The character size is equivalent to:

```
DSIZE=0.02292
```

When this device is used in the non-interactive mode, a job is run which generates a file containing the graphic orders to draw the pictures. Each picture consists of: (1) a short title, (2) a clear screen order, (3) the graphic orders themselves, and (4) commands to sound the audible alarm. The pictures are separated by IEBUPDTE control cards; that is, a card:

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

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

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

and the last card in the file is:

```
./        ENDUP
```

A program is supplied by the Unified Graphics System to allow a user to display the pictures on a terminal. On the VAX computers the program is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ RUN UGSYSTEM:DPIC4027
```

This program assumes that the pictures were produced with the default command character.

In the interactive mode, this graphic device supports the KEYBOARD and LOCATOR control units. When the LOCATOR is being read, a cross-hair appears on the screen. The console operator can move this indicator around with the arrow keys. When the indicator is properly positioned, the console operator should strike one of the keys for a normal printing character to send the screen coordinates to the host computer.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file for non-interactive use of this device. The default value is UGDEVICE.DAT on the VAX computers.

PICTID=<value> This item gives the first four characters for the picture name in the non-interactive mode. The default value is PICT.

PICTSQ=<value> This item supplies a numeric value which is incremented for each picture. It is converted to three digits and concatenated to the end of the PICTID value to form the complete picture name. The default value is 1.

GENCH This item indicates that the hardware character generator is to be used when appropriate. The default is to always use the stroke generator for the reasons described below.

DLINES=<value> This item gives the number of lines of text on the face of the CRT to be used for graphics. The default value is 26. The maximum value that is permitted is 33 and the minimum value is 10.

COMCHR=<value> This item allows the command character to be changed to any printable character. The default value is an exclamation mark.

COMBTS=<value> This is an alternative to COMCHR which allows the command character to be set to any 8 bit pattern. The default, on the VAX computers, is 00100001. This bit pattern is the sequence corresponding to the exclamation mark in ASCII.

PROMPT=<value> In the interactive mode, this item gives the prompt character string. The prompt characters will appear below the picture area when the user's program is ready to accept keyboard input. Up to 8 characters may be given. The default value is >.

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default is TT.

There are a number of difficulties associated with this device. First, the resolution can be inadequate for some pictures. Second, a character produced by the hardware character generator cannot be positioned arbitrarily on the screen; it can only be placed in its standard cell when the screen is divided into 34 lines of 80 characters. When a character produced by the hardware character generator occupies a cell, no line segment can be drawn through that cell. Because of these problems, it is actually dangerous to use the hardware character generator with graphics; for example, if you get a label too close to an axis, part of the axis disappears. Characters produced by the hardware character generator are always the same color, usually white. Clearing a window does not erase items drawn by the hardware character generator. Finally, the terminal is very sensitive to characters entered through the keyboard at the wrong time. The reason for this is that a lot of handshaking is going on in graphics mode and any extraneous characters entered at the keyboard may interfere with this handshaking. There is one restriction that applies when this graphic device is used in the non-interactive mode. In that case, the number of sides in a filled polygon is limited to eight.

SECTION 3.17: THE TEKTRONIX 4105 COMPUTER DISPLAY TERMINAL

This graphic device is a raster-scan device that can be used in the non-interactive, slave-display, or interactive mode. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers SEQ4105, SDD4105, TEK4105
IBM computers SEQ4105, SDD4105, TEK4105

On the IBM computers, this device is supported as a full-screen terminal.

The device consists of a keyboard, an audible alarm, and a color television monitor. When used in the interactive mode, the KEYBOARD, BUTTON, and LOCATOR control units are available. When used on the VAX, the keyboard contains 32 program function keys which are activated with the F1 through F8 keys in combination with the SHIFT and/or CONTROL keys. On the IBM computer, the numeric keypad can simulate 24 program function keys. In the interactive mode, another set of arrow keys can operate the LOCATOR control unit. The resolution of the screen is 480 pixels horizontally by 360 vertically although it is addressed internally as if it had 4096 by 3133 pixels. The unit can draw points, lines, filled polygons, and hardware-generated characters of a number of sizes. The sizes supported by the Unified Graphics System are equivalent to:

DSIZE=0.01660 (small size)
DSIZE=0.03320 (large size)

The hardware-generated characters may be drawn at angles of 0, 90, 180, or 270 degrees. Graphic primitives may be of any color but only a single intensity level is supported.

When this device is used in the non-interactive mode, a job is run which generates a file containing the graphic orders to draw the pictures. Each picture consists of: (1) a short title, (2) orders to clear the graphics screen, (3) the graphic orders themselves, and (4) commands to sound the audible alarm. The pictures are separated by IEBUPDTE control cards; that is, a card:

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

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

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

and the last card in the file is:

```
./        ENDUP
```

A program is supplied by the Unified Graphics System to allow a user to display the pictures on a terminal. On the VAX computers the program is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ RUN UGSYSTEM:DPIC4105
```

On the IBM computers, the program is invoked by the commands:

```
GIME UGS77
DPIC4105
```

When used as a slave-display or interactive graphic device, individual graphic items may be erased by drawing over them with the ERASE option of subroutine UGWRIT. The WINDOW option of subroutine UGPICT may be used to erase rectangular areas.

In the slave-display mode, the text display is really under the control of the user. The only thing the Unified Graphics System usually does is to clear it when the graphic picture is cleared.

In the interactive mode, the Unified Graphics System will control the text display, and the KEYBOARD input string may be initialized to any arbitrary characters. It will be read when the RETURN key is hit. The position of the LOCATOR is signaled by hitting any of the normal character keys while the graphic cursor is on the screen. If the optional mouse is used, any of the keys on it may be used to signal the LOCATOR control unit to transmit its position.

When this device is used on the IBM computer in the non-interactive mode, a FILEDEF command must be issued for the output file before the program is executed. A typical command is:

```
FILEDEF UGDEVICE DISK fn ft fm
```

The default DCB parameters are RECFM=FB, LRECL=80, and BLKSIZE=800. The BLKSIZE parameter may be changed to any reasonable value and the RECFM parameter may be changed to F.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

XMIN=<value> This item specifies the smallest addressable X value to be used. The default value is 0.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 4095.

YMIN=<value> This item specifies the smallest addressable Y value to be used. The default value is 0.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 3132.

LORES This item causes the Unified Graphics System to produce pictures with lower resolution. A maximum of four bytes per X-Y coordinate is transmitted instead of the usual maximum of five. The result can be slightly shorter execution and data transmission times.

NOOPT This item causes the Unified Graphics System to suppress the optimization of the X-Y coordinates. Normally, only the necessary bytes are transmitted to the device. When this option is used, all bytes are transmitted. The result can be slightly shorter execution times but longer data transmission times.

DDNAME=<value> This item selects the name of the output file for non-interactive use of this device. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

PICTID=<value> This item gives the first four characters for the picture name in the non-interactive mode. The default value is PICT.

PICTSQ=<value> This item supplies a numeric value which is incremented for each picture. It is converted to three digits and concatenated to the end of the PICTID value to form the complete picture name. The default value is 1.

NOCLTXT This item is only available on the VAX computers in the slave-display mode. It causes the Unified Graphics System to refrain from clearing the text display. The net effect is that the Unified Graphics

System never touches the text display; it is completely under the control of the user.

CHANNEL=<value> This item is used to open a path from the computer to the terminal when it is used in a slave-display or interactive mode. On the VAX computers, the default value is TT. On the IBM computer, the default value is UGDEVICE.

DLINES=<value> This item is used in the interactive mode to tell the Unified Graphics System how many lines the text display actually has. This value should count all of the lines, even if they are not being used in the current mode. The default value is 30.

TLINE=<value> This item is used in the interactive mode to tell the Unified Graphics System which line of text is at the top of the usable portion of the screen. The default value is 1.

BLINE=<value> This item is used in the interactive mode to tell the Unified Graphics System which line of text is at the bottom of the usable portion of the screen. The default value is the same as the DLINES value.

DCHARS=<value> This item is used in the interactive mode to tell the Unified Graphics System how many characters there are in a line of text. The default value is 80.

For this device, subroutine UGDDAT may be used to send device-dependent data to the display. The character string supplied to that subroutine in the DATA argument must have "TEK4105:" as its first eight characters. These characters will be stripped off, and the remaining characters will be sent to the device without modification. When the data is sent to the device, the unit will be in the state specified by the OPTIONS argument. In addition, the graphics cursor will be positioned at (XCOORD,YCOORD). These coordinates must be within the current window; if they are not, the scissoring module will eliminate the item. The data string supplied by subroutine UGDDAT cannot be split between records. If it cannot fit, it will be ignored. This does not pose much problem in the slave-display or interactive modes, but the small record size in the non-interactive mode can be troublesome.

This terminal has many modes that it may be in and this poses some difficult problems for the Unified Graphics System. It seems to work reasonably well when the terminal is initially in the TEK or ANSI mode. It also works, with a few problems, when the terminal is initially in the EDIT or VT52 mode. The Unified Graphics System tries to determine the mode of the terminal before it writes to the screen. When the system is done with the screen, it tries to return the terminal to its original state. In the slave-display mode, the terminal is returned to its original state at the end of each segment; in the interactive mode on the VAX, it is only returned to its original state when UGCLOS is called. If a program terminates in an unusual manner, the mode of the terminal can be unpredictable.

The BUTTON control unit also has the potential for problems. When the BUTTON control unit is enabled on the VAX, a special

string of characters is loaded into volatile memory. These strings are erased when the control unit is disabled or when UGCLOS is called. If the program terminates in an unusual manner, the strings loaded by the Unified Graphics System may still be in effect and could cause trouble.

The DLINES, TLINE, BLINE, and DCHARS options are used by the system to determine how to position the keyboard input buffer on the screen. The defaults are correct for a real TEKTRONIX 4105 in the TEK or ANSI mode, but they may have to be changed in other modes, or when devices that emulate a TEKTRONIX 4105 are used. For example, when the terminal is in the EDIT or VT52 mode TLINE should be set to 7. If a TEKTRONIX 4207 is used in the TEK or ANSI mode, DLINES should be set to 32.

The XMIN, YMIN, XMAX, and YMAX options can be used to restrict the graphics display to a subset of the full screen. In the slave-display mode on the VAX, the user can then use the NOCLTXT option to gain complete control over the text display. The result can be the effective partitioning of the screen into text and graphics areas. Such an application program, may however, be very device-dependent. In the non-interactive or slave-display modes on the IBM computer, the user may want to set YMIN=200 to keep the picture from overlapping the prompts from VM at the bottom of the screen.

The many claims made for other graphic devices or Personal Computer programs about emulating a TEKTRONIX 4105 should not be believed without proof. It will almost always turn out that only a subset of the orders are supported and some of these may not work correctly.

SECTION 3.18: THE TEKTRONIX 4207 COMPUTER DISPLAY TERMINAL

This section describes device-dependent code that can be used to drive the TEKTRONIX 4207 as a refresh device in the slave-display or interactive modes. The device-dependent code for the TEKTRONIX 4105 may also be used to drive this device if it is needed in the non-interactive mode or as a raster-scan device. There are, therefore, five different modules of device-dependent code that may be used with this device. When used in the manner described in this section, it may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers SDD4207, TEK4207
IBM computers SDD4207, TEK4207

On the IBM computers, this device is supported as a full-screen terminal.

The device consists of a keyboard, an audible alarm, and a color television monitor. A mouse and tablet are optional. Extra memory is also optional and it is very strongly recommended if

the device is to be used in the manner described in this section. When used in the interactive mode, the KEYBOARD, PICK, BUTTON, STROKE, and LOCATOR control units are all available. The resolution of the screen is 480 pixels horizontally by 360 vertically although it is addressed internally as if it had 4096 by 3133 pixels. The unit can draw points, lines, filled polygons, and hardware-generated characters of a number of sizes. The sizes supported by the Unified Graphics System are equivalent to:

DSIZE=0.01628 (small size)
DSIZE=0.03257 (large size)

The hardware-generated characters may be drawn at angles of 0, 90, 180, or 270 degrees. Graphic primitives may be of any color. In addition, either all colors can be made to blink or all colors can have a dim intensity level. Panning and zooming may be done locally without the aid of the host computer.

Since this is a refresh graphic display, individual graphic segments may be identified and individually deleted. In addition, individual graphic segments may take part in the OMIT/INCLUDE operations of subroutine UGPICT.

In the slave-display mode, the text display is really under the control of the user. The only thing the Unified Graphics System usually does is to clear it when the graphic picture is cleared.

