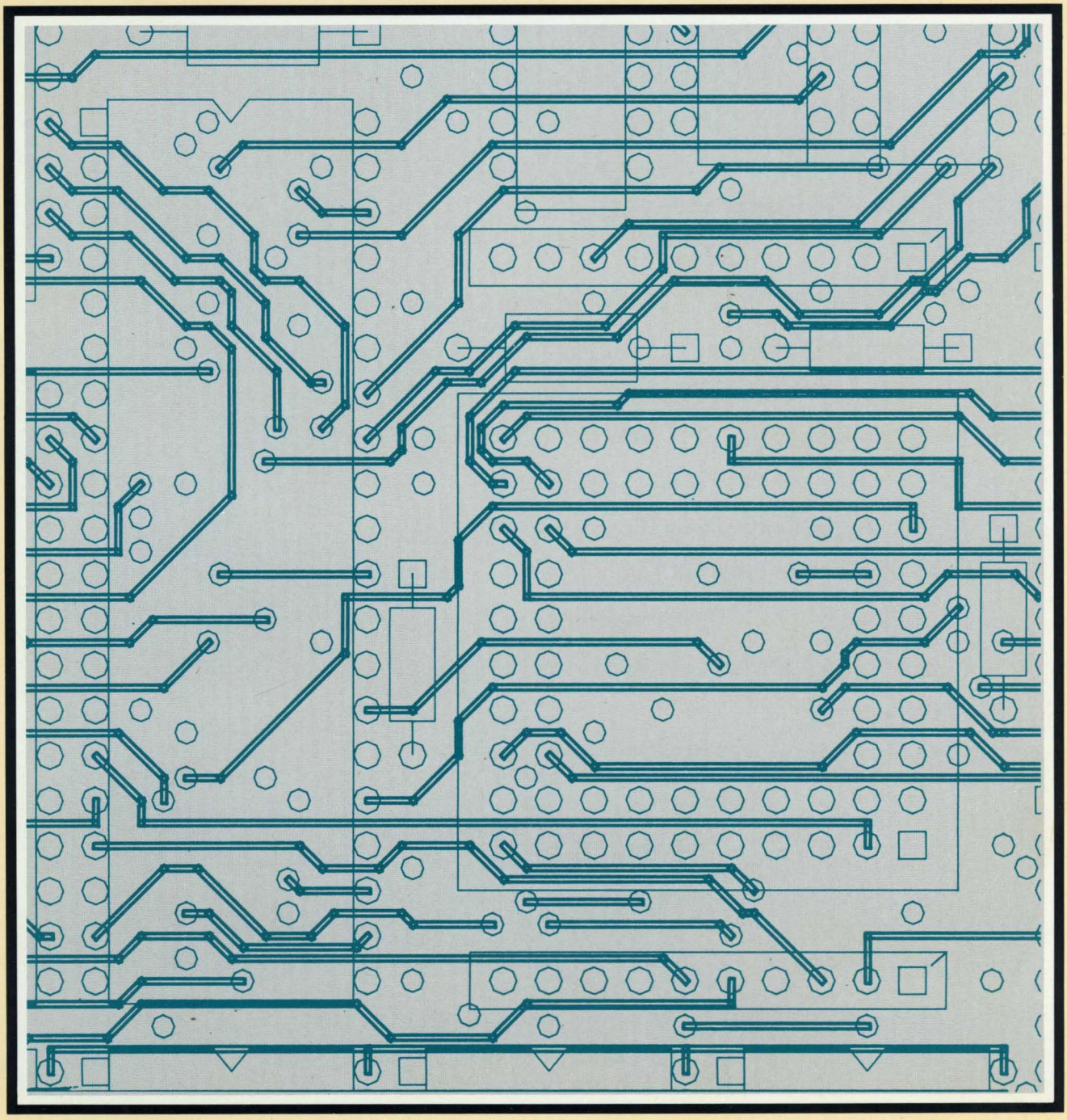
HEWLETT PACKARD

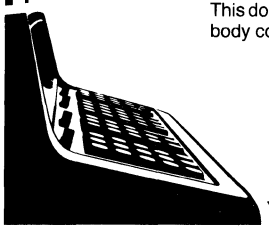
Device-independent Graphics Library

Programmer Reference Manual



Device-independent Graphics Library Supplement for HP-UX Systems

Manual Part No. 97084-90001



Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

November 1983...Second Edition
May 1985...Update

PREFACE

Who needs to use this manual?

The *DGL Programmer Reference Manual* is intended for users of Hewlett-Packard's Device-independent Graphics Library (DGL). Application programmers can use it to learn the features of DGL as well as for reference material as they write their application programs. Operators of graphics systems may better understand the system they use by reading the first two chapters of this manual.

What does it cover?

This manual teaches the DGL system and contains reference information for DGL. Designers and programmers will also find the *Device Handlers Manual* helpful. This manual explains the effect that certain DGL routines have on different devices. This manual does not contain reference information for DGL calls that are operating system dependent. Refer to the particular system supplement manual for this information.

What does it assume?

Very little is assumed by the *DGL Programmer Reference Manual*. An overall understanding of basic concepts of computers (e.g. programs, peripherals) is needed. Some familiarity with computer graphics is helpful. A thorough reading and understanding of Chapter 1 can provide this. Also, knowledge of the FORTRAN programming language is desirable, since that is the language in which examples are written. Experience with additional languages, such as Pascal, would complement the FORTRAN knowledge and allow those languages to be used for graphics application production.

How is it organized?

This manual can be divided into two sections: a "user's guide" and a "reference manual". The "user's guide", consisting of the first two chapters, teaches the basic concepts of computer graphics and the functions of the DGL system. This should be read by the first time user in order to gain a general understanding of the DGL package. The third chapter comprises the "reference manual". This chapter discusses information needed by every programmer as well as more experienced DGL users.

Table of Contents

Chapter 1
Concepts of Computer Graphics

Chapter 2
Description of DGL Functions

Chapter 3
Detailed Description of DGL Subroutines

List of Figures

List of Tables

DGL SUBROUTINES

ZAEND	Disables the enabled alphanumeric device.
ZAINT	Enables the alphanumeric device.
ZALPH	Outputs a text string to the alphanumeric device.
ZASPK	Redefines the aspect ratio of the virtual coordinate system.
ZBEGN	Initializes the DGL system.
ZBEND	Disables the enabled button device.
ZBINT	Enables the logical button device.
ZBMOD	Selects the timing mode for graphics output.
ZBUTN	Returns a button value from the enabled button device.
ZCOLM	Chooses the color model for interpreting parameters in the color table.
ZCOLR	Sets the color attribute for line primitive with the exception of polygon interior fill.
ZCSIZ	Sets the character size attribute for hardware text.
ZDCOL	Redefines the color of an entry in the color table.
ZDEND	Disables the enabled graphics display device.
ZDINT	Enables a graphic display device.
ZDLIM	Defines the logical display limits of the graphics display.
ZDPMM	Converts from world coordinates to millimeters on the graphics display.
ZDPST	Redefines the polygon style of an entry in the polygon style table.
ZDRAW	Draws a line from the starting position in the world coordinate specified.
ZEND	Terminates the DGL system.
ZHIGH	Sets the highlighting attribute for subsequently output primitives.
ZIACS	Given a desired character size, returns the actual character size that will be used by the graphics display.
ZICOL	Inquires the color of an entry in the color table.
ZIESC	Invokes a device-dependent escape function to inquire the graphics display.
ZIPST	Inquires the polygon style of an entry in the polygon style table.
ZIWS	Inquires characteristics of the DGL system.
ZKEND	Disables the enabled keyboard device.
ZKINT	Enables the keyboard device.
ZKYBD	Returns a string from the enabled keyboard device.
ZLEND	Disables the enabled locator device.

ZLINT	Enables the locator device for input.
ZLLIM	Defines the logical locator limits of the locator device.
ZLOCP	Defines the locator echo position on the graphics display.
ZLPMM	Converts from world coordinates to millimeters on the locator surface.
ZLSTL	Sets the linestyle attribute.
ZLWID	Sets the linewidth attribute.
ZMARK	Displays a marker symbol at the current position.
ZMCUR	Makes the picture current.
ZMOVE	Sets the starting position to the world coordinate position specified.
ZNEWF	Performs a new-frame-action on the graphics display.
ZOESC	Performs a device-dependent escape function on the graphics display device.
ZPGDD	Displays a polygon set in a device-dependent manner.
ZPGDI	Draws a polygon set in device-independent manner.
ZPICL	Sets the polygon interior color attribute.
ZPILS	Sets the polygon interior linestyle attribute.
ZPOLY	Draws a connected line sequence starting at the specified point.
ZPSTL	Sets the polygon style for polygon sets.
ZSLOC	Samples the locator device and returns the locator point without waiting for operator response.
ZSVAL	Samples the enabled valuator device and returns the value of the subvaluator specified without waiting for operator response.
ZTEXT	Outputs graphics text on the graphics display.
ZVEND	Disables the enabled valuator device.
ZVIEW	Sets the boundaries of the viewport in the virtual coordinate system.
ZVINT	Enables the logical valuator device.
ZWIND	Defines the boundaries of the window.
ZWLOC	Waits until activation of the locator button then reads from the enabled locator device.
ZWVAL	Waits until activation of the valuator button then returns the value of the specified subvaluator.

*System-specific Calls

Chapter 1
Concepts of Computer Graphics

INTRODUCTION

The Device-independent Graphics Library (DGL), is a set of graphics tools that allows an application programmer to interact with and perform graphics output to a variety of graphics display devices. A user can send two types of output and receive four types of input from several devices.

To provide a device-independent interface, the DGL system is centered around the concepts of a logical device and a work station. A logical device is a hypothetical device which can perform some type of input or output to different physical devices in a uniform manner. DGL has 6 different types of logical devices:

OUTPUT	INPUT
Graphics display	Button
Alphanumeric display	Keyboard
	Locator
	Valuator

DGL combines these logical devices into a "super device" called a work station. A work station is a group of, at most, one each of the six logical devices that DGL treats as a unit. It allows a single application program to send different types of output and receive different types of input from several devices. Figure 1-1 illustrates the structure of the DGL system.

DGL is a set of building blocks upon which programmers can build a wide variety of graphics tasks (which requires a minimal amount of program space) without sacrificing performance. DGL is an extremely flexible product which can be extended to support new peripherals as they become available.

The following sections describe the five major functional areas of DGL: graphics output primitives, output primitive attributes, viewing transformations, input, and control. The concepts and examples introduced here will be developed in detail in the following chapters.