In the interactive mode, the Unified Graphics System will control the text display, and the keyboard input string may be initialized to any arbitrary characters. It will be read when the RETURN key is hit. When used on the VAX, the keyboard contains 32 program function keys which are activated with the F1 through F8 keys in combination with the SHIFT and/or CONTROL keys. On the IBM computers, the numeric keypad can simulate 24 program function keys. The Unified Graphics System supports the joydisk on the keyboard, the mouse, and the TEKTRONIX 4957 tablet. The PICK or LOCATOR can be assigned to any of these physical units while the STROKE can be assigned to the mouse or tablet. There are two limitations, however, as to how these units may be assigned. First, a PICK or STROKE control unit assigned to the joydisk should not be enabled at the same time that the KEYBOARD is enabled because the joydisk is signaled by striking a keyboard character. Second, the user should not enable the PICK and STROKE control units at the same time if they are assigned to the same physical units. The PICK or LOCATOR function is signaled by hitting a keyboard character if the joydisk is being used, or by hitting a button on the mouse or tablet puck. The STROKE function is initiated by depressing a key on the mouse or tablet puck, and is terminated either by releasing the key or when the maximum number of points have been generated. The Unified Graphics System gives the user some hints as to what is expected of them when the graphic cursor is on the screen. The cursor will be red if only the PICK is enabled. It will be green if only the STROKE is enabled. It will be blue if a LOCATOR position is being requested. Finally, it will be yellow if both the PICK and STROKE are enabled. The PICKID

values are limited to 0 through 32766. PICKID values larger or smaller than these values will be changed to the maximum or minimum values.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

XMIN=<value> This item specifies the smallest addressable X value to be used. The default value is 0.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 4095.

YMIN=<value> This item specifies the smallest addressable Y value to be used. The default value is 0.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 3132.

LORES This item causes the Unified Graphics System to produce pictures with lower resolution. A maximum of four bytes per X-Y coordinate is transmitted instead of the usual maximum of five. The result can be slightly shorter execution and data transmission times.

NOOPT This item causes the Unified Graphics System to suppress the optimization of the X-Y coordinates. Normally, only the necessary bytes are transmitted to the device. When this option is used, all bytes are transmitted. The result can be slightly shorter execution times but longer data transmission times.

BLINK=<value> The value of this item may be WHITE, RED, GREEN, BLUE, YELLOW, MAGENTA, or CYAN. If this item is given, all graphic primitives with the blinking attribute will be in the indicated color and an attribute of VDIM or DIM for any color will result in a dimmer appearance on the screen. If this item is not given, all colors may blink and only white will have a dim mode.

NOCLTXT This item is only available on the VAX computers in the slave-display mode. It causes the Unified Graphics System to refrain from clearing the text display. The net effect is that the Unified Graphics System never touches the text display; it is completely under the control of the user.

CHANNEL=<value> This item is used to open a path from the computer to the terminal. On the VAX computers, the default value is TT. On the IBM computer, the default value is UGDEVICE.

DLINES=<value> This item is used in the interactive mode to tell the Unified Graphics System how many lines the text display actually has. This value should count all of the lines, even if they are not being used in the current mode. The default value is 32.

TLINE=<value> This item is used in the interactive mode to tell the Unified Graphics System which line of text is at the top of the usable portion of the screen. The default value is 1.

BLINE=<value> This item is used in the interactive mode to tell the Unified Graphics System which line of text is

at the bottom of the usable portion of the screen. The default value is the same as the DLINES value.

DCHARS=<value> This item is used in the interactive mode to tell the Unified Graphics System how many characters there are in a line of text. The default value is 80.

PICK=<value> The value of this item may be JOYDISK, MOUSE, or TABLET. It is used in the interactive mode to tell the Unified Graphics System which physical control unit is to be used as the PICK device. The default value is JOYDISK.

STROKE=<value> The value of this item may be MOUSE or TABLET. It is used in the interactive mode to tell the Unified Graphics System which physical control unit is to be used as the STROKE device. The default value is MOUSE.

LOCATOR=<value> The value of this item may be JOYDISK, MOUSE, or TABLET. It is used in the interactive mode to tell the Unified Graphics System which physical control unit is to be used as the LOCATOR device. The default value is JOYDISK.

PICKAP=<value> This item is used in the interactive mode to define the PICK aperture. The default value is 32.

TABPORT=<value> The value of this item may be 0 or 1. It is used in the interactive mode to tell the Unified Graphics System which port the tablet is connected to. The default value is 0.

SIGCHAR=<value> This item is used in the interactive mode to define the "signature characters". The default value is @$%. The Unified Graphics System uses these characters to recognize record sent to it by the terminal when the PICK, STROKE, or LOCATOR are used. The programmer should avoid requiring the terminal user to type records that contain these characters. It is possible, although unlikely, that a typed string containing one of these characters could be misinterpreted by the Unified Graphics System. If these characters are reassigned on the IBM computers, the user should be very careful not to select characters that have special significance to VM.

If a number of changes to a picture (new segments, includes, omits, and deletes) are to be made, the picture can seem to do a lot of flickering. Sometimes it helps to perform the statement:

```
CALL UGPICT('OFF',0)
```

before starting the sequence of changes, and then do:

```
CALL UGPICT('ON',0)
```

after the changes. The screen is not actually turned off, but the changes will usually be made all at once.

For this device, subroutine UGDDAT may be used to send device-dependent data to the display. The character string supplied to that subroutine in the DATA argument must have "TEK4207:" as its first eight characters. These characters will be stripped off, and the remaining characters will be sent to the device without modification. When the data is sent to the device, the unit will

be in the state specified by the OPTIONS argument. In addition, the graphics cursor will be positioned at (XCOORD,YCOORD). These coordinates must be within the current window; if they are not, the scissoring module will eliminate the item.

This terminal has many modes that it may be in and this poses some difficult problems for the Unified Graphics System. It seems to work reasonably well when the terminal is initially in the TEK or ANSI mode. It also works, with a few problems, when the terminal is initially in the EDIT or VT52 mode. The Unified Graphics System tries to determine the mode of the terminal before it writes to the screen. When the system is done with the screen, it tries to return the terminal to its original state. In the slave-display mode, the terminal is returned to its original state at the end of each segment; in the interactive mode on the VAX, it is only returned to its original state when UGCLOS is called. If a program terminates in an unusual manner, the mode of the terminal can be unpredictable.

There is a problem with using the STROKE function on the IBM computer. The STROKE sends unsolicited records back to the host computer and the operating system can lose them or get into other problems. The problems can be minimized by setting a longer time interval, about one second usually seems to work, with the STROKE option of subroutine UGECTL and by not letting the maximum number of points be transmitted. If the terminal does hang up, a simple carriage return may restore things. If that does not work, typing $UGSREJECT or $UGSACCEPT in either all upper or all lower case may help. The first string will reject the STROKE that has been produced so far, and the second will accept it and report it to the program. These problems do not occur on the VAX computer.

The BUTTON control unit also has the potential for problems. When the BUTTON control unit is enabled on the VAX, a special string of characters is loaded into volatile memory. These strings are erased when the control unit is disabled or when UGCLOS is called. If the program terminates in an unusual manner, the strings loaded by the Unified Graphics System may still be in effect and could cause trouble.

The DLINES, TLINE, BLINE, and DCHARS options are used by the system to determine how to position the keyboard input buffer on the screen. The defaults are correct for a real TEKTRONIX 4207 in the TEK or ANSI mode, but they may have to be changed in other modes, or when devices that emulate a TEKTRONIX 4207 are used. For example, when the terminal is in the EDIT or VT52 mode TLINE should be set to 9.

The XMIN, YMIN, XMAX, and YMAX options can be used to restrict the graphics display to a subset of the full screen. In the slave-display mode on the VAX, the user can then use the NOCLTXT option to gain complete control over the text display. The result can be the effective partitioning of the screen into text and graphics areas. Such an application program, may however, be very device-dependent. In the slave-display mode on the IBM

computer, the user may want to set YMIN=200 to keep the picture from overlapping the prompts from VM at the bottom of the screen.

All of the warnings about emulators for the TEKTRONIX 4010 and TEKTRONIX 4105 apply even more strongly to this device. This device is more complicated than either of those earlier devices and will be much more difficult to copy.

SECTION 3.19: THE TEKTRONIX 4510 COLOR GRAPHICS RASTERIZER

The Unified Graphics System supports this device in the non-interactive mode. The device consists of a rasterizer which can drive a number of different color plotting units. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers TEK4510

The resolution of the screen depends on the plotter attached to the rasterizer although it is addressed internally as if it had 4096 by 3133 pixels. The unit can draw points, lines, filled polygons, and hardware-generated characters of a number of sizes. The sizes supported by the Unified Graphics System are equivalent to:

DSIZE=0.01637 (small size)
DSIZE=0.03273 (large size)

The hardware-generated characters may be drawn at angles of 0, 90, 180, or 270 degrees. Graphic primitives may be of any color and intensity level can cause lines of one to four pixels in width to be drawn.

On the VAX computers, the command to send the output file to the plotter is similar to:

```
$ PRINT/PASSALL <file-name>
```

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

XMIN=<value> This item specifies the smallest addressable X value to be used. The default value is 8.

XMAX=<value> This item specifies the largest addressable X value to be used. The default value is 4087.

YMIN=<value> This item specifies the smallest addressable Y value to be used. The default value is 8.

YMAX=<value> This item specifies the largest addressable Y value to be used. The default value is 3124.

LORES This item causes the Unified Graphics System to produce pictures with lower resolution. A maximum of four bytes per X-Y coordinate is produced instead of the usual maximum of five. The result can be slightly shorter execution and data transmission times.

NOOPT This item causes the Unified Graphics System to

suppress the optimization of the X-Y coordinates. Normally, only the necessary bytes are transmitted to the device. When this option is used, all bytes are transmitted. The result can be slightly shorter execution times but longer data transmission times.

GENIL This item indicates that the intensity level data is to be simulated. Normally, all lines are composed of a string of dots three rasters wide. When GENIL is used, a VDIM or DIM intensity line is only one or two rasters wide respectively, and BRIGHT and VBRIGHT levels are four rasters wide. The use of GENIL on this device does not significantly increase the size of the data set being created; it probably does not affect the plotting speed on the plotter either.

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers.

The default options are correct for a TEKTRONIX 4693D plotting unit. These options may have to be changed for other plotting units or for devices similar to a TEKTRONIX 4510. In addition, the quality of the output can be very sensitive to the setup parameters on the plotting unit.

SECTION 3.20: THE VERSATEC ELECTROSTATIC PLOTTER

This graphic device may be used in the non-interactive mode only. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers VEP12FF
IBM computers VEP12FF, VEP12CR

The VERSATEC Electrostatic Printer/Plotter, Model 1200 or Model V80, generates pictures when it is supplied with data in raster form. 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 Unified Graphics System generates pictures for plotters with 200 dots per inch along a scan line, 200 scan lines per inch, and 11 inch wide paper. The hardware character generator on the unit can only be used in the print mode and is not used by the Unified Graphics System. The printer/plotters normally have 8.5 by 11 inch fan-fold paper mounted on them, but some can also use continuous-roll paper. When the Unified Graphics System generates pictures for continuous-roll paper, it only uses the center 10 inches of the paper. When using fan-fold paper, the resolution is 2112 raster units horizontally by 1560 raster units vertically; with continuous-roll paper, the resolution across the page (the Y direction) is 2000 raster units.

On the VAX computers, the VERSATEC Electrostatic Printer/Plotters are connected to the computer over a parallel I/O port. In this

case, the scan lines themselves are written to the printer/plotter. The Unified Graphics System generates these scan lines and writes them to the output file in a highly compressed form. When the output file is sent to the printer/plotter, it is expanded into the actual scan lines. Thus, the data that is actually spooled to the disk usually uses only a few percent as much space as the uncompressed data would have used. However, the time to generate a picture on the VAX computer for the VERSATEC is usually much greater than for most other graphic devices. The command to send the output file to the printer/plotter is:

```
$ PRINT/NOFEED <file-name>
```

On the IBM computers, the VERSATEC is connected to the computer 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. A FILEDEF command must be issued for the output file before your program is executed. A typical command is:

```
FILEDEF UGDEVICE DISK fn ft fm
```

The default DCB parameters are RECFM=FBA, LRECL=126, and BLKSIZE=1260. The LRECL and BLKSIZE parameters may be changed to any reasonable value. The commands to send the output file to the printer/plotter are:

```
CP SPOOL PRINTER CLASS P
PRINT fn ft fm (CC
```

If you wish to have your output plotted on continuous-roll paper the commands are:

```
CP SPOOL PRINTER CLASS P FORM ROLL
PRINT fn ft fm (CC
```

After issuing these commands, the user should restore the print class and form to their original state. Actually, there are other ways to send the file to the printer/plotter but the ones described above are the simplest. The file that is created by the Unified Graphics System contains special carriage control characters. The user must never do anything that causes these carriage control characters to be changed. If they are changed, the controller will not recognize the file as graphic data and will try to print it.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

DDNAME=<value> This item selects the name of the output file. The default value is UGDEVICE.DAT on the VAX computers and UGDEVICE on the IBM computers.

GENIL This item indicates that the intensity level data is to be simulated. Normally, all lines are composed of a single string of dots. When GENIL is used, a MEDIUM intensity line, for example, consists of three closely spaced parallel lines and other intensity levels are similarly generated. GENIL should be used sparingly; on the VAX computers, this option will double the execution time to produce a picture; on the IBM

computers, the data set being created is nearly three times larger.

ROTAXIS The normal orientation of a picture on a fan-fold sheet of paper has the X axis as the longest axis. This option will rotate the picture so that the Y axis is the longest axis. The option also works on continuous-roll paper.

TOPMAR=<value> This option controls the margin at the top of a page of fan-fold output. It should have a value of 0 if a Model 1200 is to be used to plot the file and a value of approximately 75 for a Model V80. The default value is 0. This item is ignored when continuous-roll paper is used.

MAXNSA=<value> MAXNSA gives the size, in words, of the blocks of memory where the line segments that constitute the picture will be accumulated. The default value is 1024.

LINEMX=<value> LINEMX gives the line multiplicity and is the number of scan lines which are developed concurrently. This item is ignored on the VAX computers, on the IBM computers the default value is 32.

The default values for MAXNSA and LINEMX should not normally be changed.

As noted in the description of the TOPMAR parameter, there is a difference between the Model 1200 and the Model V80 when they are used as plotters. The Model 1200 begins plotting about one-half inch down from the perforations while the Model V80 begins plotting almost exactly on the perforations. It is unclear, especially on the IBM computers, how a user can know which model will be connected to the computer when the file is finally plotted.

The controller on the IBM computer can run out of memory when it is preparing a picture. In that case, the controller adds an "error mark" to the picture and discards some of the line segments in the picture. The error mark is at the lower left of the picture and consists of a pair of triangles joined at one vertex in a configuration like a "bow tie". The best thing the programmer can do in this case is to simplify the picture. Sometimes the problem can be overcome by reducing the LINEMX value; however, this change will not always help and can actually increase the number of discarded line segments.

It is also worth emphasizing that the format of the file produced on the VAX computers is totally different from the format of the file produced on the IBM computers; there is no hope of transmitting a file in this form from one system to the other and plotting it.

SECTION 3.21: THE X-WINDOWS PROTOCOL

The Unified Graphics System supports the X-Windows protocol on the DEC VAXSTATION's using their DECwindows programming system. It supports the units as a raster-scan display device in the slave-display and interactive modes. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... SDDXWDO, XWINDOW

The basic VAXSTATION consists of a MicroVAX II, a monochrome or color CRT, an audible alarm, a keyboard, and a mouse. A user normally runs with a number of "windows", that is, independent, rectangular, possibly overlapping, displays on the screen, and the Unified Graphics System writes to one of these windows. Note that this use of the word "window" is different than its use in most graphic packages. It is also possible, by using multiple graphic devices, to have the Unified Graphics System write to more than one window in the slave-display mode. When used in the interactive mode, the KEYBOARD, BUTTON, and LOCATOR control units are available. To use the keyboard, it is the users responsibility to attach the physical keyboard to the window that the Unified Graphics System is using. The 36 buttons that the Unified Graphics System supplies are activated by the keys on the numeric keypad. The pattern was chosen to be consistent with the ASCII terminal emulation on the IBM computer. Buttons 1 through 12 are activated by the PF1, PF2, PF3, 7, 8, 9, 4, 5, 6, 1, 2, and 3 keys respectively. Striking the 0 key immediately before one of these keys adds 12 to its value, and striking the comma key adds 24 to its value. When the LOCATOR is active, the cursor's shape is that of a cross. Any key on the mouse will signal an event by the LOCATOR control unit. The full resolution of the screen is 1024 pixels horizontally by 864 vertically, although a window usually uses less than the full screen. The unit can draw points, lines, filled polygons, and horizontal hardware-generated characters in a number of fonts. The Unified Graphics System supports four fixed spaced fonts whose character sizes are equivalent to:

DSIZE=0.01333 (small size)
DSIZE=0.01833 (medium size)
DSIZE=0.02667 (large size)
DSIZE=0.03667 (very large size)

when the default window size is used. When other screen sizes are used, the corresponding value of DSIZE is proportionally larger or smaller. Graphic primitives may be of any color.

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

XSIZ=<value> The required horizontal size of the window in raster units. The default value is 800.

YSIZ=<value> The required vertical size of the window in raster units. The default value is 600.

XORG=<value> The X coordinate of the upper left corner of the window. The default is to center the window on the screen.

YORG=<value> The Y coordinate of the upper left corner of the window. The default is to center the window on the screen. When using this value, the user must remember that zero is at the top of the screen.

TITLE=<value> The title of the window to be created. This will appear in a band at the top of the window and in the icon for the window. The default value is "Unified Graphics System". If no title is wanted, the user can eliminate this title by supplying the item TITLE=' '.

EXPOSE=<value> This item controls the action to be taken when a window is "exposed", that is, when part or all of it becomes visible after having been covered by something else. Valid values are 0, 1, 2, and 3. The value 0 means newly exposed areas are not redrawn. This is the fastest mode of operation but can result is severe problems if the application program is not prepared to re-draw the current picture on demand. A value of 1 means the X-Windows system is to do the re-drawing if possible. Values of 2 and 3 cause the Unified Graphics System to restore the window when necessary. The difference between these two values is the time at which the screen is normally written. A value of 2 causes the window and picture save area to be written as the picture primitives become available. A value of 3 causes the window to be written as a unit when UGWRIT has processes all of the picture primitives. At present DECwindows does not have the ability to handle exposures and a value of 1 has the same result as a value of 0. The default value is 2. The execution time can vary considerably depending on the value of this item and the manner in which the picture is constructed, for example, from a large number of small graphic segments or a small number of large segments.

GENCH This item indicates that the characters are always to be generated by the software generator in the Unified Graphics System. The default is to try to use the character generators within the device. This item is included because it is not clear how X-Windows will handle remote device; in some cases it may not be possible to use the hardware character generators. In the slave-display mode, no hardware character generator will be used when this option is given. However, in the interactive mode, a hardware character generator is still needed for the keyboard input buffer.

CHANNEL=<value> This specifies an internal symbol for the output device. The default value is a null string which selects the screen of the local device. Under DECwindows, the user can select the display screen on another VAXSTATION by giving this item a value similar to <node>0.0 where <node> represents the DECNET node.

SYNC This item specifies that all actions are to be done in the synchronized mode. It may be necessary when programs are being checked-out. The default is to run in the asynchronous mode.

SECTION 3.22: GRAPHIC PSEUDO-DEVICES

This section describes three pseudo-devices which may be used in the non-interactive mode. These pseudo-devices process the graphic data to varying degrees and then make it available to the user. They may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... PDEVUGS, PDEVLIN, PDEVSVR
IBM computers ..... PDEVUGS, PDEVLIN, PDEVSVR

In all cases, the data is made available to the user by calling a user-supplied subroutine. The skeleton for this subroutine is:

```
      SUBROUTINE    UGXzzz(FLAG,DATA)
      INTEGER       FLAG
      INTEGER*4     DATA(*)
        ...
      END
```

where "zzz" is the last three letters of the selection parameter in UGOPEN. Thus, the subroutine that will be called by the PDEVUGS pseudo-device is UGXUGS. The FLAG parameter is a value supplied when UGOPEN is called. It may be used when more than one pseudo-device of the same type is open at one time. The DATA parameter contains the processed graphic data. DATA(1) always contains the total number of words in the array. DATA(2) contains, with one exception, a value which identifies the graphic data. The actual graphic data, if any, starts in DATA(3).

The first pseudo-device makes the graphic data available to the user-supplied subroutine as unprocessed graphic segments. Its purpose is to make it easy for a user to save pictures in the internal device-independent form of the Unified Graphics System. The graphic data may be saved in a data set or passed to another task or process.

The other pseudo-devices process the picture and present it to the user-supplied subroutine in a form that is similar to the requirements of actual graphic devices. The purpose of these pseudo-devices is to make it possible for a user to incorporate one-of-a-kind graphic devices into the Unified Graphics System.

SECTION 3.22.1: UNPROCESSED GRAPHIC SEGMENTS

The PDEVUGS pseudo-device generates the graphic data as unprocessed graphic segments and other control signals. The graphic data is thus made available in a device-independent form where it may be saved in a data set or passed to another task or process. If the data and control signals are processed correctly, this second task or process can generate the identical picture that the original task or process would have produced.

In addition to the parameter in UGOPEN which selects the device, an additional item will be recognized. This item is:

FLAG=<value> This item is passed to the user-supplied subroutine to identify the pseudo-device. The default value is 1.

The output records that will be passed to the user-supplied subroutine are described below. The user should be aware that new types of records may have to be added in the future. The safest way to code an application program is probably to ignore any additional types of records. That way, if new records are added in the future, existing programs should continue to work as they have in the past.

A record is generated when the graphic pseudo-device is opened:
DATA(1): 2, the word count.
DATA(2): 1, identifies this as an open record.

A record is generated when the graphic pseudo-device is closed:
DATA(1): 2, the word count.
DATA(2): 2, identifies this as a close record.

A record is generated when a new picture is to be started:
DATA(1): 2, the word count.
DATA(2): 3, beginning of picture flag.

A record is generated when the drawing space parameters as supplied by subroutine UGDSPC are available:
DATA(1): 5, the word count.
DATA(2): 4, drawing space flag.
DATA(3): XSIZE.
DATA(4): YSIZE.
DATA(5): AFF.
The graphic data in DATA(3) through DATA(5) are in floating point form.

A record is generated when the view port and window parameters as supplied by subroutine UGWDOW are available:
DATA(1): 10, the word count.
DATA(2): 5, view port and window flag.
DATA(3): XLO of VIEWPRT.
DATA(4): YLO of VIEWPRT.
DATA(5): XHI of VIEWPRT.
DATA(6): YHI of VIEWPRT.
DATA(7): XLO of WINDOW.
DATA(8): YLO of WINDOW.
DATA(9): XHI of WINDOW.
DATA(10): YHI of WINDOW.
The graphic data in DATA(3) through DATA(10) are in floating point form.

A record is generated when a shield parameter as supplied by subroutine UGSHLD is available:
DATA(1): 7, the word count.
DATA(2): 6, shield flag.
DATA(3): XLO of SHIELD.
DATA(4): YLO of SHIELD.

DATA(5): XHI of SHIELD.
DATA(6): YHI of SHIELD.
DATA(7): The shield index (1, 2, 3, or 4).

The graphic data in DATA(3) through DATA(6) are in floating point form.

A record is generated when an extended character set should be selected:

DATA(1): 4, the word count.
DATA(2): 7, extended character set flag.
DATA(3)...DATA(4): The characters "SIMPLEX " or "DUPLEX ".

A record is generated when a graphic segment is available:

DATA(1): 2, the word count.
DATA(2): 8, indicates that the next record is an unprocessed graphic segment.

The record following a type 8 record will be a graphic segment. A graphic segment is the one exception to the rule that DATA(2) contains an identification value. Notice however, that the first word of a graphic segment is the number of words in use in the array, so DATA(1) is consistent with the other records.

A record is generated when the three-dimensional view port and world volume parameters as supplied by subroutine UG3WRD are available:

DATA(1): 12, the word count.
DATA(2): 9, three-dimensional view port and world volume flag.
DATA(3): XLO of VPRT3D.
DATA(4): YLO of VPRT3D.
DATA(5): XHI of VPRT3D.
DATA(6): YHI of VPRT3D.
DATA(7): XLO of WRLDVOL.
DATA(8): YLO of WRLDVOL.
DATA(9): ZLO of WRLDVOL.
DATA(10): XHI of WRLDVOL.
DATA(11): YHI of WRLDVOL.
DATA(12): ZHI of WRLDVOL.

The graphic data in DATA(3) through DATA(12) are in floating point form.

A record is generated when the object volume, eye point, upward direction, and projection flag parameters as supplied by subroutine UG3TRN are available:

DATA(1): 15, the word count.
DATA(2): 10, world volume, eye point, upward direction, and projection flag.
DATA(3): XLO of OBJVOL.
DATA(4): YLO of OBJVOL.
DATA(5): ZLO of OBJVOL.
DATA(6): XHI of OBJVOL.
DATA(7): YHI of OBJVOL.
DATA(8): ZHI of OBJVOL.
DATA(9): X of EYEPT.

DATA(10): Y of EYEPT.
DATA(11): Z of EYEPT.
DATA(12): Delta X of UPDIR.
DATA(13): Delta Y of UPDIR.
DATA(14): Delta Z of UPDIR.
DATA(15): Value of PFLAG.

The graphic data in DATA(3) through DATA(15) are in floating point form.

Some existing programs can get into trouble when this pseudo-device is used. The problem occurs in those programs that try to produce pictures with a given physical size. A program that tries to do this will be using an assumed size for the pseudo-device and not the actual size of the eventual output device.