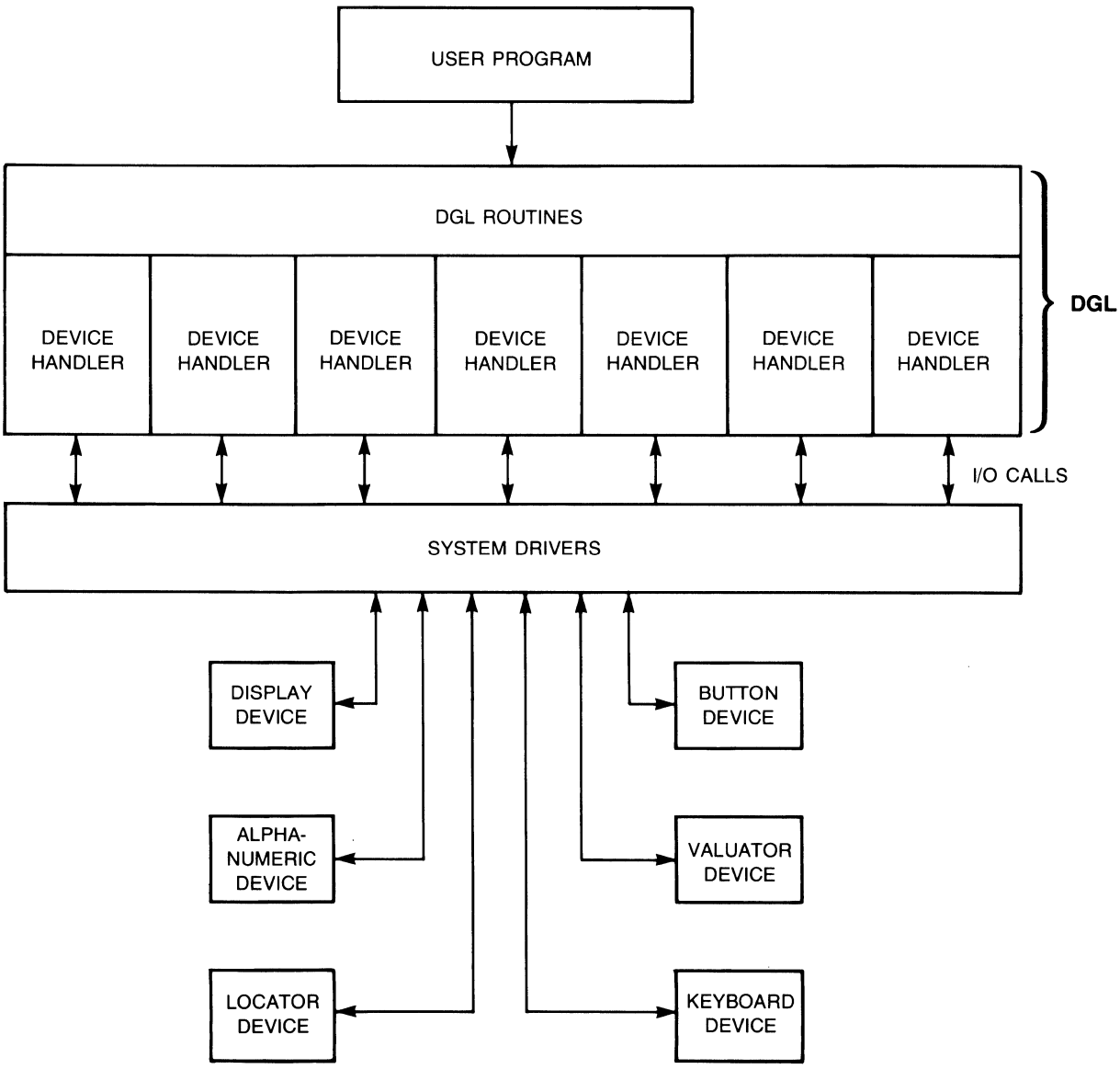


Figure 1-1. DGL System Structure

GRAPHICS OUTPUT PRIMITIVES

Graphics output primitives are the building blocks of a graphics picture. Just as an algorithm is broken into the simplest possible instructions to create a computer program, a picture can be broken down into graphics output primitives. DGL has six types of graphics primitives: lines, polylines, text, markers, polygon sets, and moves.

- o A line is a single line segment.

- o A polyline is a a series of connected lines specified as one primitive. Note that the polyline primitive can define the outline of a polygon but not its interior region. The polyline primitive makes it possible to draw a series of connected lines all at one time, instead of making a series of draws. This is done by specifying the number of points to be drawn and the X and Y arrays of the coordinates to be connected.

- o Text is a string of graphics text characters which is output. The lower left corner of the first text character begins at the starting position. Text is normally output by the device's hardware character generator. For devices that do not generate graphics text, DGL will output software-generated text.

- o A marker is a symbol, such as a diamond or star, that represents a data point.

- o A polygon is a multisided, closed, plane figure that includes both the edge or sides of the figure and the region enclosed by the edge. A polygon set is a group of associated polygons each of which is a closed plane figure bounded by straight lines. The polygon set primitive allows the user to draw a series of connected lines to form closed figures (polygons) in the world coordinate system. Like polylines, the user specifies the number points to be drawn and the X and Y arrays to be connected. Unlike polylines, the polygon set primitive is planar and allows the user to specify how the interior of these bounded regions is to be filled with a specific style and color of fill pattern.

- o A move redefines the starting position for the next graphics primitive. Otherwise, each primitive begins where the previous primitive ended.

In Figure 1-2, moves and lines were used to create the axes and grid, markers generated the data points, and polylines connected the data points. The move primitive positioned the labels on the axes, and the text primitive wrote the labels.

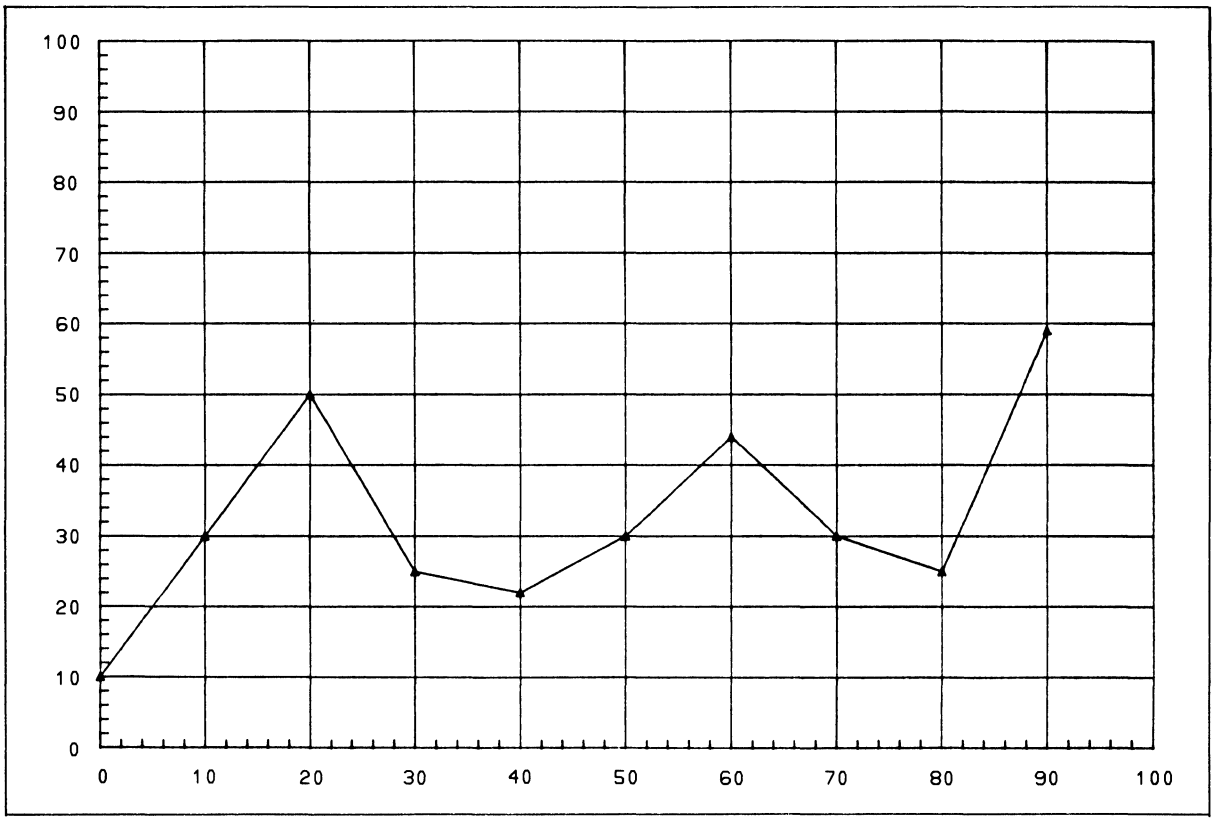


Figure 1-2. Examples of Graphics Output Primitives

The location of each output primitive is specified in the world coordinate system. The world coordinate system is an abstract, two-dimensional space expressed in units selected by the user.

A point is defined by its X and Y coordinates. The starting point of an output primitive is the ending point of the previous output primitive. A line can be created by specifying a point to be connected to the previous point. This action is known as a "draw". The end of that line then becomes the starting point of the next output primitive and another line can begin at those coordinates. The starting position can also be moved to another point without drawing a line; this is known as a "move". In this way, a picture can be drawn by successive moves and draws.

Using lines, polylines, text, markers, polygon sets, and moves, complex graphics images can be constructed as shown in Figure 1-3.

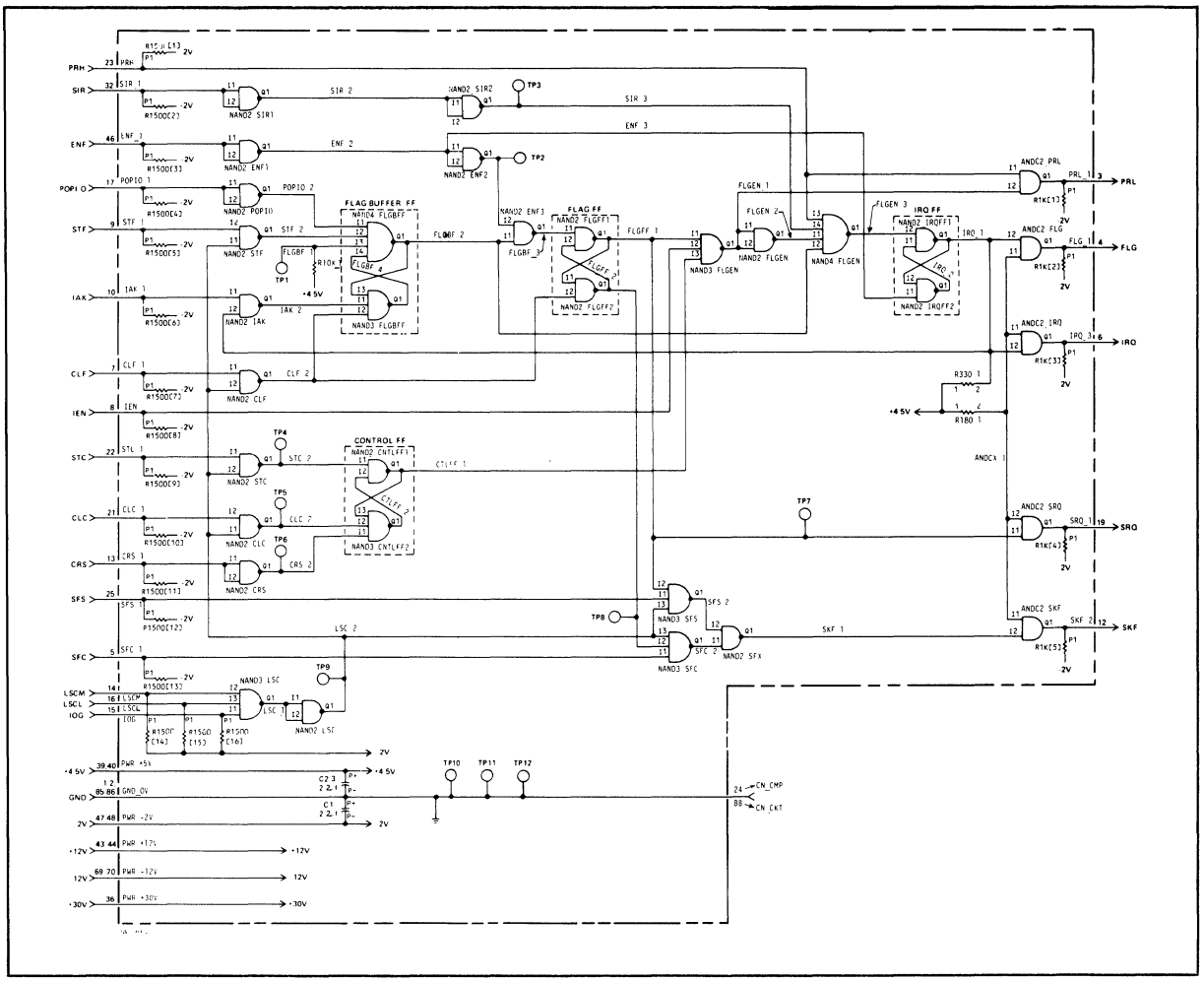


Figure 1-3. Examples of Graphic Output Using DGL

PRIMITIVE ATTRIBUTES

Primitive attributes are the visual characteristics of graphics primitives. For example, a line has the attributes of color, highlighting, linewidth, and linestyle. Different values of the linestyle attribute give a line the different appearances shown in Figure 1-4.