SECTION 3.22.2: LINE DRAWING PSEUDO-DEVICE

The PDEVLIN pseudo-device generates the graphic data in a form that is similar to the requirements of many two-dimensional line drawing graphic devices.

In addition to the parameter in UGOPEN which selects the device, a number of additional items will be recognized. These items are:

FLAG=<value> This item is passed to the user-supplied subroutine to identify the pseudo-device. The default value is 1.

PASSLS This item specifies that line structure flags are to be passed to the output records. The default is to generate the line structure and pass solid line segments and points to the output records.

XMIN=<value> The minimum X coordinate on the drawing medium. The default is 0.

YMIN=<value> The minimum Y coordinate on the drawing medium. The default is 0.

XMAX=<value> The maximum X coordinate on the drawing medium. The default is 1023.

YMAX=<value> The maximum Y coordinate on the drawing medium. The default is 1023.

XSIZ=<value> The size of the drawing medium in centimeters in the X direction. The default is 25.4.

YSIZ=<value> The size of the drawing medium in centimeters in the Y direction. The default is 25.4.

EXTENSION=<value> This item specifies if the picture may be extended in any direction. The value may be 1 through 4. 1 means the picture can be extended in the low X direction, 2 in the low Y direction, 3 in the high X direction, and 4 in the high Y direction. The default is to not allow any of the extensions.

You may specify that characters may be passed through to the output records by defining the sizes of one or two hardware character generators.

CHR1SZ=<value> Gives the size of the spacing between

characters in raster units.

CHR1XO=<value> Gives the number of raster units from the center of the character to its position point in the Y direction. The default is 0.

CHR1YO=<value> Gives the number of raster units from the center of the character to its position point in the Y direction. The default is 0.

CHR2SZ=<value> Gives the size of the spacing between characters in raster units.

CHR2XO=<value> Gives the number of raster units from the center of the character to its position point in the X direction. The default is 0.

CHR2YO=<value> Gives the number of raster units from the center of the character to its position point in the Y direction. The default is 0.

If only one character size is to be defined, it must be the CHR1 group. The default, if CHR1SZ and CHR2SZ are not given, is to produce all characters with the stroke generator.

CHR90 Indicates that the characters can be drawn at an angle of 90 degrees.

CHR180 Indicates that the characters can be drawn at an angle of 180 degrees.

CHR270 Indicates that the characters can be drawn at an angle of 270 degrees.

The default is to allow characters only at an angle of 0 degrees.

The output records that will be passed to the user-supplied subroutine are described below. The user should be aware that new types of records may have to be added in the future. The safest way to code an application program is probably to ignore any additional types of records. That way, if new records are added in the future, existing programs should continue to work as they have in the past.

A record is generated when the graphic pseudo-device is opened:

DATA(1): 2, the word count.
DATA(2): 1, identifies this as an open record.

A record is generated when the graphic pseudo-device is closed:

DATA(1): 2, the word count.
DATA(2): 2, identifies this as a close record.

A record is generated when a new picture is to be started:

DATA(1): 2, the word count.
DATA(2): 3, beginning of picture flag.

A record is generated when picture description parameters are available:

DATA(1): 6, the word count.
DATA(2): 4, picture description parameters flag.
DATA(3): Intensity level data (1 means VDIM, 2 means DIM, 3 means MEDIUM, 4 means BRIGHT, and 5 means VBRIGHT).
DATA(4): Color data (1 means WHITE, 2 means RED, 3 means GREEN, 4 means BLUE, 5 means YELLOW, 6 means

MAGENTA, 7 means CYAN, and 8 means BLACK).

DATA(5): Blink value (1 means STEADY and 2 means BLINK).

DATA(6): Line structure data (1 means SOLID, 2 means DASHED, 3 means DOTTED and 4 means DOTDASH). If PASSLS is not given, this value will always be 1.

A record is generated when a simple mark, that is, a single dot, is available:

DATA(1): 4, the word count.
DATA(2): 5, mark data flag.
DATA(3): The X coordinate of the mark.
DATA(4): The Y coordinate of the mark.

A record is generated when a line end point is available:

DATA(1): 5, the word count.
DATA(2): 6, line data flag.
DATA(3): The X coordinate of the end point.
DATA(4): The Y coordinate of the end point.
DATA(5): The blanking bit.

A record is generated when a character string is available:

DATA(1): 7+(K+3)/4, the word count.
DATA(2): 7, character string data flag.
DATA(3): The X coordinate of the position point of the first character.
DATA(4): The Y coordinate of the position point of the first character.
DATA(5): The character size (1 means the CHR1 group and 2 means the CHR2 group).
DATA(6): The angle in degrees. This value is always between 0 and 359.
DATA(7): K, the number of characters in the string.
DATA(8): The start of the character string.

This record will only be produced if the options items CHR1SZ or CHR2SZ are given in the call to UGOPEN.

SECTION 3.22.3: SINGLE VALUED RASTER PSEUDO-DEVICE

The PDEVSVR pseudo-device generates the graphic data in a form suitable for a two-dimensional monochrome raster scan graphic device. A single bit of information is produced for each raster point. The bits are organized into scan lines and made available to the user.

In addition to the parameter in UGOPEN which selects the device, a number of additional items will be recognized. These items are:

FLAG=<value> This item is passed to the user-supplied subroutine to identify the pseudo-device. The default value is 1.

GENIL This item specifies that intensity level is to be generated by drawing lines of 1 to 5 raster unit width. The default is to draw all lines with a width of one

raster unit.

XMAX=<value> The maximum X coordinate on the drawing medium. The default is 511.

YMAX=<value> The maximum Y coordinate on the drawing medium. The default is 511.

XSIZ=<value> The size of the drawing medium in centimeters in the X direction. The default is 25.4.

YSIZ=<value> The size of the drawing medium in centimeters in the Y direction. The default is 25.4.

STARTSIDE=<value> This item specifies the side of the picture from which the scan lines are to be produced. The value may be 1 through 4. 1 means the scan lines start with the low X side, 2 with the low Y side, 3 with the high X side, and 4 with the high Y side. The default is 4, that is, the scan lines are produced starting at the top of the picture.

EXTENSION This item specifies if the picture may be extended. The direction of the extension is the side opposite the STARTSIDE side. The default is to not allow any of the extensions.

The maximum length that a scan line can be is 2048 bits. This limitation should be kept in mind when the values of XMAX, YMAX, and STARTSIDE are assigned.

The output records that will be passed to the user-supplied subroutine are described below. The user should be aware that new types of records may have to be added in the future. The safest way to code an application program is probably to ignore any additional types of records. That way, if new records are added in the future, existing programs should continue to work as they have in the past.

A record is generated when the graphic pseudo-device is opened:

DATA(1): 3, the word count.
DATA(2): 1, identifies this as an open record.
DATA(3): K, the number of words per scan line.

A record is generated when the graphic pseudo-device is closed:

DATA(1): 2, the word count.
DATA(2): 2, identifies this as a close record.

A record is generated when a new picture is to be started:

DATA(1): 2, the word count.
DATA(2): 3, beginning of picture flag.

A record is generated when a scan line is available:

DATA(1): K+3, the word count.
DATA(2): 4, scan line flag.
DATA(3): The number of times the scan line should be repeated.
DATA(4): The start of the scan line data. The bits start with the high order bit of the first word and proceed from there. The last word is padded with zeros if necessary.

SECTION 3.23: A GENERIC WORKSTATION

This device-dependent code supports any programmable workstation from the host computer provided a companion program, called a server,. has be prepared for the workstation. The device-dependent code and the server pass information back and fourth between them. The server maintains the picture on the display and monitors any interactive control units that are available. A server program will describe its capabilities to the device-dependent code and the device-dependent code will then treat it as any other device supported by the Unified Graphics System. Although the principal purpose of the device-dependent code is to support high performance workstations with three-dimensional ability, the device-dependent code can operate as a non-interactive, slave-display, or interactive device at any level. It may be selected on the appropriate computer with the following UGOPEN parameters:

VAX computers ..... GENWKST
IBM computers ..... GENWKST

In addition to the parameters in UGOPEN which select the device, a number of additional items will be recognized. These items are:

CHANNEL=<value> This item is used to identify a path from the host computer to the workstation. The required value of this item depends on the CONNECT item. The default value is a null string.

CONNECT=<value> The connection type for the path to the workstation. Values of 0 to 3 are accepted. A value of 0 causes the device-dependent code to act like an interactive pseudo-device. It works by invoking a user supplied subroutine named UGXGWS to do all of the I/O. In this mode, the CHANNEL item is ignored. A value of 1 specifies that the information is to be exchanged between the device-dependent code and the server over an ETHERNET connection. The CHANNEL item must give the identification of the ETHERNET socket to be used. Values of 2 and 3 are reserved for future use. The default value is 1.

ALTORDER This item is used to control the order that the bytes in the integer values are sent and received from the workstation. The default is to use the IBM format, that is, the high order byte is sent out first. When this option is used, the bytes are sent out in the VAX format, that is, the low order byte is sent out first.

In addition the server program may offer the workstation operator many setup options.

At present, values of 0 or 1 for CONNECT are valid on the VAX computers. Only the value 0 is valid on the IBM computers.

When this device-dependent code is used as an interactive pseudo-device, that is, when CONNECT=0, the skeleton of the subroutine is:

```
      SUBROUTINE     UGXGWS(FLAG,DATA)
      INTEGER        FLAG
      INTEGER*2      DATA(*)
        ...
      END
```

The value of FLAG is either zero or one. A zero means data is being supplied by the device-dependent code and a one means the device-dependent code is trying to read data.

The format of the records exchanged by the device-dependent code and the server will not be described here; instead they are described in the Unified Graphics System Internal Operation and Maintenance Manual [Bea83].

At present, a server program exist for Silicon Graphics workstations. The workstations are connected to the host computers over an ETHERNET link. The ALTORDER option should not be used with this device.

SECTION 4: THE SUPPORTED HOST COMPUTERS

The following sections describe the details of using the Unified Graphics System on the various computers which support the system.

SECTION 4.1: THE VAX COMPUTERS

The Unified Graphics System at SLAC runs on the VAX computers under Digital Equipment Corporation's VMS operating system.

This version of the Unified Graphics System supports the full system as described in this document; multiple graphic devices, as many as 32, may be open at one time and the extended character stroke generator may be dynamically selected.

The LINK command may be used to bring together your program, the nucleus of the Unified Graphics System, any extended character sets (simplex and/or duplex), and the device-dependent modules for any graphic devices you plan to use. In addition a library containing the device-independent modules must be supplied. For example, if a program is to use a TEKTRONIX 4010 in an interactive capacity, a VERSATEC plotter in a non-interactive role, and also use the simplex and duplex character sets, then the statements:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ LINK ... <your-program>+ -
  UGSYSTEM:NUCLEUS+ -
  UGSYSTEM:SIMPLEX+UGSYSTEM:DUPLEX+ -
  UGSYSTEM:TEK4010+UGSYSTEM:VEP12FF, -
  UGSYSTEM:OBJLIB/LIBRARY
```

will bring all of the required modules together. Note that the extended character sets must be listed after the nucleus.

The statements shown above include a DEFINE statement to equate the symbol UGSYSTEM to the identification of the directory containing the Unified Graphics System. It would be most convenient for the majority of users if this symbol were put into the System Logical Name Table; however, this document will not assume that this has been done.

SECTION 4.2: THE IBM COMPUTERS

The Unified Graphics System at SLAC runs on an IBM 3081 and an IBM 3033 computer under IBM's VM/XA operating system. Version 1.3 or higher of the VS-FORTRAN compiler is required.

This version of the Unified Graphics System supports the full system as described in this document; multiple graphic devices, as many as 32, may be open at one time and the extended character stroke generator may be dynamically selected. The one exception is that the TIME parameter in subroutine UGEVNT cannot be honored; the subroutine will always act as if TIME were negative.

The mini-disk containing the modules in the Unified Graphics System is made available by the statement:

```
GIME UGS77
```

When you create an executable load module, you should make the subroutine library UGOBJLIB available with a "GLOBAL TXTLIB" statement. In addition, your LOAD command should include any extended character sets (simplex and/or duplex), the nucleus of the Unified Graphics System, and the device-dependent modules for any graphic devices you plan to use. For example, if a program is to use a TEKTRONIX 4010 in an interactive capacity, a VERSATEC plotter in a non-interactive role, and also use the simplex and duplex character sets, then the statements:

```
GIME UGS77
GLOBAL TXTLIB UGOBJLIB VSF2FORT
GLOBAL LOADLIB VSF2LOAD
LOAD <your-program> SIMPLEX DUPLEX NUCLEUS TEK4010 VEP12FF
```

will bring all of the required modules together. Note that the extended character sets must be listed before the nucleus.

There is, however, a problem with the preceding LOAD statement. SIMPLEX, DUPLEX, and NUCLEUS contain common blocks and NUCLEUS contains small dummy versions of the much larger common blocks in SIMPLEX and DUPLEX. As a result, the LOAD statement issues error messages stating that UGA013 (the simplex character set) and UGA014 (the duplex character set) are duplicate identifiers. These messages may be suppressed by the NODUP option on the LOAD statement.

At present it is really not possible to call FORTRAN programs from a PL/1 program when character strings are being passed. This facility was available with earlier versions of PL/1 and FORTRAN compilers but has been lost. There is some faint hope that the facility will someday be restored. If this facility is restored, the Unified Graphics System subroutines will be usable from PL/1 and the following information should be helpful.

For the convenience of the user, a library of declarations for the Unified Graphics System subroutines is available. This library should be made available to the compiler with a statement like:

```
GLOBAL MACLIB UGPLIDCL
```

The declaration statements can then be included in a PL/1 program with statements similar to:

```
% INCLUDE UGOPEN, UGCLOS;
```

When the modules are LOAD'ed, the GLOBAL TXTLIB statement will be similar to:

```
GLOBAL TXTLIB UGOBJLIB PLILIB VSF2FORT IBMLIB
```

Using these declarations will assure that the correct arguments are supplied to the subroutines and will minimize the amount of work done in switching between the PL/1 and FORTRAN environments.

There is a major problem associated with the mixing of PL/1 and FORTRAN modules under the VM/CMS operating system. Under VM/CMS, a program can cross a language boundary only to a depth of one. This means that a PL/1 main program or its PL/1 subroutines can call a FORTRAN subroutine but that FORTRAN subroutine can only call other FORTRAN subroutines; it cannot call a PL/1 subroutine. This means that the error processing subroutine, for example, must be written in FORTRAN. This restriction does not occur under IBM's MVS operating system.

SECTION 5: PROGRAMMING EXAMPLES

This section contains some examples of programs which use the Unified Graphics System.

SECTION 5.1: ELLIPSES AND AN ASTROID

This program shows a relatively simple use of a non-interactive graphic device. The program, as shown, writes its picture to the VERSATEC Electrostatic Plotter, but any non-interactive device may be substituted by changing the call to UGOPEN.

The program illustrates the use of windowing and shielding. Notice that the text within the shields is written before the shields are established. The text within the shields also illustrates some simple superscripting. The use of an error processor is also illustrated; if the error processing subroutine were not used, the array SEGM would have to be many times its current size.

The labels within the shields display the equations of the family of ellipses and the astroid in closed form but the program generates the curves from their parametric representation.

The picture produced by the following program is shown in Figure 5.1.1.

```
      PROGRAM        ELIPSE
C
C  SAMPLE PROGRAM:   ELLIPSES AND AN ASTROID.
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(250)
C
      REAL           VPRT(2,2),WDOW(2,2)
      REAL           SHD1(2,2),SHD2(2,2)
      REAL           PARM,ANGL,COSN,SINE
      INTEGER        BBIT
      REAL           XCRD,YCRD
C
      DATA           VPRT/ 4.5, 0.5,13.5, 9.5/
      DATA           WDOW/-1.0,-1.0, 1.0, 1.0/
      DATA           SHD1/-0.8, 0.3, 0.1, 0.6/
      DATA           SHD2/ 0.0,-0.6, 0.6,-0.3/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  SELECT THE DUPLEX CHARACTER GENERATOR.
      CALL UGOPEN('VEP12FF',99)
      CALL UGFONT('DUPLEX')
C
C  PLOT TITLES AND WINDOW OUTLINE:  FIRST A FRESH PLOTTING
```

```
C  SPACE IS REQUESTED, THEN THE INITIAL DRAWING SPACE IS
C  CREATED, THE SEGMENT IS CLEARED, AND THE TITLES AND
C  WINDOW OUTLINE ARE ADDED TO THE SEGMENT.  FINALLY THE
C  SEGMENT IS TRANSMITTED TO THE DEVICE.
      CALL UGPICT('CLEAR',0)
      CALL UGDSPC('PUT',14.0,10.0,1.0)
      CALL UGINIT('CLEAR',SEGM,250)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,8.0,
     X  'THE',' LL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,7.0,
     X  'ASTROID',' LLLLLL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,6.0,
     X  'AS AN','LL LL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,5.0,
     X  'ENVELOPE',' LLLLLLL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,4.0,
     X  'OF','LL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.5',2.0,3.0,
     X  'ELLIPSES',' LLLLLLL',SEGM)
      CALL UGLINE(' ',VPRT(1,1),VPRT(2,1),0,SEGM)
      CALL UGLINE(' ',VPRT(1,2),VPRT(2,1),1,SEGM)
      CALL UGLINE(' ',VPRT(1,2),VPRT(2,2),1,SEGM)
      CALL UGLINE(' ',VPRT(1,1),VPRT(2,2),1,SEGM)
      CALL UGLINE(' ',VPRT(1,1),VPRT(2,1),1,SEGM)
      CALL UGWRIT(' ',0,SEGM)
C
C  PLOT THE LABELS:  FIRST THE WINDOW IS ESTABLISHED, THEN
C  THE SEGMENT IS CLEARED, THE LABEL OUTLINE AND LABEL ARE
C  ADDED TO THE SEGMENT, AND LASTLY THE SEGMENT IS
C  TRANSMITTED TO THE DEVICE.
      CALL UGWDOW('PUT',VPRT,WDOW)
      CALL UGINIT('CLEAR',SEGM,250)
      CALL UGLINE(' ',SHD1(1,1),SHD1(2,1),0,SEGM)
      CALL UGLINE(' ',SHD1(1,2),SHD1(2,1),1,SEGM)
      CALL UGLINE(' ',SHD1(1,2),SHD1(2,2),1,SEGM)
      CALL UGLINE(' ',SHD1(1,1),SHD1(2,2),1,SEGM)
      CALL UGLINE(' ',SHD1(1,1),SHD1(2,1),1,SEGM)
      CALL UGXTXT('SIZE=0.05',-0.7,0.5,
     X  'THE ELLIPSES.',
     X  ' LL  LLLLLLLP',SEGM)
      CALL UGXTXT('SIZE=0.05',-0.7,0.4,
     X  'X223/C223+Y223/(1-C)223=1',
     X  ' X X  X X  X X      X X  ',SEGM)
      CALL UGLINE(' ',SHD2(1,1),SHD2(2,1),0,SEGM)
      CALL UGLINE(' ',SHD2(1,2),SHD2(2,1),1,SEGM)
      CALL UGLINE(' ',SHD2(1,2),SHD2(2,2),1,SEGM)
      CALL UGLINE(' ',SHD2(1,1),SHD2(2,2),1,SEGM)
      CALL UGLINE(' ',SHD2(1,1),SHD2(2,1),1,SEGM)
      CALL UGXTXT('SIZE=0.05',0.1,-0.4,
     X  'THE ASTROID.',
     X  ' LL  LLLLLLP',SEGM)
      CALL UGXTXT('SIZE=0.05',0.1,-0.5,
     X  'X22/33+Y22/33=1',
     X  ' X   X  X   X  ',SEGM)
      CALL UGWRIT(' ',0,SEGM)
```

```
C
C  PLOT THE CURVES:  FIRST THE LABELS ARE SHIELDED, THEN
C  THE SEGMENT IS CLEARED, THE CURVES ARE GENERATED AND
C  ADDED TO THE SEGMENT, AND THEN THE SEGMENT IS
C  TRANSMITTED TO THE DEVICE.
      CALL UGSHLD('PUT,SHIELD=1',SHD1)
      CALL UGSHLD('PUT,SHIELD=2',SHD2)
      CALL UGINIT('CLEAR',SEGM,250)
      DO 102 PARM=0.0,0.901,0.1
        BBIT=0
        DO 101 ANGL=0.0,360.0,5.0
          COSN=COS(ANGL/57.2957795)
          SINE=SIN(ANGL/57.2957795)
          IF (PARM.EQ.0.0) THEN
            XCRD=COSN**3
            YCRD=SINE**3
          ELSE
            XCRD=PARM*COSN
            YCRD=(1.0-PARM)*SINE
          END IF
          CALL UGLINE(' ',XCRD,YCRD,BBIT,SEGM)
          BBIT=1
  101   CONTINUE
  102 CONTINUE
      CALL UGWRIT(' ',0,SEGM)
C
C  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C  AND THE PROGRAM STOPS.
      CALL UGCLOS(' ')
      STOP
C
      END
C*****************************************************************
      SUBROUTINE    UGXERR(LEVL,SNAM,INDX)
C
C  SEGMENT OVERFLOW SUBROUTINE:  THE GRAPHIC SEGMENT
C  IS TRANSMITTED AND RE-INITIALIZED.
C
      INTEGER       LEVL
      CHARACTER*8   SNAM
      INTEGER       INDX
C
      COMMON        /PLOT/SEGM
      INTEGER*4     SEGM(250)
C
      IF (INDX.EQ.11) THEN
        CALL UGWRIT(' ',0,SEGM)
        CALL UGINIT('CONTINUE',SEGM,250)
        LEVL=0
      END IF
      RETURN
C
      END
```

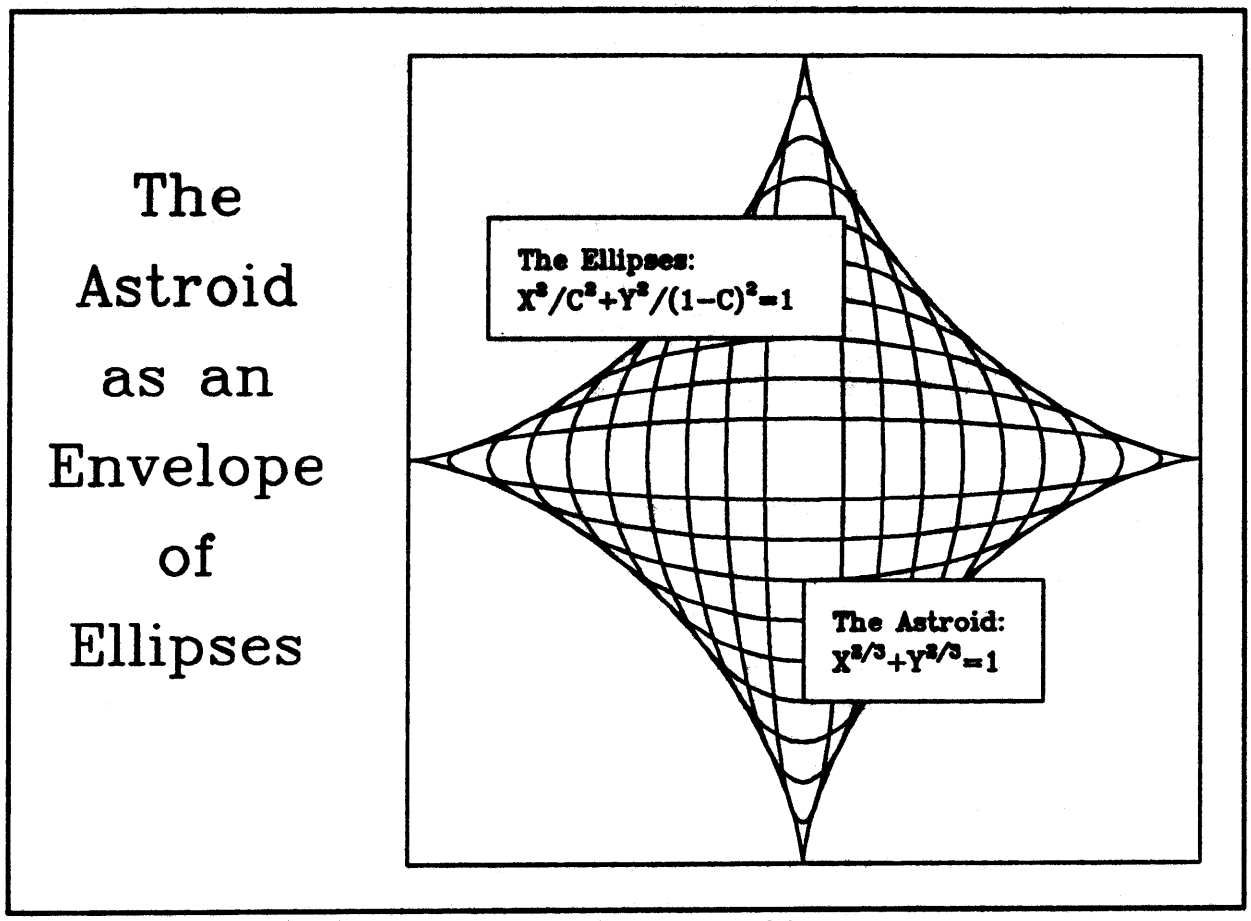


Figure 5.1.1: The Ellipses and an Astroid.

SECTION 5.2: AN INTERACTIVE DRAWING PROGRAM

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 4010 series terminal to draw pictures on the CRT. At a command from the console operator, the picture can be written to the VERSATEC Electrostatic Plotter. Thus, the VERSATEC is acting as a hard-copy device for the TEKTRONIX. Any non-interactive device may be easily substituted for the VERSATEC by changing the first call to UGOPEN. Any interactive device with a keyboard and locator control unit could be used instead of the TEKTRONIX by changing the second call to UGOPEN. If another device is substituted for the TEKTRONIX, it may also be advantageous to change the second and third arguments of UGDSPC because the values shown here were chosen to match the screen of the TEKTRONIX.

The program first puts some instructions for its use on the screen. These instructions describe a number of single letter commands that the program will accept. A command is entered by typing the letter and hitting the RETURN key. The "B" and "D" commands control the drawing, and each of these commands results in the cross-hairs being put on the screen. The console operator may then position the cross-hairs and type any normal key (followed by the RETURN key on some units). "B" blanks to the indicated position and "D" draws to the position. The command "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 execution.

```
      PROGRAM        INTDRW
C
C  SAMPLE PROGRAM:   AN INTERACTIVE DRAWING PROGRAM.
C
      INTEGER        NPTS,NSEG
      PARAMETER      (NPTS=500,NSEG=1100)
C
      INTEGER*4      SEGM(NSEG)
      REAL           XPTS(NPTS),YPTS(NPTS)
      INTEGER        BBTS(NPTS-1)
      INTEGER        KPTS
      CHARACTER*32   STRG
      INTEGER        IARY(2)
      REAL           XARY(1),YARY(1)
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICES,
C  SET THE DRAWING SPACE LIMITS, ENABLE THE KEYBOARD
C  AND LOCATOR CONTROL UNITS, AND SET THE END POINT
C  COUNT TO ZERO.
      CALL UGOPEN('VEP12FF',98)
      CALL UGDSPC('PUT',1.023,0.779,1.0)
      CALL UGOPEN('TEK4010',99)
      CALL UGDSPC('PUT',1.023,0.779,1.0)
      CALL UGENAB('KEYBOARD,LOCATOR')
      KPTS=0
C
C  GENERATE THE INSTRUCTIONS DISPLAY:  THE SCREEN IS
C  CLEARED AND THE INSTRUCTIONS ARE WRITTEN ON THE
C  SCREEN.
  101 CALL UGPICT('CLEAR',0)
      CALL UGINIT('CLEAR',SEGM,NSEG)
      CALL UGTEXT('HARDGN,CENTER',0.50,0.70,
     X  'INTERACTIVE DRAWING PROGRAM',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.60,
     X  'THE FOLLOWING ARE VALID COMMANDS:',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.55,
     X  '  B...BLANK TO LOCATOR POSITION',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.50,
     X  '  D...DRAW TO LOCATOR POSITION',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.45,
     X  '  E...ERASE LAST POINT',SEGM)
```

```
      CALL UGTEXT('HARDGN',0.10,0.40,
     X  '  C...CLEAR ALL CURRENT POINTS',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.35,
     X  '  R...REGENERATE CURRENT PICTURE',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.30,
     X  '  S...SAVE PICTURE ON VERSATEC',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.25,
     X  '  T...TERMINATE THE PROGRAM',SEGM)
      CALL UGTEXT('HARDGN',0.10,0.20,
     X  'ALL OTHER KEYS RESTORE THIS DISPLAY',SEGM)
      CALL UGWRIT(' ',0,SEGM)
C
C  WAIT FOR AN EVENT:  THE PROGRAM GOES INTO THE WAIT
C  STATE UNTIL THE CONSOLE OPERATOR GENERATES AN EVENT
C  ON THE KEYBOARD.  THE INPUT IS EXAMINED AND THE
C  PROGRAM TRANSFERS TO THE PROPER SECTION.
  201 CALL UGEVNT(' ',-1.0,STRG,IARY,XARY,YARY)
      IF (STRG(1:1).EQ.'B') GO TO 301
      IF (STRG(1:1).EQ.'D') GO TO 311
      IF (STRG(1:1).EQ.'E') GO TO 321
      IF (STRG(1:1).EQ.'C') GO TO 331
      IF (STRG(1:1).EQ.'R') GO TO 341
      IF (STRG(1:1).EQ.'S') GO TO 351
      IF (STRG(1:1).EQ.'T') GO TO 361
      GO TO 101
C
C  PROCESS OPERATOR ACTIONS:  THE ACTION INDICATED BY
C  THE CONSOLE OPERATOR IS CARRIED OUT.
C  OPERATION (B):  BLANK TO LOCATOR POSITION.
  301 IF (KPTS.GE.NPTS) GO TO 401
      CALL UGECTL('LOCATOR',STRG,IARY,XARY,YARY)
      KPTS=KPTS+1
      XPTS(KPTS)=XARY(1)
      YPTS(KPTS)=YARY(1)
      IF (KPTS.GT.1) BBTS(KPTS-1)=0
      GO TO 201
C  OPERATION (D):  DRAW TO LOCATOR POSITION.
  311 IF (KPTS.GE.NPTS) GO TO 401
      IF (KPTS.EQ.0) GO TO 301
      CALL UGECTL('LOCATOR',STRG,IARY,XARY,YARY)
      KPTS=KPTS+1
      XPTS(KPTS)=XARY(1)
      YPTS(KPTS)=YARY(1)
      BBTS(KPTS-1)=1
      CALL UGINIT('CLEAR',SEGM,NSEG)
      CALL UGLINE(' ',XPTS(KPTS-1),YPTS(KPTS-1),0,SEGM)
      CALL UGLINE(' ',XPTS(KPTS),YPTS(KPTS),1,SEGM)
      CALL UGWRIT(' ',0,SEGM)
      GO TO 201
C  OPERATION (E):  ERASE LAST POINT.
  321 IF (KPTS.GT.0) KPTS=KPTS-1
      GO TO 201
C  OPERATION (C):  CLEAR ALL CURRENT POINTS.
  331 KPTS=0
      CALL UGPICT('CLEAR',0)
```

```
      GO TO 201
C  OPERATION (R):  REGENERATE CURRENT PICTURE.
  341 CALL UGPICT('CLEAR',0)
      CALL UGINIT('CLEAR',SEGM,NSEG)
      CALL UGPLIN(' ',XPTS,YPTS,KPTS,BBTS,KPTS-1,SEGM)
      CALL UGWRIT(' ',0,SEGM)
      GO TO 201
C  OPERATION (S):  SAVE PICTURE ON VERSATEC.
  351 CALL UGSLCT(' ',98)
      CALL UGPICT('CLEAR',0)
      CALL UGINIT('CLEAR',SEGM,NSEG)
      CALL UGPLIN(' ',XPTS,YPTS,KPTS,BBTS,KPTS-1,SEGM)
      CALL UGWRIT(' ',0,SEGM)
      CALL UGSLCT(' ',99)
      GO TO 201
C  OPERATION (T):  TERMINATE THE PROGRAM.
  361 CALL UGCLOS('ALL')
      STOP
C
C  DISPLAY AN ERROR MESSAGE ON THE SCREEN WHEN THE
C  INTERNAL POINT BUFFER IS FULL.
  401 CALL UGINIT('CLEAR',SEGM,NSEG)
      CALL UGTEXT('HARDGN',0.20,0.10,
     X  'ERROR...POINT BUFFER IS FULL',SEGM)
      CALL UGWRIT(' ',0,SEGM)
      GO TO 201
C
      END
```

SECTION 6: SPECIAL TOPICS

This section discusses a few special topics which should make it possible for a programmer to make better use of the Unified Graphics System.

SECTION 6.1: SAVING PICTURES IN A DEVICE-INDEPENDENT FORM

A group of graphic segments, along with some auxiliary information, constitute a device-independent description of a picture. The PDEVUGS pseudo-device makes it very easy for a program to acquire and manipulate pictures in this form. The programmer may supply a subroutine named UGXUGS to write this picture information to disk or tape, or pass it on to another process or task.

For the convenience of the Unified Graphics user who wants to save pictures on a disk in a device-independent form and later send them to an actual graphic device, a subroutine and a pair of programs are supplied. The module PDEVUGSX contains a subroutine named UGXUGS which writes its input to a file. The programs PDEVUGSN and PDEVUGSI can read this file and send the pictures to an actual graphic device. PDEVUGSN is a non-interactive program which may send pictures to any non-interactive graphic device supported by the Unified Graphics System. PDEVUGSI is an interactive program which may display the pictures on any slave-display or interactive graphic device supported by the system. These modules will be described in the following sections.

SECTION 6.1.1: MODULE PDEVUGSX

The subroutine in the module PDEVUGSX is really very simple. In fact, the entire code, with the comments deleted, consists of:

```
      SUBROUTINE     UGXUGS(FLAG,DATA)
      INTEGER        FLAG
      INTEGER*4      DATA(*)
      INTEGER        INT1
      WRITE (UNIT=FLAG) (DATA(INT1),INT1=1,DATA(1))
      RETURN
      END
```

Thus the subroutine writes the data supplied to it to a file whose FORTRAN unit number is the parameter FLAG.

To obtain this subroutine, the user should include the module PDEVUGSX along with PDEVUGS in the LINK or LOAD statement.

On the IBM computers, a FILEDEF command must be issued for the output file before your program is executed. A typical command is:

FILEDEF <unit> DISK fn ft fm (RECFM VBS BLKSIZE 2048
The BLKSIZE value may be set to any reasonable value but the one shown above is strongly recommended. One good reason for using the values shown above is that the program PDEVUGSI can only process files with these values. Notice that these values or any other do not restrict the size of the graphic segments that can be saved; the actual logical records written by FORTRAN are independent of these values.

SECTION 6.1.2: PROGRAM PDEVUGSN

PDEVUGSN is a program which runs in a non-interactive mode and will read any number of files prepared by the PDEVUGS pseudo-device and the subroutine in PDEVUGSX. It will write to any number of graphic devices on a single run. Specification statements which direct the operation of the program are read from FORTRAN unit 5, and a summary of operations is written to unit 6. Any other unit numbers may be used for the input of files created by PDEVUGS.

It is the responsibility of the user when using this program to form an executable load module and supply the necessary ASSIGN commands (on the VAX computers) or FILEDEF commands (on the IBM computers).

There are three specification statements that this program will recognize. The first is used to define a current input unit which contains PDEVUGS pictures. Its format is:

..FILE UNIT=<number>

The second serves to define a current graphic device to which the pictures will be written. Its format is:

..GDEV <options-for-UGOPEN>

The "..GDEV" specification statement can be continued on the next record by terminating the record to be continued with a minus sign. The final specification record is used to select a series of pictures from the current input unit and write them to the current output graphic device. Its format is:

..PICT FIRST=<number>,LAST=<number>

These specification records must start in column one. Only the first 72 characters of each specification record are used. Both a "..FILE" and a "..GDEV" specification record must precede the first "..PICT" record. No defaults are supplied for the UGOPEN parameters or the UNIT item, but a default of 1 is supplied for FIRST and a default of 9999 is supplied for LAST. Only one input and output file may be open at once. When a second "..FILE" specification statement is found, the previously selected input unit is closed. When a second "..GDEV" specification statement is encountered, the previously selected graphic device is closed. Multiple "..PICT" statements may be given. It is more efficient, but not necessary, for the pictures to be read in sequential order. The items on these specification statements are all processed by subroutine UGOPTN, so misspelled items will be ignored and no error message will be generated.

On the IBM computers, the file containing the specification records should have a RECFM of F and an LRECL of 80.

Consider a simple example on a VAX computer. Suppose a file containing PDEVUGS pictures is named PICS.DAT and that this entire file is to be translated so that it may be plotted on a VERSATEC electrostatic printer/plotter. A command file may be created consisting of:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ LINK/EXECUTABLE=ZZZZTEMP -
  UGSYSTEM:PDEVUGSN+UGSYSTEM:NUCLEUS+ -
  UGSYSTEM:SIMPLEX+UGSYSTEM:DUPLEX+UGSYSTEM:VEP12FF, -
  UGSYSTEM:OBJLIB/LIBRARY
$ ASSIGN PICS.DAT FOR007
$ RUN ZZZZTEMP
..FILE UNIT=7
..GDEV VEP12FF, -
       GENIL, -
       ROTAXIS
..PICT ALL
$ DELETE ZZZZTEMP.EXE;*
```

This command file may then be run interactively (with the @ command) or it may be run as a batch job (with the SUBMIT command). In either case, a file named UGDEVICE.DAT will be created with the pictures for the VERSATEC. The "..GDEV" statement illustrates how that statement can be continued on multiple records.