Figure 1-4. A Primitive Attribute: Linestyle

For most CRTs and all graphics plotters, primitive attributes are fixed when they are output; they cannot be changed retroactively. To change the image of an output primitive, clear the display, change the desired attribute values, and display the primitive again.

VIEWING TRANSFORMATIONS

As objects are created, they are transformed for viewing. A *viewing transformation* is the method whereby an object, a collection of graphics primitives defined in an abstract coordinate system, is converted to an image displayed on a physical graphics display device. The bounds of the system in which an object is defined and the location of the object's image on the display device can be defined by the user.

DGL's viewing transformations are all in two dimensions. To define the two-dimensional viewing transformation, three questions must be addressed:

1. What portion of the world coordinate system will be used to define and display the object (where is the *window*)?

2. Where will the window be mapped in the virtual coordinate system (where is the *viewport*)?

3. Where will the virtual coordinate system be placed (what are the *logical display limits*)?

Setting the window allows the application to specify which portion of the world coordinate space is to be used. This provides the application program with the flexibility of working in units that are relevant to the application. Since the window is in the same coordinate space in which objects are defined, the bounds of the window can affect the size of the image displayed. The larger the limits of the window, the smaller the object's image. If the window is specified as having smaller limits than the object, unpredictable device-dependent results will occur, since the DGL system does not perform software clipping. Figure 1-5 shows the effect of two different windows on the same object. The window on the left shows a window which is approximately as large as the object contained within it. The window on the right shows the same object in a larger window.

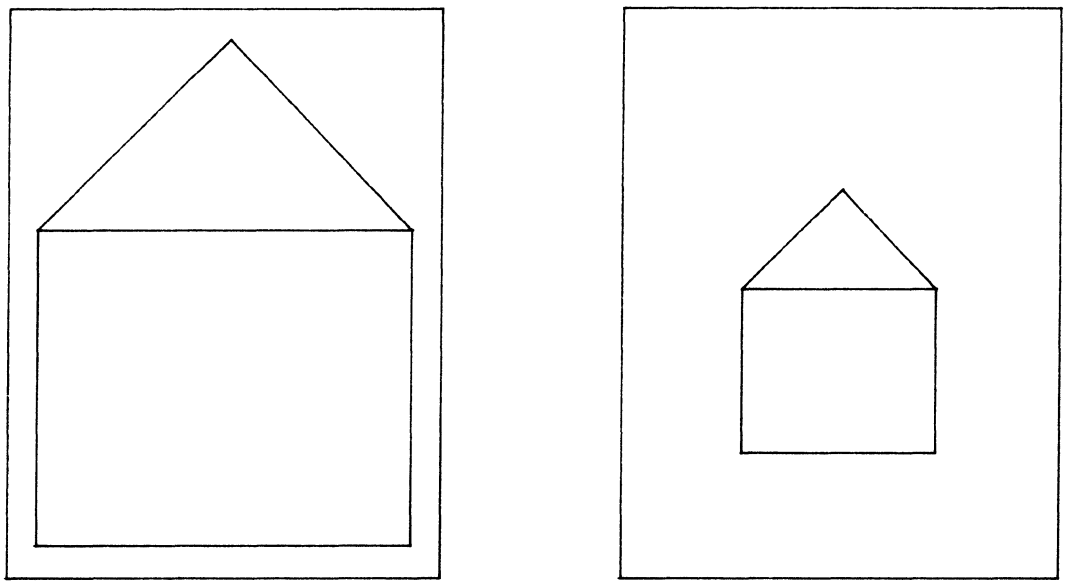


Figure 1-5. Two Images Resulting From Different Windows

The area on a display device where images appear is called a viewport. The dimensions of the viewport are set in the virtual coordinate system, which is the two-dimensional system whose units range from 0.0 to a maximum of 1.0. The viewport can map to any area on the surface of the graphics display device. Multiple viewports can be defined sequentially so that several images can be displayed together. Figure 1-6 shows such an example.

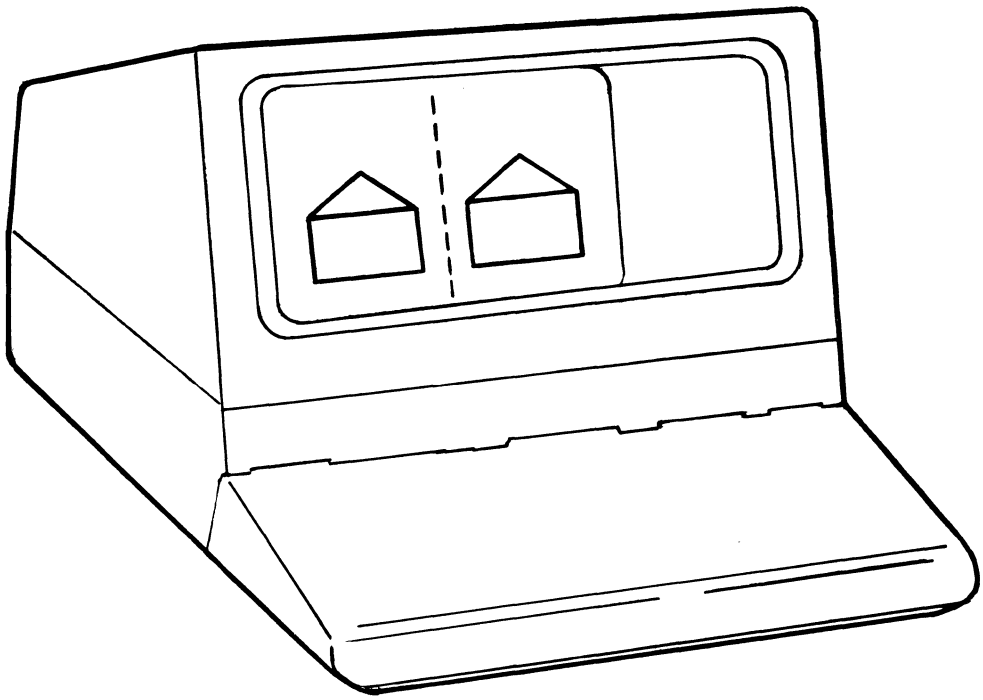


Figure 1-6. Two Viewports, One Object

Once the window and viewport have been defined, output primitives within the window are mapped into the viewport so that each point in the window corresponds to a point in the viewport. If the aspect ratio (height-to-width ratio) of the window and viewport are not equal, this mapping will distort (shrink or stretch) the image. This is the only stage of the transformation process where distortion can occur.

To offer full flexibility in accessing display devices of different sizes and shapes, DGL lets the application program control the mapping of the virtual coordinate system to the display surface. The logical display limits can be defined to alter the absolute size of the image that is produced. These limits bound the logical display surface, making it the entire display surface or some subset of it. The adjective "logical" distinguishes this surface from the true surface of the display device itself, sometimes called the physical display surface.

The steps in the mapping of objects in the window to images on a logical display surface are somewhat complex. They are described more fully in Chapter 2, where the DGL routines that control this mapping are presented.

INPUT

Input functions return information to an application program from input devices. There are four available devices: button, keyboard, locator, and valuator.

A button device returns an integer value corresponding to a button selected. Button devices are used for applications that require only simple choices by the operator (rather than complex strings of characters such as those entered with a keyboard). A button device is analogous to a start/stop/select device, such as a button on a slide projector.

A keyboard device allows the operator to enter a string of characters, typically from an alphanumeric keyboard. The keyboard function returns the string that was entered and the number of characters in the string to the calling program. This device is useful for complex responses by operators.

A locator device provides a means of inputting a real coordinate pair that depicts a point on the view surface. Typical locator devices are graphics tablets and the graphics cursor on a terminal. Using a locator is analogous to using one's finger: With a finger, one can point to the location of a city on a map thereby determining its latitude and longitude.

A valuator device allows the operator to return a value between 0.0 and 1.0 to the application program. The device can be a control dial or multiposition switch, analogous to a dimmer switch on a light. The dimmer can be rotated to any position, which sets the light at the desired intensity.

DGL obtains information from the input devices by either *requesting* or *sampling* the input. Before requested information can be returned to the application program, the operator must enter a termination command (such as pressing the return key). Sampled output is returned to the application program immediately, with no operator interaction required. Information from all logical input devices can be obtained by the request method. Only the locator and valuator devices may be sampled.

CONTROL

Control functions are used to manage various aspects of a graphics application. Some of these are initialization and modification of the graphics environment, inquiry of the features currently in effect, and escape functions.

ALPHANUMERIC OUTPUT

DGL supports a mechanism that allows the application program to send non-graphics text and data to an alphanumeric display device. This call can be used to send prompts, give status, or print other messages. To maintain the integrity of communications between the DGL system and the devices accessed, this mechanism, rather than I/O mechanisms of other subsystems, should be used to generate alphanumeric output on DGL devices.

Chapter 2
Description of DGL Functions

GENERAL

DGL is a set of graphics tools that allows the application programmer to interact with and send output to a wide variety of graphics devices. A user can send two types of output and receive four types of input in a uniform manner from several devices. This chapter describes the functions of DGL. Detailed descriptions of DGL routines are given in Chapter 3.

An application program can perform two types of output: graphics and alphanumeric. Graphics output creates elements of a graphics object. An object may consist of line segments, polylines, markers, text, polygon sets and moves. The appearance of the graphics output can be controlled by varying the values of the primitive attributes: color, highlighting, linestyle, linewidth, and character size.

Alphanumeric output can be used to send operator prompts and messages. It is composed entirely of alphanumeric text and is not affected by the primitive attributes.

DGL transforms objects that are expressed in terms of an abstract two-dimensional space into visible images on a physical graphics device, by means of a viewing transformation (see Figure 2-1).

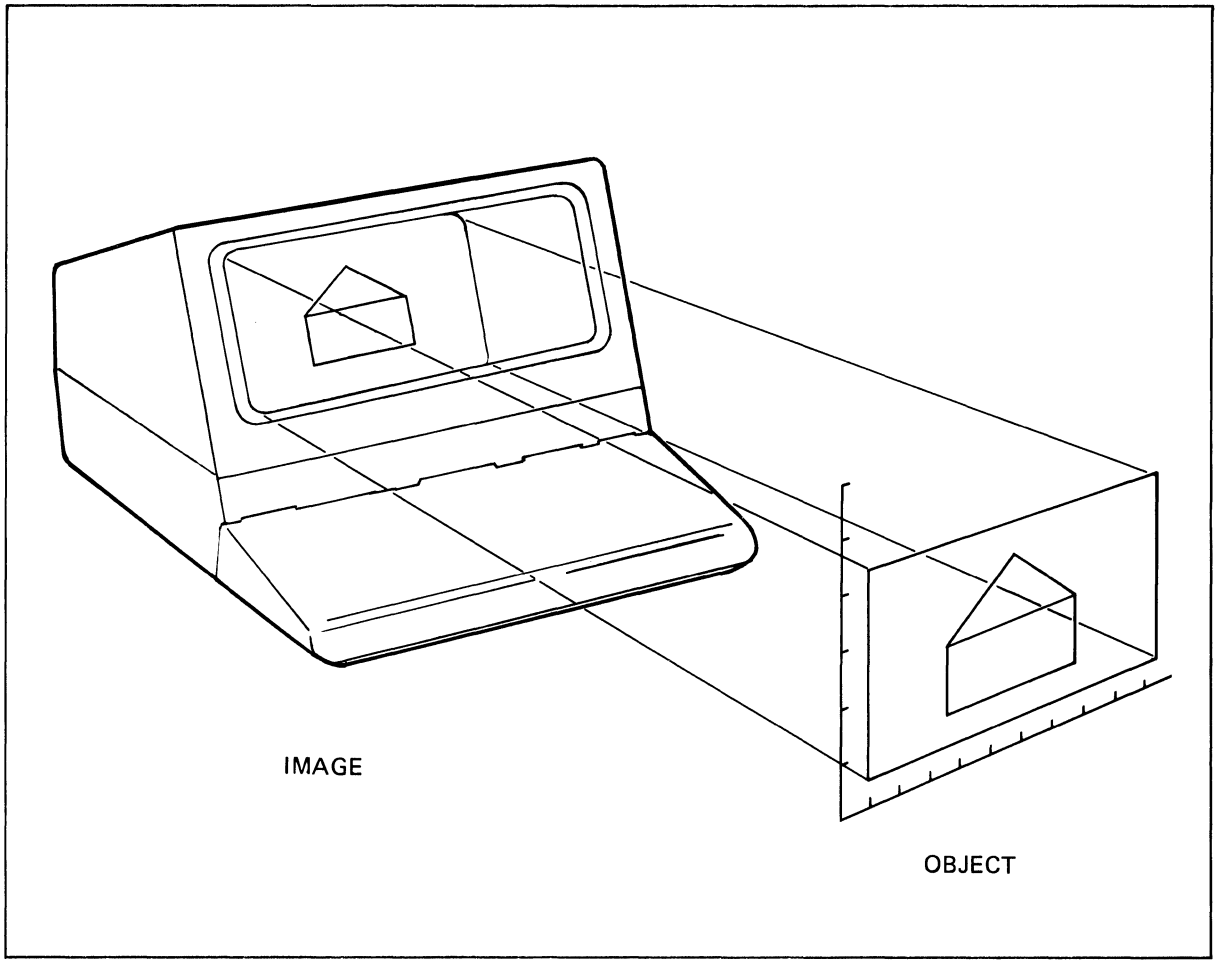


Figure 2-1. The Result of the Viewing Transformation

To provide a device-independent interface, the DGL system is centered around the concepts of a logical device and a work station. A logical device is a hypothetical device which can perform a certain type of input or output in a uniform manner independent of the physical device. For example, a logical locator always returns an (X,Y) coordinate regardless of whether the physical device is an HP 9111 Data Tablet or an HP 2623 Graphics Terminal.

DGL has six different types of logical devices:

OUTPUT

Graphics display
Alphanumeric display

INPUT

Button - returns an integer
Keyboard - returns alphanumeric text
Locator - returns an (X,Y) point
Valuator - returns a real value between 0.0 and 1.0

Several logical devices can be combined to form a work station. A work station is a collection of logical devices and can be thought of as a "super" logical device. It can be composed of any or all of the logical device types but cannot contain more than one of any type. For example, it is not possible to have an HP 2623 locator and an HP 9111 locator in the same work station. One work station can input data points from a locator on a graphics tablet, input a text string from a keyboard on a terminal, and display the data points on a CRT display.

An application program may also control when visual changes occur on the graphics display. Graphics commands can either be sent immediately to a device or can be placed in an internal DGL buffer before being transmitted. The mode used affects the efficiency of the DGL system and the immediacy of visual changes.

The DGL system allows an application program to inquire the status of the system at any time after the program initializes the system.

In summary, DGL can perform many graphics tasks. This chapter gives a description of each of the DGL functions as well as how to use them. Example programs and their output illustrate how one can use these functions to easily create graphics application programs.

OUTPUT

General Information

This section discusses the two types of DGL output. The elements of graphics output and how they can be used to create objects are presented first. The method of sending alphanumeric messages is explained at the end of the section.

Graphics Output Primitives

GENERAL

Graphics output primitives are used to define an object in terms of the world coordinate system. This is an abstract two-dimensional space expressed in units defined by the user. The programmer can define an object in familiar units, such as kilometers, seconds, or grams.

Graphics output primitives are interpreted as commands to generate lines, markers, polygon sets, and graphics text at specified locations on the graphics display device. Each primitive begins at the position where the previous primitive ended. This means that the starting position of a primitive is not explicitly specified, but rather implied by the previous output primitive. For example, a marker is always output centered about the starting position.

STARTING POSITION

ZMOVE (WX, WY) — Define a starting position

The subroutine ZMOVE allows the user to specify a new starting position. A call to this subroutine causes the next output primitive to start at the coordinate point (WX,WY). This point will then be the endpoint of a line segment, the left edge of a graphics text string, or the point that a marker is centered around.

LINES

ZDRAW (WX, WY)	Draw a line to point (WX,WY)
ZPOLY (NPTS, XVEC, YVEC)	Draw a connected line sequence

Line output primitives are used to draw graphics objects. A call to ZDRAW causes a line to be drawn from the starting position to the world coordinate point specified by (WX,WY). The starting position then becomes (WX,WY).

DGL also provides the capability to draw a series of connected lines by making one call to the subroutine ZPOLY. This call first does a move to the point specified by the first element of the arrays XVEC and YVEC, where XVEC and YVEC contain a series of (X,Y) coordinate points. The line sequence begins at this point and is drawn to the second specified point, then to the third and continues until NPTS-1 lines are drawn. The starting position for the next primitive becomes the point specified by XVEC(NPTS) and YVEC(NPTS).

The primitive attributes of color, highlighting, linestyle, and linewidth affect the appearance of lines. Primitive attributes are discussed later in this chapter. Figure 2-2 illustrates how a picture can be drawn using a combination of ZMOVE and ZDRAW commands or one call to ZPOLY.

```
      .
      .
      .
      INTEGER NUMHOS, NUMTRE
      REAL XHOUS(7), YHOUS(7), XTREE(23), YTREE(23)
      DATA NUMHOS/7/, NUMTRE/23/
      DATA XHOUS/-.69,-.69,-.15,-.15,-.42,-.69,-.15/
      DATA YHOUS/-.38,-.85,-.85,-.38,-.08,-.38,-.38/
      DATA XTREE/.65,.65,.46,.54,.50,.58,.54,.62,.58,.65,.62,
     *           .69,.77,.73,.81,.77,.85,.81,.88,.85,.92,.73,.73/
      DATA YTREE/-.85,-.69,-.69,-.62,-.62,-.54,-.54,-.46,-.46,-.38,
     *           -.38,-.31,-.38,-.38,-.46,-.46,-.54,-.54,-.62,-.62,
     *           -.69,-.69,-.85/
      .
      .
      .
C
C     Draw a house using a combination of ZMOVE & ZDRAW commands
C
      CALL ZMOVE(XHOUS(1), YHOUS(1))
      DO 20 I=2, NUMHOS
         CALL ZDRAW(XHOUS(I),YHOUS(I))
20    CONTINUE
C
C     Now draw the tree using one ZPOLY command
C
      CALL ZPOLY (NUMTRE,XTREE,YTREE)
      .
      .
      .
```

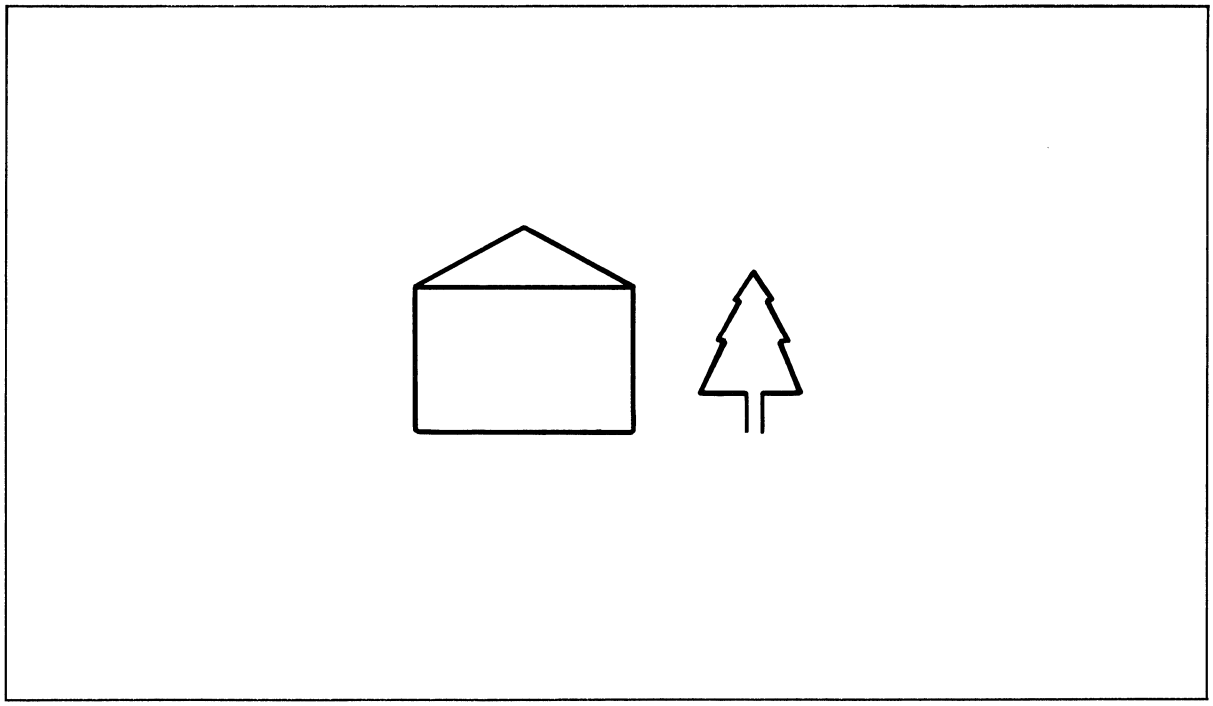


Figure 2-2. Use of ZMOVE, ZDRAW and ZPOLY

TEXT

ZTEXT (NCHARS, STRING) Generate graphics text

DGL provides the ability to draw graphics text at a desired location on a graphics display device. Text is normally generated by a device's hardware character generator. DGL will simulate hardware text generation for any device that does not support a hardware character generator.

A call to ZTEXT outputs the first NCHARS number of characters contained in the array STRING. The character string is drawn or displayed on the graphics display device beginning at the starting position. Text is normally drawn from left to right and without character slant. The primitive attributes, color, linestyle, linewidth, highlighting, and character size affect graphics text.

ZTEXT leaves the current position at a device-dependent location. After calling ZTEXT, define the starting position explicitly by calling ZMOVE, ZPOLY, ZPGDD, or ZPGDI before generating more graphics primitives.

```
      INTEGER TITLE(11)
      DATA TITLE/2H H,2HOM,2HE ,2H   ,2HSW,2HEE,
     *           2HT ,2H   ,2HHO,2HME,2H  /
      .
      .
      .
C
C     Draw a house using a combination of ZMOVE &
C     ZDRAW commands
C
      CALL ZMOVE(XHOUS(1),YHOUS(1))
      DO 20 I=2,NUMHOS
           CALL ZDRAW(XHOUS(I),YHOUS(I))
20    CONTINUE
C
C     Now draw the tree using one ZPOLY command
C
      CALL ZPOLY(NUMTRE,XTREE,YTREE)
C
C     Write out the title, 'HOME SWEET HOME', using graphics text
C
      CALL ZMOVE(X1,Y1)
      CALL ZTEXT(22,TITLE)
      .
      .
      .
```

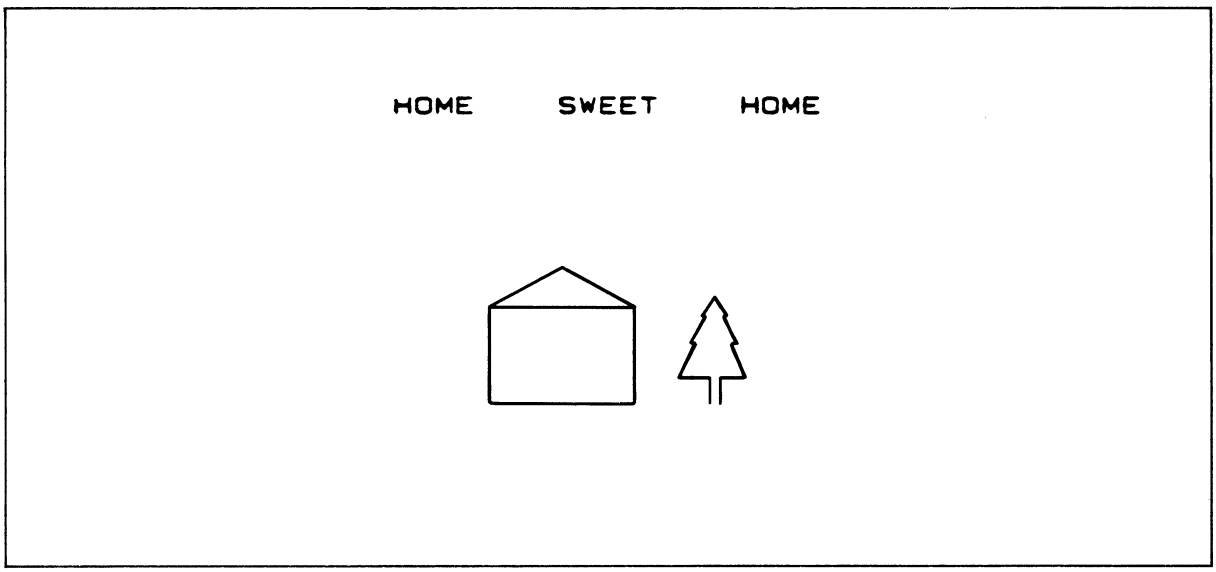


Figure 2-3. Use of ZTEXT

MARKERS

ZMARK (MARKNO) Generate a marker

Markers are used to identify individual points in the world coordinate system. Markers are center-oriented symbols which are output at the starting position. Depending on a particular display device's capabilities, DGL utilizes either software or hardware to generate the marker symbols.