Now consider a more complicated example on an IBM computer. Suppose two PDEVUGS picture files are available with names of PICS1 DATA A and PICS2 DATA A. Also suppose that all pictures from both files are to be plotted on the VERSATEC Electrostatic Plotter, and pictures 5 through 7 and picture 10 of the first file are to be sent to a high quality device like one of the IMAGEN Laser Plotters. To handle this job, we first create a file containing:

```
..GDEV VEP12FF,DDNAME=UGDEV1
..FILE UNIT=1
..PICT ALL
..FILE UNIT=2
..PICT ALL
..GDEV IMGN300,DDNAME=UGDEV2
..FILE UNIT=1
..PICT FIRST=5,LAST=7
..PICT FIRST=10,LAST=10
```

and give it the name SPECS DATA A. Then the statements:

```
GIME UGS77
GLOBAL TXTLIB UGOBJLIB VSF2FORT
GLOBAL LOADLIB VSF2LOAD
FILEDEF 1 DISK PICS1 DATA A (RECFM VBS BLKSIZE 2048
FILEDEF 2 DISK PICS2 DATA A (RECFM VBS BLKSIZE 2048
FILEDEF 5 DISK SPECS DATA A (RECFM F LRECL 80
FILEDEF 6 DISK TRACE LISTING A
FILEDEF UGDEV1 DISK PICS VEP12FF A
FILEDEF UGDEV2 DISK PICS IMGN300 A
```

```
LOAD PDEVUGSN SIMPLEX DUPLEX NUCLEUS VEP12FF IMGN300 (CLEAR
START
```

may be issued. This series of commands will create a file with a trace of the program (named TRACE LISTING A), a file for the VERSATEC (named PICS VEP12FF A), and a file for the IMAGEN (named PICS IMGN300 A).

SECTION 6.1.3: PROGRAM PDEVUGSI

PDEVUGSI is an interactive program which may read any number of PDEVUGS device-independent picture files and display them on any slave-display or interactive graphic device supported by the Unified Graphics System. In this case, a command file (on the VAX computers) and an EXEC file (on the IBM computers) is supplied to run the program. On the VAX computers the command file is invoked by:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ @ UGSYSTEM:DPICPDEV
```

On the IBM computers, the EXEC program is invoked by the commands:

```
GIME UGS77
DPICPDEV
```

The command or EXEC file will first ask you to enter the type of graphic device to be used. You should respond with something like SDD4010, TEK4010, or any of other slave-display or interactive graphic devices. At this time you should only enter the 7 letter identification and not any additional UGOPEN parameters. The difference between entering SDD4010 and TEK4010 will be in the position of the cursor when it is time for you to enter information into PDEVUGSI; with SDD4010, the cursor will be at the top of the screen while, for TEK4010, the cursor will be in a box at the bottom of the screen. After entering the device type, the command or EXEC file will ask for any additional UGOPEN parameters. These additional parameters may be entered on one or more lines. A load module will then be created, and control will be transferred to it.

When PDEVUGSI itself begins running, it displays a screen of information which shows the valid commands and information about the identity of the current PDEVUGS picture file and the current position within that file. Valid commands to PDEVUGS include a command to select a new PDEVUGS data set for display, a command to display the next picture in the current data set, and a command to skip a number of pictures and display the following picture. These commands may be issued at any time, even if the initial informative display is not on the screen.

SECTION 6.2: TRANSFERRING PICTURES BETWEEN COMPUTERS

A picture file produced by the PDEVUGS pseudo-device and the subroutine in the module PDEVUGSX is a device-independent representation of a group of pictures. Unfortunately, this file is not independent of the computer on which it was generated. This is because the file contains a mixture of INTEGER, REAL, and CHARACTER data, and all of these may have different representations on different computers. To overcome this problem, two additional programs, PDEVUGSE and PDEVUGSD, are provided. The program PDEVUGSE takes a file produced by PDEVUGS and PDEVUGSX, or a part of such a file, and encodes it into a computer-independent format. This computer-independent format consists of 80 character records whose content is normally limited to simple alphabetic and numeric characters. This file may then be transferred to another computer. It is assumed that this file transfer will take care of any necessary conversion of characters between ASCII, EBCDIC, or BCD. On this second computer, the user may reconstruct the device-independent form of the file by running the program PDEVUGSD to decode the computer-independent form. This device-independent file may then be processed by the programs PDEVUGSN or PDEVUGSI to obtain the pictures.

There are a few minor problems that a user must be aware of. The first problem originates in the translation of the characters that occurs during the file transfer. Certain characters that a user may have used in a call to UGTEXT, UGXTXT, or UG3TXT may not translate correctly. For example, the BCD representation does not contain nearly as many characters as ASCII or EBCDIC. Also, both ASCII and EBCDIC contain an exclamation mark but the "standard" translation between these two representations does not convert exclamation marks to exclamation marks. Characters equivalent to the ASCII NULL, DELETE, or BACKSPACE characters may cause serious difficulty and prevent the file from being successfully transferred. However, if a user limits the characters in a call to UGTEXT, UGXTXT, and UG3TXT to the basic character set of the Unified Graphics System, then this problem will never arise.

A second problem that could arise is with the range of a floating point number. The range of the magnitude of a floating point number on an IBM computer is much larger than the range on a VAX. Thus, a computer-independent picture file prepared on an IBM computer may not be able to be transferred to a VAX. However, floating point numbers of the extreme range required for this problem almost never occur in a device-independent picture file.

A third potential problem is with device-dependent data supplied by subroutine UGDDAT. The convention that programs PDEVUGSE and PDEVUGSD use is to pass the first eight characters of the device-dependent data as characters and the rest of the data as hexadecimal characters. This means that the ninth through the last characters are transmitted without modification. This problem really should not occur because most users have no

business using subroutine UGDDAT.

SECTION 6.2.1: PROGRAM PDEVUGSE

PDEVUGSE is a non-interactive program that encodes a device-independent picture, prepared by the PDEVUGS pseudo-device and the subroutine in PDEVUGSX, into a computer-independent picture file. The program can read any number of device-independent picture files, select pictures from them, and write the output to any number of computer-independent picture files in a single run. Specification statements which direct the operation of the program are read from FORTRAN unit 5, and a summary of operations is written to unit 6. Any other unit numbers may be used for input and output files.

It is the responsibility of the user when using this program to form an executable load module and supply the necessary ASSIGN commands (on the VAX computers) or FILEDEF commands (on the IBM computers).

There are three specification statements that this program will recognize. The first is used to define a current input unit which contains a device-independent picture file. Its format is:

```
..DIPF UNIT=<number>
```

The second serves to define a current output unit which will have the computer-independent picture file written to it. Its format is:

```
..CIPF UNIT=<number>
```

The final specification record is used to select a series of pictures from the current input unit and write them to the current output unit. Its format is:

```
..PICT FIRST=<number>,LAST=<number>
```

These specification records must start in column one. Only the first 72 characters of each specification record are used. Both a "..DIPF" and a "..CIPF" specification record must precede the first "..PICT" record. No defaults are supplied for the UNIT items, but a default of 1 is supplied for FIRST and a default of 9999 is supplied for LAST. Only one input and output file may be open at once. When a second "..DIPF" specification statement is found, the previously selected input unit is closed. When a second "..CIPF" specification statement is encountered, the previously selected output unit is closed. Multiple "..PICT" statements may be given. It is more efficient, but not necessary, for the pictures to be read in sequential order. The items on these specification statements are all processed by subroutine UGOPTN, so misspelled items will be ignored and no error message will be generated.

The computer-independent form of a picture file will be larger than the equivalent device-independent picture file. The expansion ratio is variable but can be as big as two or three.

On the IBM computers, the file containing the specification records should have a RECFM of F and an LRECL of 80. The output files should be defined by FILEDEF statements similar to:

```
FILEDEF <unit> DISK fn ft fm (RECFM FB LRECL 80 BLKSIZE 800
```

The BLKSIZE parameter may be changed to any reasonable value, and the RECFM parameter may be changed to F.

Now consider the following simple example on a VAX computer:

```
$ DEFINE UGSYSTEM <UGS-directory>
$ LINK/EXECUTABLE=ZZZZTEMP -
  UGSYSTEM:PDEVUGSE+UGSYSTEM:OBJLIB/LIBRARY
$ ASSIGN PICS.DVI FOR007
$ ASSIGN PICS.CMI FOR008
$ RUN ZZZZTEMP
..DIPF UNIT=7
..CIPF UNIT=8
..PICT ALL
$ DELETE ZZZZTEMP.EXE;*
```

This command file will cause the device-independent picture file contained in PICS.DVI to be converted to a computer-independent picture file and written to PICS.CMI.

SECTION 6.2.2: PROGRAM PDEVUGSD

PDEVUGSD is a non-interactive program that decodes a computer-independent picture file and creates a device-independent picture file similar to the files produced by the PDEVUGS pseudo-device and the subroutine in PDEVUGSX. The program can read any number of computer-independent picture files and produce a device-independent picture file for each input file. All of the pictures in a file are processed; it is not possible to select individual pictures at this stage. Specification statements which direct the operation of the program are read from FORTRAN unit 5, and a summary of operations is written to unit 6. Any other unit numbers may be used for input and output files.

It is the responsibility of the user when using this program to form an executable load module and supply the necessary ASSIGN commands (on the VAX computers) or FILEDEF commands (on the IBM computers).

There is a single specification statement that this program will recognize. It is used to define a current input and output unit numbers. Its format is:

```
..FILE CIPF=<number>,DIPF=<number>
```

This specification record must start in column one. Only the first 72 characters of each specification record are used. No defaults are supplied for the CIPF or DIPF items. When a second "..FILE" specification statement is found, the previously selected input and output units are closed. The items on these specification statements are all processed by subroutine UGOPTN, so misspelled items will be ignored and no error message will be generated.

On the IBM computers, the file containing the specification records should have a RECFM of F and an LRECL of 80. The output files should be defined by FILEDEF statements similar to:

```
FILEDEF <unit> DISK fn ft fm (RECFM VBS BLKSIZE 2048
```

Consider a simple example on an IBM computer. Suppose a file named PICS CMI A contains a computer-independent picture file and it must be converted to a device-dependent picture file named PICS DVI A. To handle this job, we first create a file containing:

```
..FILE CIPF=1,DIPF=2
```

and give it the name SPECS DATA A. Then the statements:

```
GIME UGS77
GLOBAL TXTLIB UGOBJLIB VSF2FORT
GLOBAL LOADLIB VSF2LOAD
FILEDEF 1 DISK PICS CMI A (RECFM FB LRECL 80  BLKSIZE 800
FILEDEF 2 DISK PICS DVI A (RECFM VBS BLKSIZE 2048
FILEDEF 5 DISK SPECS DATA A (RECFM F LRECL 80
FILEDEF 6 DISK TRACE LISTING A
LOAD PDEVUGSD (CLEAR
START
```

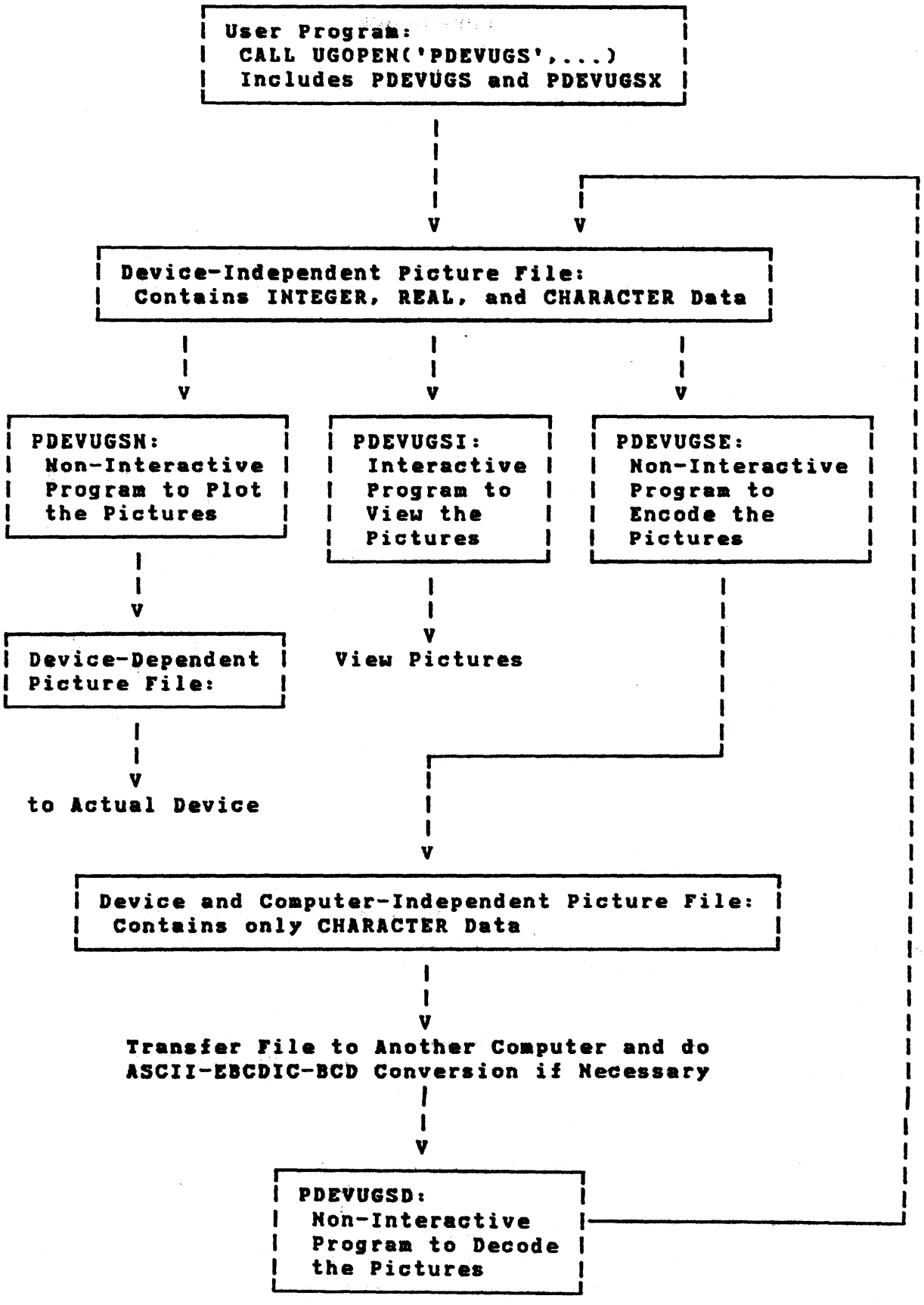
may be issued. This series of commands will create a file with a trace of the program (named TRACE LISTING A) and the required device-independent picture file.

This section on Transferring Pictures Between Computers and the previous section on Saving Pictures in a Device-Independent Form have described a number of programs that can work together with the PDEVUGS pseudo-device to manipulate picture files. These programs are summarized in the chart on the next page.

The chart shows the permissible flow of files through the programs and the properties of the files. For example, the rectangle at the top of the chart indicates a user program. The minimal call to UGOPEN is shown, and the chart indicates that the modules PDEVUGS and PDEVUGSX should be included in the load module. The output of such a program is a device-independent picture file.

This file may be sent to actual graphic devices using the programs PDEVUGSN or PDEVUGSI. Alternatively, the file may be processed by the program PDEVUGSE to produce a device and computer-independent picture file.

This new file may then be sent to another computer. The file only contains character data so any necessary translation is easily handled. The file may then be processed by the program PDEVUGSD to reconstruct a device-independent picture file that is equivalent to the one produced by the user program.


User Program:
CALL UGOPEN('PDEVUGS',...)
Includes PDEVUGS and PDEVUGSX
Device-Independent Picture File:
Contains INTEGER, REAL, and CHARACTER Data
PDEVUGSN:
Non-Interactive
Program to Plot
the Pictures
PDEVUGSI:
Interactive
Program to
View the
Pictures
PDEVUGSE:
Non-Interactive
Program to
Encode the
Pictures
Device-Dependent
Picture File:
View Pictures
to Actual Device
Device and Computer-Independent Picture File:
Contains only CHARACTER Data
Transfer File to Another Computer and do
ASCII-EBCDIC-BCD Conversion if Necessary
PDEVUGSD:
Non-Interactive
Program to Decode
the Pictures

SECTION 6.3: WRITING DEVICE-INDEPENDENT PROGRAMS

The user of the Unified Graphics System may write programs that will run on any graphic device. The programmer may also choose to use some of the more elaborate mechanisms available on some graphic devices and thereby limit the transferability of the program from one device to another. This section will discuss some of these trade-offs.

Picture generation is essentially the same for all graphic devices. However, the polygon-fill primitive should be used with care and in a manner that does not depend on the polygon erasing previously drawn primitives. Another thing that the programmer should do to assure transferability is to examine the implications of the AFF argument of UGDSPC. For non-interactive devices, there are few additional opportunities for device-dependence.

Slave-display devices and interactive devices do, however, provide opportunities for specialization. The lowest common denominator is a storage display device. If a program does not attempt the partial erasure of a picture, the program will run on any graphic device. For raster-scan display devices, graphic segments may be erased by retransmitting the segment with the ERASE option in UGWRIT. Raster-scan displays can usually also erase the entire contents of a window. For refresh display devices, graphic segments may be manipulated by UGPICT. A program that tries to do any of these operations on an inappropriate graphic device will not work correctly. Attempts to bridge the gap between the differences in these types of devices with software have not been particularly successful; such attempts require large amounts of memory and can result in awkward and inefficient use of simple storage display devices. A program may, however, use UGINFO with the DMEDIUM option to determine the type of graphic device in use and proceed accordingly.

On interactive graphic devices, the programmer should carefully choose the type of control devices that the program uses. All interactive devices can be assumed to have a keyboard. Pick, button, stroke, locator, and valuator control units are not universal, and the use of these devices may limit transportability of the program. The Unified Graphics System does make some attempt to simulate nonexisting control devices on some graphic devices, but it is usually best if the program does not rely on these simulated controls. Attempts to simulate a light pen by a keyboard, for example, are very awkward. It is best to write an application program in such a manner that all interaction may be done with the keyboard; additional control units can be used when they are available and are more appropriate for better interaction. A program may use UGINFO with the CONTROLS option to determine the interactive controls available on the graphic device in use.

SECTION 6.4: USING MULTIPLE GRAPHIC DEVICES

Most implementations of the Unified Graphics System allow a program to control a number of different graphic devices. The scheme for doing this is quite simple. Each graphic device is opened with UGOPEN and given a unique identification. Subroutine UGSLCT can use this identification to make any open device active. Only one graphic device is active at one time and all of the other subroutines communicate solely with the active device.

In general, it is possible to do multiple opens on a given graphic device. For example, one may open the VERSATEC printer/plotter twice. Each call to UGOPEN must supply a unique identification and the result will be two distinct output files for the VERSATEC. This can be useful if a program is, for example, creating a large number of pictures and some summary pictures. This scheme will keep the pictures separate but still allow them to be sent to the same device. If there are any problems with multiple openings of a graphic device, they are described in the sections on the supported graphic devices.

There is, however, a serious problem in writing programs which use more than one interactive graphic device. This problem exists if the devices are of the same type or of different types. The problem is that the Unified Graphics System does not supply any way to wait for an event on either of two devices. The Unified Graphics System does not support this because it requires too much code that is not transportable between computers and operating systems. A program that tries to use multiple interactive devices can call UGEVNT with a small positive value of TIME to wait for an event. If nothing occurs when the time interval expires, the user may activate the other interactive device and wait on it for a small time interval. The difficulty, of course, is that this scheme is quite inefficient and response time on the graphic devices will be slow and inconsistent. In addition, some graphic devices may lose those events that occur when it is not active. Also, this scheme will not work on operating systems which do not support the time-out option.

SECTION 6.5: MORE ON THREE-DIMENSIONAL GRAPHIC DEVICES

Two-dimensional graphic devices have been available for a long time. Even though many newer devices have a large number of spurious bells and whistles, they still all have a core of useful features, and it is this core that the Unified Graphics System tries to support. With respect to three-dimensional graphic devices, things are not that straightforward; there is not yet a recognized core and existing hardware varies greatly in its three-dimensional facilities. For this reason, the level of support that the Unified Graphics System can furnish for three-dimensional devices will vary greatly. This section will try to

explain some of these variations.

First, a review of the operations that are possible on a high performance three-dimensional graphic device under the Unified Graphics System will be given. On such a device, any of the three-dimensional graphic primitives may be utilized. The console operator will be able to rotate, translate, and zoom this image. The three-dimensional data will be properly clipped when any part of it is moved out of the three-dimensional view port. The projection parameters may also be changed by the host computer and either a point or parallel projection may be selected. The current projection parameters, as modified by the console operator, may be read by the host computer; using this data, the application program may then send the same graphic segments to another graphic device to reproduce the current picture on this second graphic device.

There are a number of important requirements for a three-dimensional graphic device before it is capable of supporting all of these functions. Some of these requirements are:

1. The device will probably have to be programmable. The reason for this is that it is impossible to imagine a three-dimensional graphic device being produced which provides exactly what the Unified Graphics System requires.
2. Special hardware must be supplied in the graphic device to draw lines and characters. If the device relies on a micro-processor to break a line down into individual dots for the display screen, performance will not be adequate.
3. Special hardware must be supplied to do the clipping. Again, current general purpose micro-processors are not fast enough to do this job.
4. Special hardware must be supplied to perform the three-dimensions to two-dimensions transformation.
5. Finally, a reasonable arithmetic capability must be supplied. For example, 16 bit integer arithmetic is not adequate unless additional higher level operations are also provided.

Because of these problems, there are many ways in which the support given to a specific graphic device by the Unified Graphics System may fall short of the ideal. Possible problem areas are:

1. The initial projection data supplied by the user may not be able to be honored. In this case, a device-dependent initial projection will be used.
2. The three-dimensional data may not be clipped at the three-dimensional view port. In fact, the three-dimensional view port may have to be ignored entirely and the full screen used instead.
3. A point projection may not be possible and a parallel projection will always be used.
4. The full complement of manipulation functions may not be available. In particular, only the rotation and not

the translate and zoom functions may be available.

5. When only rotations are possible, and clipping cannot be done in hardware, it may be necessary to clip the three-dimensional data at the object volume instead of the world volume.
6. The LEFT, RIGHT, and CENTER options of UG3TXT may not be able to be honored. In this case the character string will not be positioned exactly where it is supposed to be but will only be close.
7. The GET,REMOTE operation of subroutine UG3TRN may not be possible. In this case, the values available on the host computer will be returned and an error message will be issued.

It is very important for the user of a three-dimensional graphic device to study the section in this document on that specific device to see exactly what the device is capable of doing.

The Unified Graphics System is not meant to solve every possible problem in computer graphics. There are some elaborate features of some graphic devices that simply cannot be supported by this system. It is important to recognize what can and cannot be done. With respect to three-dimensional devices, the following are examples of things which cannot be done.

1. Any nonrigid motion.
2. Multiple views of an object, for example, top and side views, that rotate together.
3. Loading three-dimensional data into the graphic device and then selecting only part of it for rotation.

The first of these items normally requires local computation and special purpose software. The other two items are conceivable but would require that multiple copies of the three-dimensional view port and transformation parameters be maintained. The extra utility seems to be marginal and such a facility would add considerable complications to the Unified Graphics System, both internally and from a users point of view. It is also unclear how many three-dimensional graphic devices could support such a facility.

On the other hand, there are things that can be done which may not be immediately obvious. It is, for example, possible to create three-dimensional pictures on a computer that does not have a three-dimensional graphic device and transfer the pictures to another computer which has a three-dimensional device and view the pictures there. A user on a VAX without a three-dimensional device could create pictures using the PDEVUGS pseudo-device and the subroutine in the module PDEVUGSX. These pictures can be encoded using the program PDEVUGSE and transferred to the IBM computer. On the IBM computer, the pictures may be decoded using the program PDEVUGSD and viewed on a three-dimensional graphic device using the program PDEVUGSI. Any three-dimensional data in these pictures will be able to be manipulated in any of the ways supported by the three-dimensional device.

REFERENCES

This section contains a list of all of the publications that have been referenced in this document.

[Bea81] R. C. Beach, The Unified Graphics System for FORTRAN 77, Graphic Algorithms Manual, Stanford Linear Accelerator Center, Stanford California 94309, CGTM Number 204 (August 1981, Revised October 1983, November 1985, October 1988, and April 1993).

[Bea83] R. C. Beach, The Unified Graphics System for FORTRAN 77, Internal Operation and Maintenance Manual, Stanford Linear Accelerator Center, Stanford California 94309, CGTM Number 205 (October 1983, November 1985, October 1988, October 1990, and April 1993).

[Boy80] A. Boyarski, HANDYPAK, A Histogram and Display Package, (Release 6.0), Stanford Linear Accelerator Center, Stanford California 94309, SLAC-Report-234 (November 1980, Revised June 1986).

[Cha76] R. B. Chaffee, Top Drawer, Stanford Linear Accelerator Center, Stanford California 94309, CGTM Number 178 (September 1976, Revised January 1978, August 1978, November 1980, April 1987, and October 1989).

[Cha77] R. B. Chaffee, Introduction to Top Drawer, Stanford Linear Accelerator Center, Stanford California 94309, CGTM Number 189 (August 1977, Revised August 1979 and November 1980).

[Cha79] R. B. Chaffee, TD3D, Drawing Three-Dimensional Things with Top Drawer, Stanford Linear Accelerator Center, Stanford California 94309, CGTM Number 199 (November 1979).