All graphics display devices support 19 standard marker symbols. The marker used is specified by the value of MARKNO. MARKNO is an integer in the range 1-255. The 19 standard markers are defined as follows:

Table 2-1. Standard Markers

1 - . 2 - + 3 - * 4 - O 5 - X 6 - triangle	7 - rectangle 8 - diamond 9 - rectangle with cross 10 - 0 11 - 1 12 - 2	13 - 3 14 - 4 15 - 5 16 - 6 17 - 7 18 - 8 19 - 9

Besides these basic markers, each display device supports a device-dependent number of other markers that use any special marker symbols of that device (see the *Device Handlers Manual*). If the marker value specified is greater than the number of distinct markers supported by a device, then marker 1 (".") is used.

The orientation of a marker is centered around the starting position. The size of markers is device dependent and cannot be changed. The appearance of markers is affected only by the color and highlighting attributes. The starting position is unchanged after a call to ZMARK.

POLYGONS

ZPGDD (NPOINT,XVEC,YVEC,OPCODE)	Draw a polygon set(device-dependent)
ZPGDI (NPOINT,XVEC,YVEC,OPCODE)	Draw a polygon set(device-independent)

A polygon is a multisided, closed, plane figure. A polygon set includes one or more polygons, which may overlap. The DGL polygon routines draw the edge (the sides) of the polygon set, or the interior region bounded by the edge, or both.

A polygon set is defined in a single call to ZPGDD or ZPGDI and treated as a single polygon. The member polygons can intersect each other. The polygon set has one edge and one interior (see below) but either or both may be discontinuous.

The edge of a polygon set is the set of edges of its member polygons. The edge of a member polygon is the set of edge segments, or sides of that polygon. Since member polygons may be totally unconnected, the edge of a polygon set may be discontinuous.

The interior of the polygon set is the region or regions enclosed by the edge an odd number of times. A point which would be in the interior of a member polygon if it were considered by itself could be in the exterior of the polygon set. To decide whether a point is inside the set, draw a line (a ray) in any direction from that point. If the ray intersects the edge an odd number of times, the point is inside the polygon set. If the ray intersects the edge an even number of times, the point is outside the set.

This test is useful to see how DGL will fill the interior of a set consisting of concave, self-intersecting polygons, or multiple polygons. The test is also useful if member polygons overlap or enclose one another. Figure 2-4 shows several examples of polygon sets. Some have one member polygon (a,b,c), while the rest (d,e,f) have two or more.

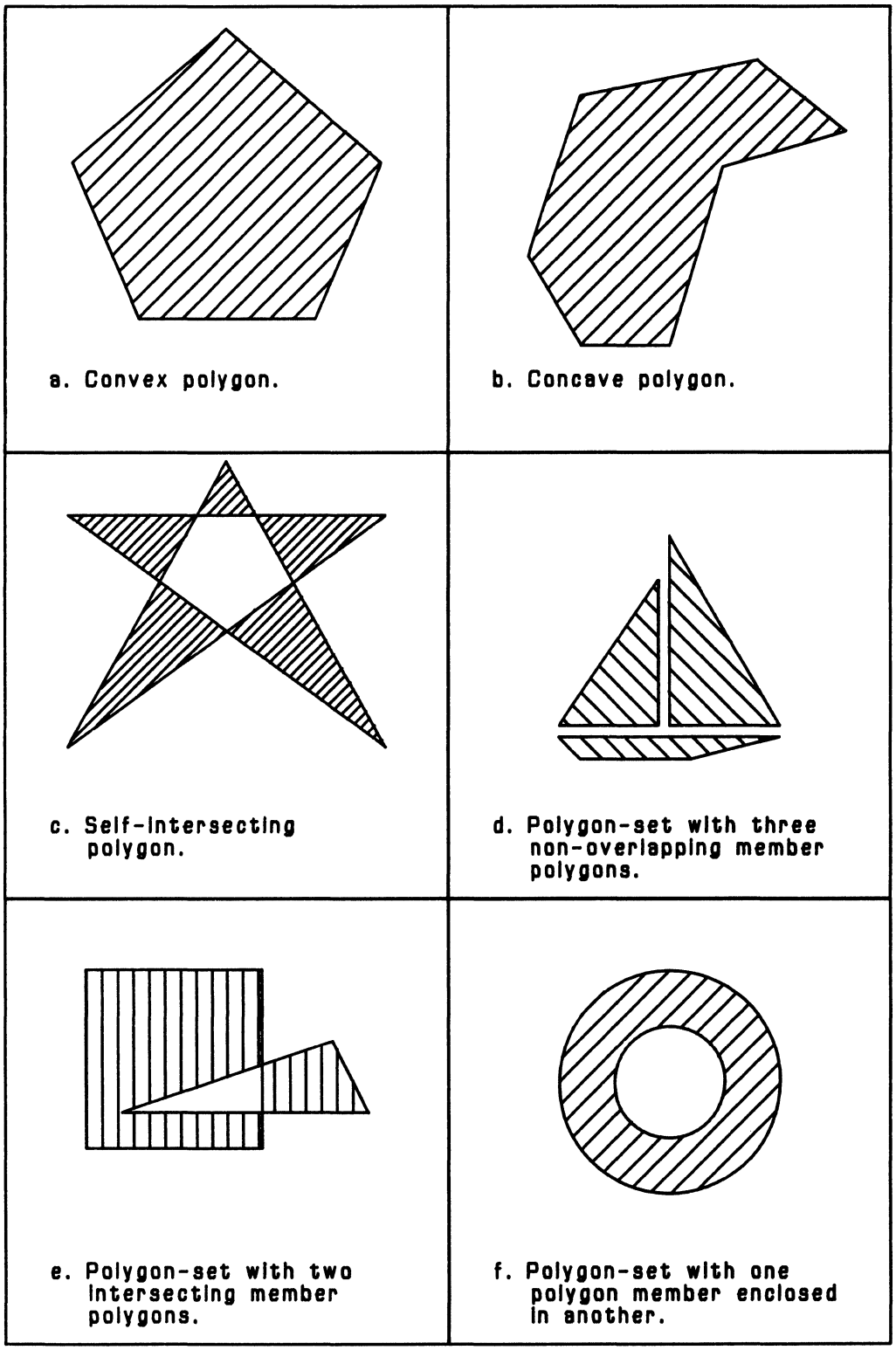


Figure 2-4. Examples of Polygon Sets

NPOINT specifies the number of vertices in the polygon set. XVEC and YVEC are arrays specifying the vertex locations in world coordinates. OPCODE is an array which holds opcodes corresponding to each vertex in the polygon set. The opcode for a vertex indicates whether it is the beginning vertex of a member polygon or whether it is a subsequent one. If it is a subsequent vertex, the opcode indicates whether the edge which connects this vertex with the previous one is a displayable edge or a non-displayable edge.

The first vertex specified in XVEC and YVEC must have the proper OPCODE value for the beginning vertex of a member polygon. ZPGDI and ZPGDD move to this first vertex and use the subsequent vertices to define the edge of the first member polygon until NPOINT vertices are processed or a vertex is reached which is marked as the beginning of a new member polygon.

In either case, if the last vertex of the member polygon is not the same as its first vertex, DGL adds a non-displayable edge to close the figure. If NPOINT vertices have not been processed, DGL processes the next member polygon in the same way. When the entire polygon set is finished, the starting position for the next primitive is at the first vertex of the last member polygon.

ZPGDD does a fast approximation to a polygon set in the polygon, but the results will vary with the capablities of the display device since ZPGDD makes use of its available polygon functionality. ZPGDD instructs the device to approximate visible polygon edges when the polygon style indicates that edges are to be displayed. If the device cannot generate any polygon edges, ZPGDD will simulate the functionality.

If filling is requested, ZPGDD instructs the device to fill the polygon set in a device-dependent approximation of the current polygon style. If the device cannot meet the request for fill at all, ZPGDD will not simulate fill functionality.

However, when no fill is output although requested, and the polygon style indicates that edges are not to be displayed, ZPGDD will generate an outline of the polygon set so that there will be some representative visible output. ZPGDD will generate this outline in the interior fill color and interior linestyle and will output only those edges which have an opcode indicating that they are displayable.

ZPGDI, by contrast, generates polygon sets with software; it will not use the polygon capabilities of the device unless it could do so accurately for the entire style range. ZPGDI creates polygon sets with reasonable accuracy and uniformity on all devices, but on some not as quickly as ZPGDD.

Alphanumeric Output

ZALPH (NCHARS, STRING) Output to the alphanumeric display

Alphanumeric strings can be used to prompt for operator input, give status reports, or output debugging messages. Because this output is not graphical, it is not affected by the graphics primitive attributes. A typical alphanumeric device is a terminal.

A call to ZALPH sends the first NCHARS number of characters contained in array STRING to the alphanumeric device. A carriage-return line-feed (CRLF) will normally be appended to STRING before it is sent to the device. If an underscore is the last character of the string sent out, the CRLF will be suppressed and the underscore will not be displayed.

If the alphanumeric display is physically the same device as the graphics display device, there are some restrictions as to what may be passed in STRING. Any escape codes which affect the graphics display should not be sent, since they may place the graphics display in an unknown state. The following program illustrates how ZALPH could be used to tell the operator that the graphics is complete.

```
      INTEGER MSG(17)
      DATA MSG/2HGr,2Hap,2Hhi,2Hng,2H o,2Hf ,2Hth,2He ,
     *         2Him,2Hag,2He ,2His,2H c,2Hom,2Hpl,2Het,2He /
           .
           .
           .
      CALL HOUSE          * Subroutine to generate the house
      CALL TREE           * Subroutine to generate the tree
           .
           .
C
C     Use ZALPH to tell the operator 'Graphing of the image is complete
C
      CALL ZALPH (34,MSG)
           .
           .
           .
```

PRIMITIVE ATTRIBUTES

Primitive attributes determine the appearance of the graphics output. Attributes are divided into two clases: attributes that apply to polygon interiors and attributes that apply to all other graphics primitives. As an example, the linestyle attribute specifies the pattern (such as solid or dashed) in which DGL draws lines. There are two separate linestyle attributes: one that applies to polygon interiors and another that applies to all other line segments.

Table 2-2 shows how attributes affect graphics primitives. Notice that highlighting affects all primitives, including polygon interiors. However, there are two distinct color attributes: the 'color' attribute that only affects primitives outside of polygons and a second color attribute, 'polygon color' that only affects polygon interiors.

Table 2-2. How Attributes Affect Graphics Primitives

ATTRIBUTE	PRIMITIVE					
	Line	Polyline	Marker	Text	Polygon Edge	Polygon Interior
Color	X	X	X	X	X	
Highlighting	X	X	X	X	X	X
Linestyle	X	X		X	X	
Linewidth	X	X		X	X	
Character Size				X		
Polygon color						X
Polygon linestyle						X
Polygon style						X

Attributes will be applied to each primitive according to the devices' capabilities. For instance, an HP 9872 plotter can apply linestyle to lines and polylines, but not to text. See the *Device Handlers Manual* for a description of device capabilities.

DGL maintains a table of values for each attribute. When DGL enables a graphics device, each table is initialized with default values suited to the device. The value of the current entry can be inquired, or a different entry chosen, as long as the graphics display device stays enabled. Except for the color and polygon-style tables, entries in the attribute tables cannot be redefined.

Color

ZCOLM (MODEL)	Choose color model
ZCOLR (COLOR)	Choose color for graphics primitives except polygon-set interiors
ZDCOL (COLOR,COLP1,COLP2,COLP3)	Redefine entry in color table
ZPICL (COLOR)	Choose color for polygon-set interiors

Every graphics device handler in DGL has its own default color table. For some devices, the size of the color table may be changed before program execution. Every default color table has at least an entry at index 1 and display devices for which there is a background color have an entry at index 0. Color 1 is the default color for all primitives and color 0 is the background color of the display device.

The user who does not wish to use the default color 1 for all primitives may choose other entries in the color table to be used instead. The color entry to be used for polygon interiors is chosen independently of the color entry to be used for all other primitives. When ZPICL is called the color entry specified is the color entry used to output subsequently defined polygon interiors. A call to ZCOLR specifies an entry that is applied in the same way to all other output primitives.

The user may redefine a color table entry using ZDCOL. A new color value for the entry specified in parameter COLOR is set according to the values of the parameters COLP1, COLP2 and COLP3. When the value of a color entry is redefined through ZDCOL, any subsequently output primitives which use that entry will reflect the newly defined value. The effect of redefinitions on previously output primitives is device dependent.

ZDCOL accepts color definitions in two color models, RGB and HSL. ZCOLM allows the user to specify which color model should apply to subsequent calls to ZDCOL. RGB is the default specification. When ZCOLM is called, it does not affect color definitions previously made using another model. When the value of a color table entry is inquired, it is returned in the current model, which may not be the model in which it was originally specified.

The RGB model, shown in Figure 2-5, is a color cube with the primary additive colors, red, green and blue, as its axes. Each point within the cube has a red intensity value (X coordinate), a green intensity value (Y coordinate) and a blue intensity value (Z coordinate). Each axis or intensity ranges from zero to one, and each point defines a unique color.

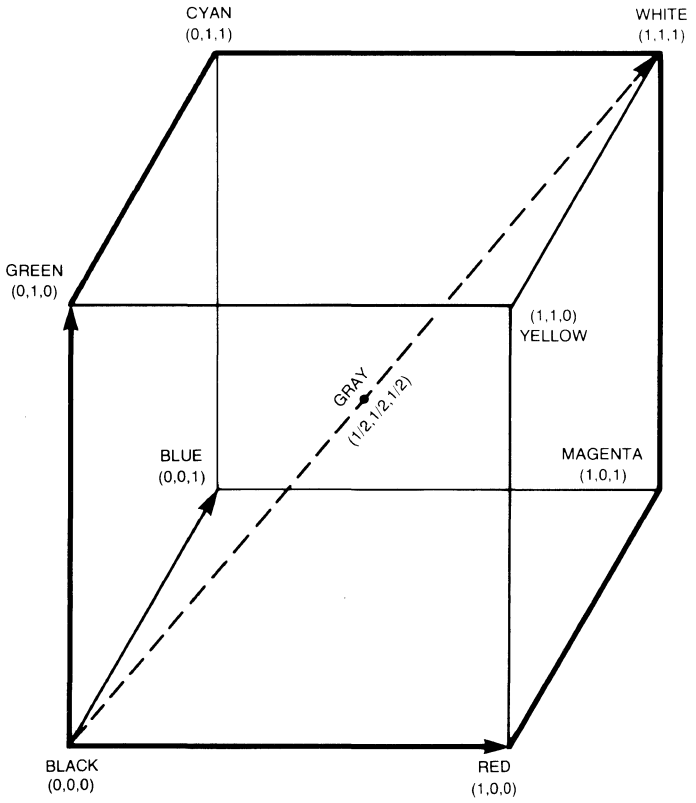


Figure 2-5. Red-Green-Blue Color Cube

The HSL model, shown in Figure 2-6, is a color cylinder in which:

- The angle about the axis of the cylinder, in fractions of a circle, is the hue.
- The radius is the saturation.
- The height is the luminosity (intensity or brightness per unit area).

Angle, radius and height all range from zero to one, and each point within the cylinder defines a unique color with a hue value, a saturation value, and a luminosity value.

The RGB model is most useful in applications where the primary colors are used. For example, in printed circuit board artwork layout, colors are used to distinguish between different material layers. Here the user does not care how bright the shade of red is, but simply that the color red is used.

The HSL model, however, is easier to use when objects are shaded to simulate three-dimensional curvature. In this case, conceptual features like saturation and luminosity enable the display of shaded polygons, taking into account the light sources, polygon characteristics, and the positions and orientations of the polygons and sources.

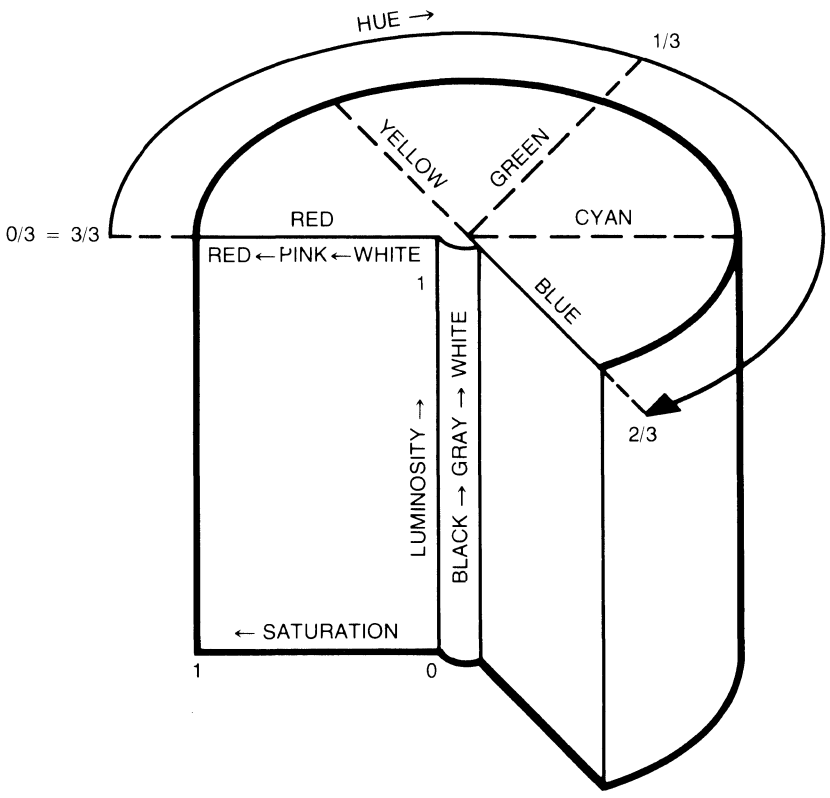


Figure 2-6. Hue-Saturation-Luminosity Color Cylinder

In summary:

to display a graphics primitive in a particular color may require any or all of the following actions:

- In the *Device Handlers Manual*, search the default color table of the graphics display device for a color index that will produce the desired color.
- If the desired color is not in the default table, call ZCOLM to switch to a different color model. Call ZDCOL to change the color definition assigned to an index.
- To specify the color of subsequent graphics primitives, call ZPICL for polygon interiors, ZCOLR for line primitives.

Highlighting

ZHIGH (HIGH)	Set highlighting

Highlighting is a means of emphasizing portions of an image. The DGL system provides a highlighting attribute to control the way that graphics output is displayed on the graphics device. A device may perform highlighting by blinking or intensifying certain output primitives. There is a maximum of 255 types of highlighting, but the actual number supported on the graphics display is device dependent. The mapping between the value of the highlighting attribute and the types of highlighting supported is also device dependent, with the exception of the value of 1 which never produces highlighting.

If highlighting is not supported on the selected graphics display device, the attribute value will be set to 1. The DGL display device initialization also sets the value of the attribute equal to 1.

Linestyle

ZLSTL (LSTYLE)	Choose linestyle for graphics primitives except polygon-set interiors
ZPILS (LSTYLE)	Choose linestyle for polygon-set interiors

Each graphics device handler in DGL has a fixed linestyle table, shown in the *Device Handlers Manual*. DGL maintains two indices to the table of the current device: one for the hatch lines used to fill polygon interiors, and one for other graphics primitives except markers. Markers are always drawn using solid lines.

Both linestyle indices are set to 1 at display device initialization. ZPILS chooses the index for the interiors of subsequent polygon sets. ZLSTL chooses the index for lines, polylines, graphics text characters, and the edges of polygon sets.

A linestyle is a recurring pattern of dots, blanks, short line segments and long line segments. In DGL, a linestyle can be drawn in any of three types: start-adjusted, continuous, or vector-adjusted. The available types depend on the device. On some devices, for instance, all linestyles are drawn in a continuous manner.

Start-adjusted linestyles always begin a vector with the first elements of the specified pattern. For example, if a pattern starts with two dots, each vector drawn will start with two dots. Likewise, if the vector starts with five blank spaces and then a dot, each vector drawn will start with five blank spaces. In the second example, if the vectors are short, the pattern may not be displayed at all if the vector is not longer than the section of blanks. This type of linestyle will usually degrade when attempting to draw smooth curves with many small line segments.

Continuous linestyles start the pattern with the first vector, but subsequent vectors will be continuations of the pattern. Thus it may take several vectors to complete one cycle of the pattern. This type is useful for drawing smooth curves, but does not necessarily reproduce either endpoint of the vector. As with start adjusted linestyles, if a vector is small enough, it might be composed only of the space between points or dashes in the pattern. In this case, the vector may not be displayed at all.

With vector-adjusted linestyles, each vector is treated individually. Individual treatment guarantees that a solid component of the pattern will be generated at both ends of the vector. Thus, the endpoints of each vector will be clearly identifiable. Linestyle integrity degenerates with very small vectors. Since some component of the pattern must appear at both ends of the vector, a short vector will often be drawn as a solid line.

Figure 2-7 illustrates how one pattern would be displayed using each one of the different linestyle type.

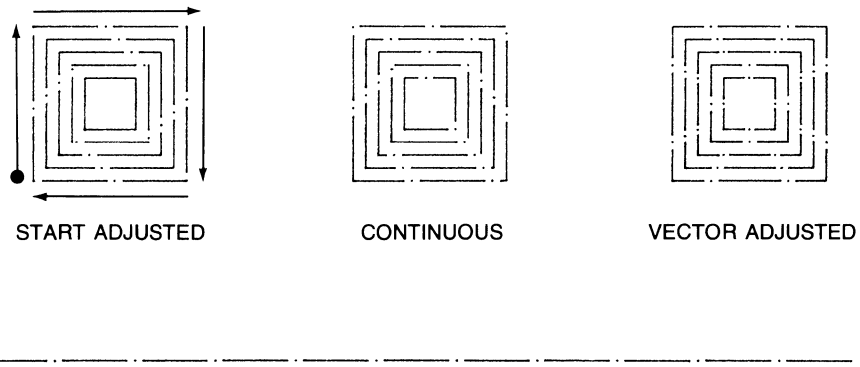


Figure 2-7. Types of Linestyles

Linewidth

ZLWID (LWDTH) Set the linewidth

A programmer may output lines of different widths by changing the linewidth attribute. A call to subroutine ZLWID sets the linewidth attribute for subsequent lines, polylines, polygon edges, and graphics text characters. Markers are not affected by this call because they are defined to always be output using the thinnest linewidth supported on the graphics display device.

All devices support at least one linewidth. The number of supported widths is device dependent. A value of 1 always specifies the thinnest width possible. When multiple widths are supported, the width of the line increases as the value of LWDTH does, until the device-supported maximum is reached. If LWDTH is greater than the number of line widths supported by the graphics display, or if LWDTH is less than 1, then the linewidth will be set to the thinnest available width. The default value of LWDTH is 1.

Character Size

ZCSIZ (WIDTH, HEIGHT) Set character size

The DGL system provides a character size attribute to control the size of graphics text generated. A subroutine call to ZCSIZ defines the desired width and height of the character cell in which characters are positioned. Actual sizes of text which can be generated on the graphics display are device dependent, as is the placement of text within the character cell.

The desired width and height of the cell are specified in world coordinate system units. If the requested character size is not available on the graphics display, then the "smaller best fit" character size will be used. See Figure 2-8 for the different text widths and heights available. The definition of the "smaller best fit" character size is as follows:

1. The largest character whose cell height is less than or equal to the requested height and whose cell width is less than or equal to the requested width, or,

2. If the minimum hardware size does not meet criterion 1, the minimum character size is used.

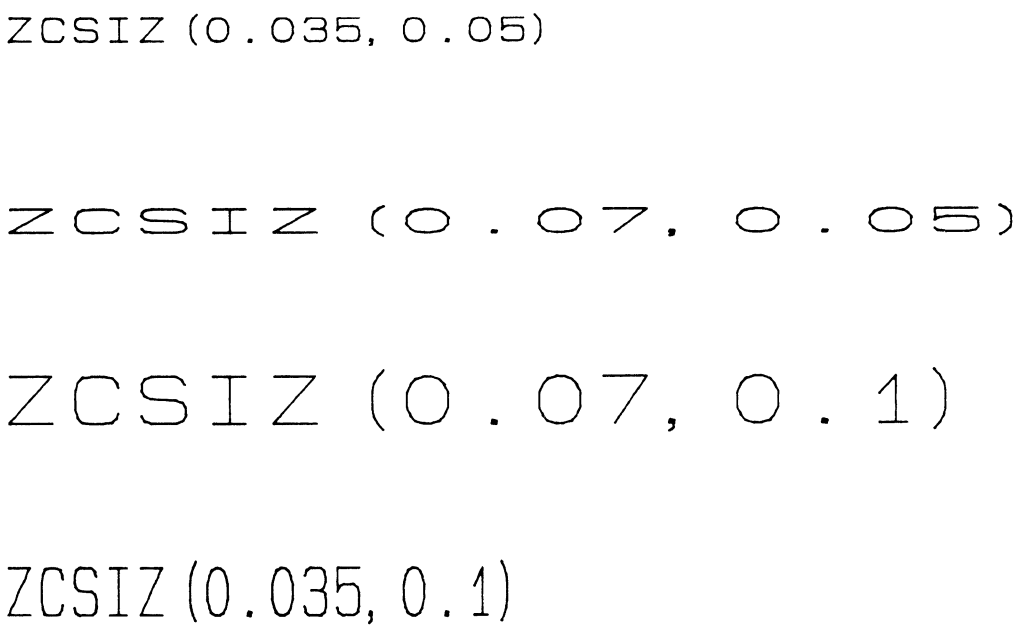


Figure 2-8. Different Text Widths and Heights

Polygon Style

ZDPST (PINDEX,DENSTY,ORIENT,EDGE)	Redefine entry in polygon style table
ZPSTL (PINDEX)	Choose polygon style

Polygon style has three components:

- DENSTY is the density of fill lines used to fill the interiors of polygon sets. Density ranges from no fill to solid fill. A positive density means fill lines in one direction only (all parallel); a negative density means perpendicular cross-hatching.

- ORIENT is the angle of interior fill lines to the horizon, ranging from -90.0 degrees to +90.0. For cross-hatching, the angle refers to the first set of hatch lines, with the second set perpendicular to the first.

- EDGE defines whether the segments of the edge of a polygon set which are defined as "draw" segments (in ZPGDI or ZPGDD) should be displayed. Edge segments specified as "move" segments are never displayed. Displayed edge segments are drawn using the entries specified by ZCOLR, ZLSTL and ZLWID.

Every graphics device handler in DGL has a default polygon style table. The number of entries in the table and their values depend on the graphics device. Default definitions are designed to suit each graphics device. The first 16 styles are the same for all devices (see Figure 2-9). For some devices, entries can be added to or deleted from the table before program execution.

When ZPSTL is called, specifying an index into the polygon table, the polygon style defined by this entry overrides any previously chosen polygon style and is then the polygon style used to output subsequently defined polygon sets.

The user may redefine an entry in the polygon style table using ZDPST. When the value of a polygon style entry is redefined through ZDPST, any subsequently output polygon sets which use that entry will reflect the newly defined value.

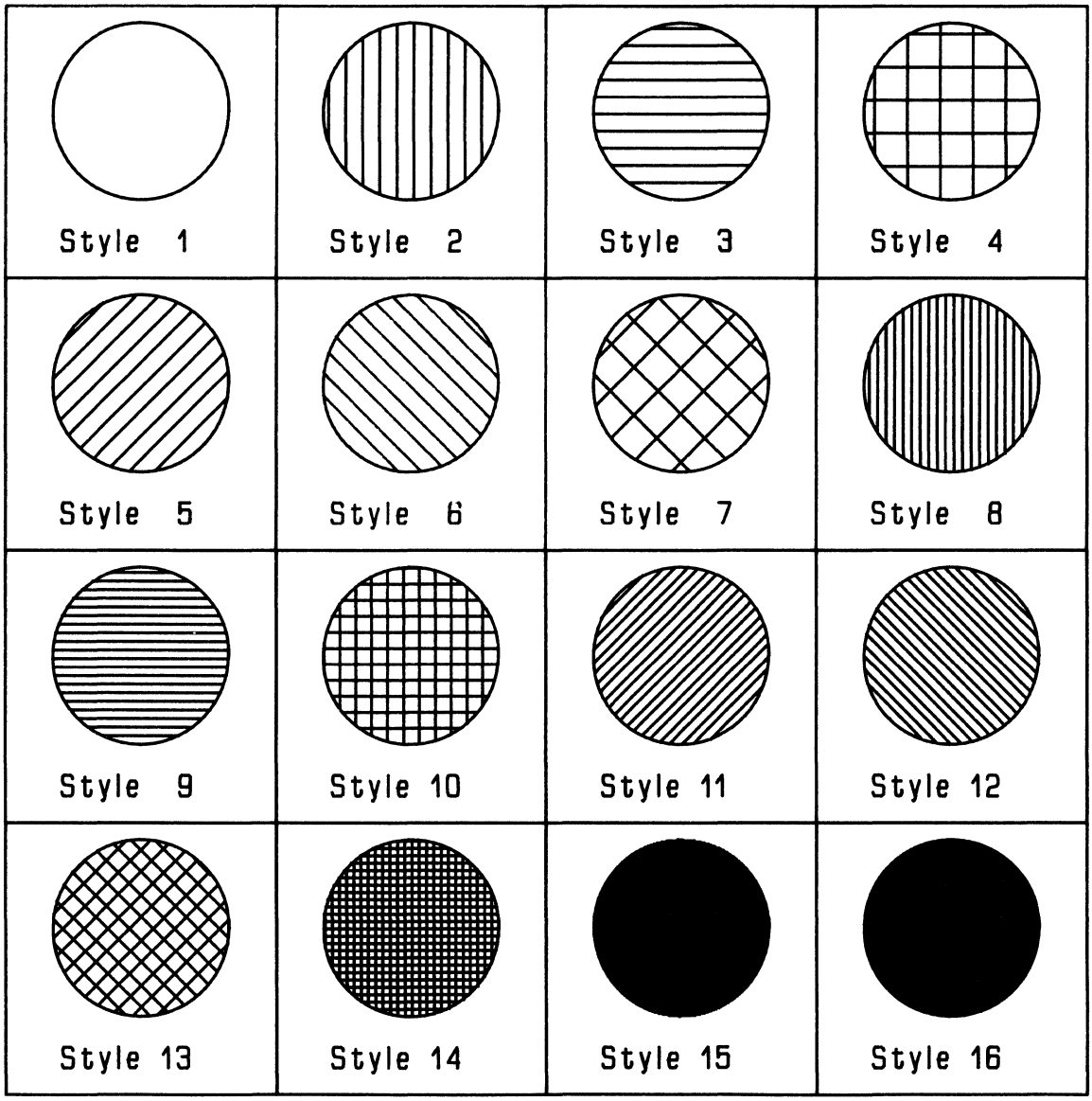


Figure 2-9. Default Polygon Styles

VIEWING TRANSFORMATIONS

General

A viewing transformation is the method by which objects are turned into viewable images. DGL's viewing transformations modify objects for viewing by transforming them from the units in which they were defined to device-dependent units (see Figure 2-10).

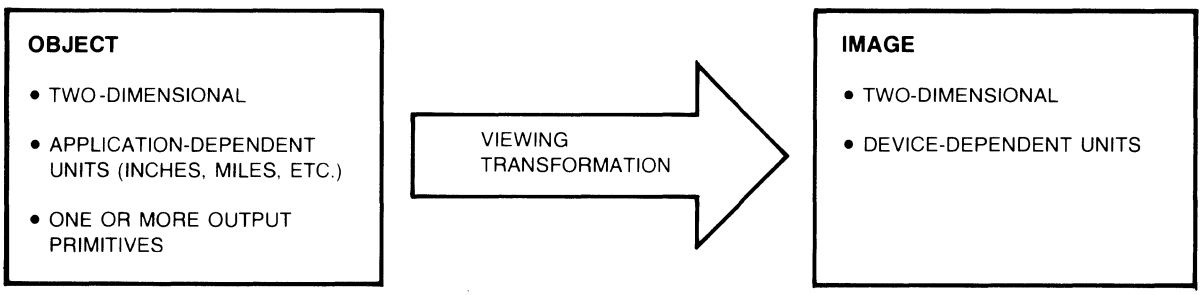


Figure 2-10. The Viewing Transformation

Graphics primitives are used to create two-dimensional objects in the world coordinate system. A number of concepts are important in understanding how DGL turns two-dimensional objects in the world coordinate system into images on a display device. The DGL viewing transformation is summarized in Figure 2-11.

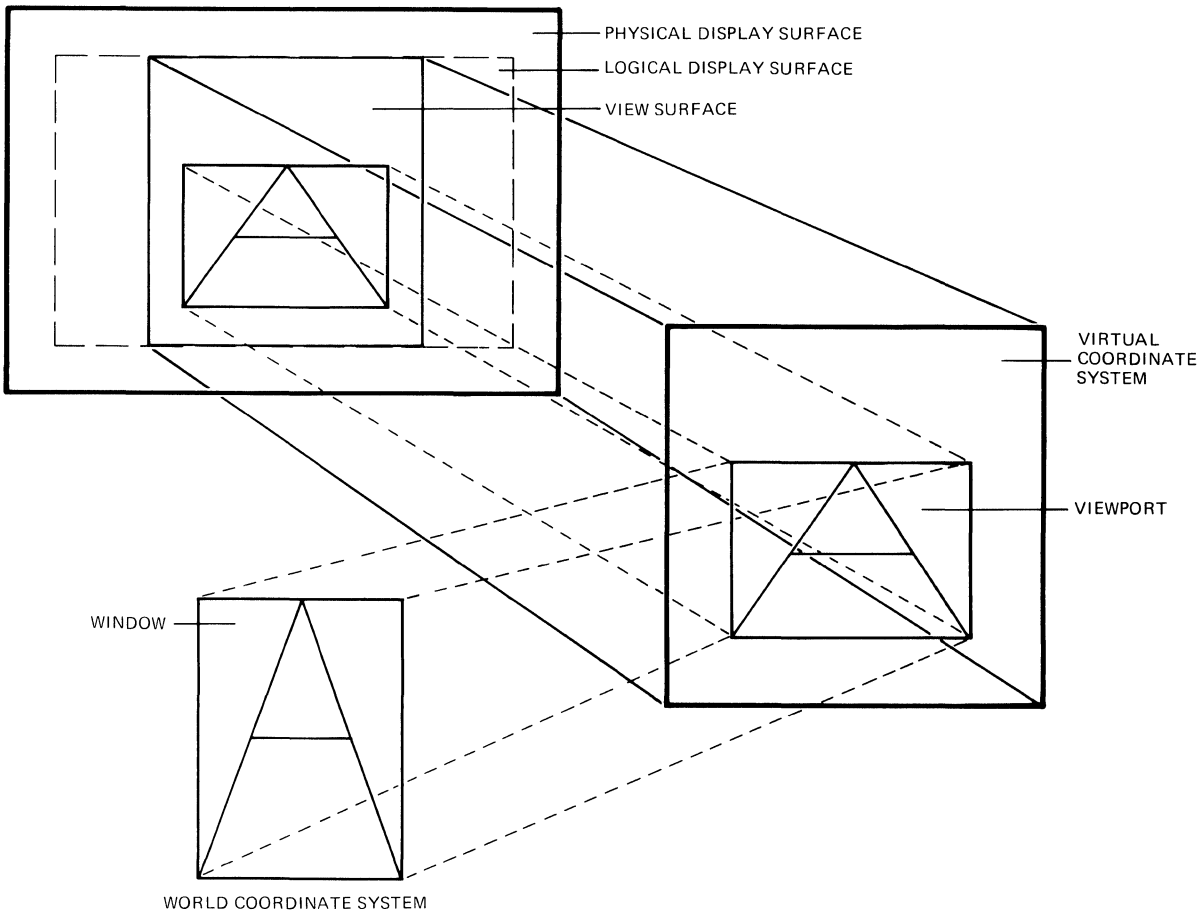


Figure 2-11. The DGL Viewing Transformation Process

Three questions must be addressed when defining the DGL viewing transformation:

1. What portion of the world coordinate system will be used to define and display the object (where is the *window*)?

2. Where will the window be mapped in the virtual coordinate system (where is the *viewport*)?

3. Where will the virtual coordinate system be placed on the physical display surface (what are the *logical display limits*)?

The window defines the portion of the world coordinate space to be used. This provides the application program with the flexibility of working in units that are relevant to the application. If the elements of an object extend beyond the limits of the window, the resulting image may appear to run off the edges of the display surface, since software clipping is not done by DGL. This phenomenon is not well defined because out-of-range world coordinate data is handled in a device-dependent manner. Refer to the *Device Handlers Manual* to see how a particular device handles this problem.

The user also has control over the virtual coordinate system, the two-dimensional system whose units range from 0.0 to a maximum of 1.0. Since many display devices do not have the same aspect ratio, the placement of the virtual coordinate system on the logical display surface requires planning. If the virtual coordinate system were always a unit square, a non-square logical display surface would not be entirely available. To make full use of a non-square logical display surface, DGL allows the aspect ratio of the virtual coordinate system to be changed as needed.

The size and shape of the logical display surface is also under the user's control. Logical display limits allow a subset of a display device to be used for output. The bed of a plotter might be 500 millimeters by 1000 millimeters, while the paper used in it might be only 250 millimeters square. The logical display limits of the plotter can be set to that size, so that, from that point on, DGL would treat the plotter as if it were physically 250 millimeters square.

The logical display limits determine the size and placement of the view surface. The view surface's dimensions and placement are determined by:

1. The aspect ratio of the virtual coordinate system, and

2. The logical display limits of the display device.

Once the view surface is determined, a portion of it is designated to be the *viewport*. An application might use several viewports to show different views of the same object. Since the window is mapped directly onto the viewport, distortion may result if the window does not have the same aspect ratio as that of the viewport. This type of distortion is illustrated in Figure 2-12.

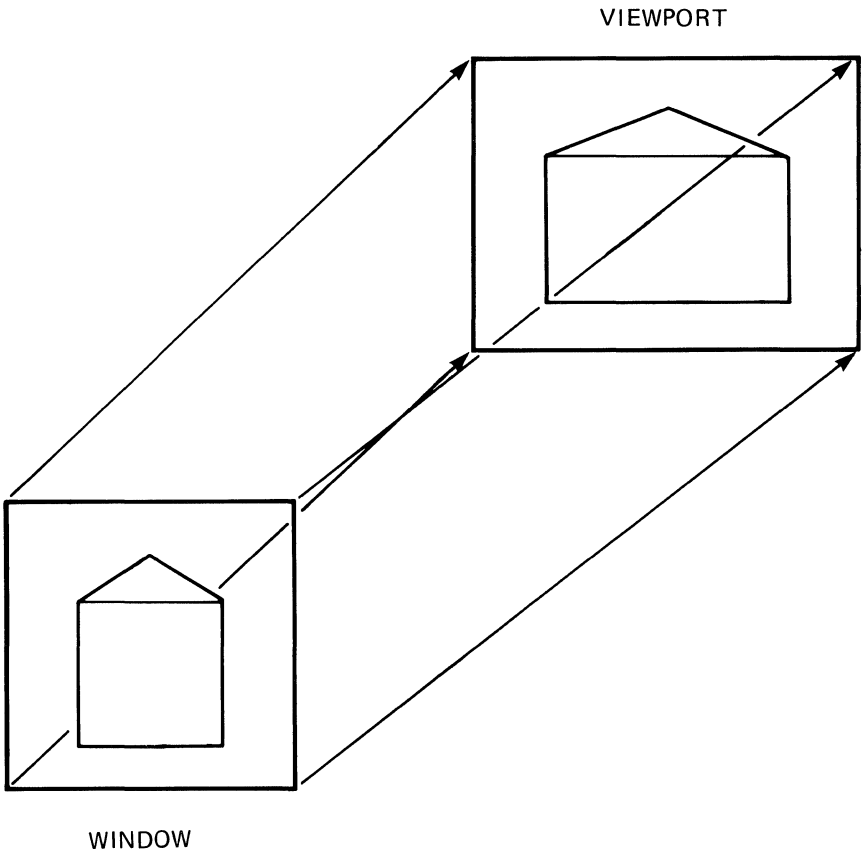


Figure 2-12. Distortion in the Window to Viewport Mapping

Logical Display Limits

ZDLIM (XMIN,XMAX,YMIN,YMAX,IERR) Set logical display limits

ZDLIM specifies a subset of a physical display surface to be used for graphics output. ZDLIM defines the logical display limits. The limits of this area are expressed in terms of millimeters offset from the physical origin of the device. The location of the physical origin of a display device is device dependent (see the *Device Handlers Manual*).

The virtual coordinate system is mapped onto the largest region within the logical display limits which has the aspect (height to width) of the virtual

coordinate system. Therefore, the displayed image will not be distorted; it will merely be scaled smaller or larger.

Aspect Ratio

ZASPK (WIDTH, HEIGHT) Set aspect ratio of virtual coordinate system

ZASPK sets the aspect ratio of the virtual coordinate system (and hence the aspect ratio of the view surface) to be HEIGHT divided by WIDTH. A ratio of 1.0 defines a square virtual coordinate system; a ratio greater than 1.0 specifies it to be higher than it is wide; and a ratio less than 1.0 specifies it to be wider than it is high. Since WIDTH and HEIGHT are used to form a ratio, they may be expressed in any units (as long as they are in the same units).

DGL calculates the range of the coordinates of the virtual coordinate system based on the value of the aspect ratio as shown in Table 2-3. The coordinates of the longer axis are always set to range from 0.0 to 1.0, and those of the shorter axis from 0.0 to a value that achieves the specified aspect ratio. Thus, ZASPK also defines the limits of the virtual coordinate system.

Table 2-3. Virtual Coordinate Limits

Aspect Ratio (AR)	X Limits	Y limits
AR < 1.0	0.0, 1.0	0.0, 1.0*AR
AR = 1.0	0.0, 1.0	0.0, 1.0
AR > 1.0	0.0, 1.0/AR	0.0, 1.0

When a call to ZASPK is made, the DGL system sets the viewport equal to the limits of the virtual coordinate system. This call can therefore be used to access the entire logical display surface. A program could display an image on the entire logical display surface in the following manner:

```
      .
      .
INTEGER ILIST, IERR
REAL MAXS(2), WIDTH, HEIGHT
      .
      .                                        * Get the dimensions
CALL ZIWS (253, 0, 2, ILIST, MAXS, IERR)       * of the display
                                               * surface.

WIDTH = MAXS(1)                          *Width of the physical display
HEIGHT = MAXS(2)                         *surface. Height of the physical
CALL ZDLIM(0,WIDTH,0.,HEIGHT,IERR)       *display surface. Set logical
                                         *limit to their maximum.

                                         *Set the aspect ratio of the
CALL ZASPK(WIDTH,HEIGHT)                 *virtual coordinate system to
      .                                  *the maximum aspect ratio of
      .                                  *the display surface.
      .
```

The initial aspect ratio of the virtual coordinate system is 1.0, meaning the virtual coordinate system is a unit square. This produces a view surface that is the largest inscribed square within the logical display limits. By changing the aspect ratio, the view surface defines the largest inscribed *rectangle* within the logical display limits. The placement of the view surface is dependent upon the device being used. It is generally centered on CRT displays and is usually placed in the lower left-hand corner of plotters. Refer to the *Device Handlers Manual* to determine the placement of the view surface on the logical display surface of a particular display device.

Viewport

ZVIEW (VXMIN,VXMAX,VYMIN,VYMAX) — Set the viewport

ZVIEW sets the limits of the viewport in units of the virtual coordinate system. The viewport must be within the limits of the virtual coordinate system. The initial viewport is:

(VXMIN=0.0, VXMAX=1.0, VYMIN=0.0, VYMAX=1.0)

This initial viewport is mapped onto the maximum visible square within the logical display limits. This area is called the view surface.

By changing the limits of the viewport, an application program can display an image in several different positions on the same display surface. Figure 2-13 illustrates how an image could be displayed sequentially in the four corners of a display surface.

```
REAL AR(2), VMAXX, VMAXY
INTEGER ILIST, IERR

CALL ZIWS (254,0,2,ILIST,AR,IERR)       * Get the aspect ratio of the
                                        * logical display surface.

VMAXX = 1.0                             * Determine the virtual
VMAXY = 1.0                             * coordinate bounds given
IF (AR(2).LT.1.0) VMAXY = AR(2)         * the maximum
IF (AR(2).GT.1.0) VMAXX = 1.0/AR(2)     * aspect ratio:
VMIDX = VMAXX/2.0                       * middle of X range
VMIDY = VMAXY/2.0                       * middle of Y range

CALL ZASPK(1.0,AR(2))                   * Set the aspect ratio of the
     .                                  * virtual coordinate system to
     .                                  * the aspect ratio of the
     .                                  * display surface.
     .                                  * -- Viewport 1 --
CALL ZVIEW(0.0,VMIDX,VMIDY,VMAXY )      * Set viewport to upper left-
CALL HOUS(XHOUS,YHOUS)                  * hand corner and draw the
CALL TREE(XTREE,YTREE)                  * house and tree.
     .
     .
     .                                  * -- Viewport 2 --
CALL ZVIEW(VMIDX,VMAXX,VMIDY,VMAXY)     * Set viewport to upper right-
CALL HOUS(XHOUS,YHOUS)                  * hand corner and draw the
CALL TREE(XTREE,YTREE)                  * house and tree.
     .
     .
     .                                  * -- Viewport 3 --
CALL ZVIEW(0.0,VMIDX,0.0,VMIDY)         * Set viewport to lower left-
CALL HOUS(XHOUS,YHOUS)                  * hand corner and draw the
CALL TREE(XTREE,YTREE)                  * house and tree.
     .
     .
     .                                  * -- Viewport 4 --
CALL ZVIEW(VMIDX,VMAXX,0.0,VMIDY)       * Set viewport to lower right-
CALL HOUS(XHOUS,YHOUS)                  * hand corner and draw the
CALL TREE(XTREE,YTREE)                  * house and tree.
```

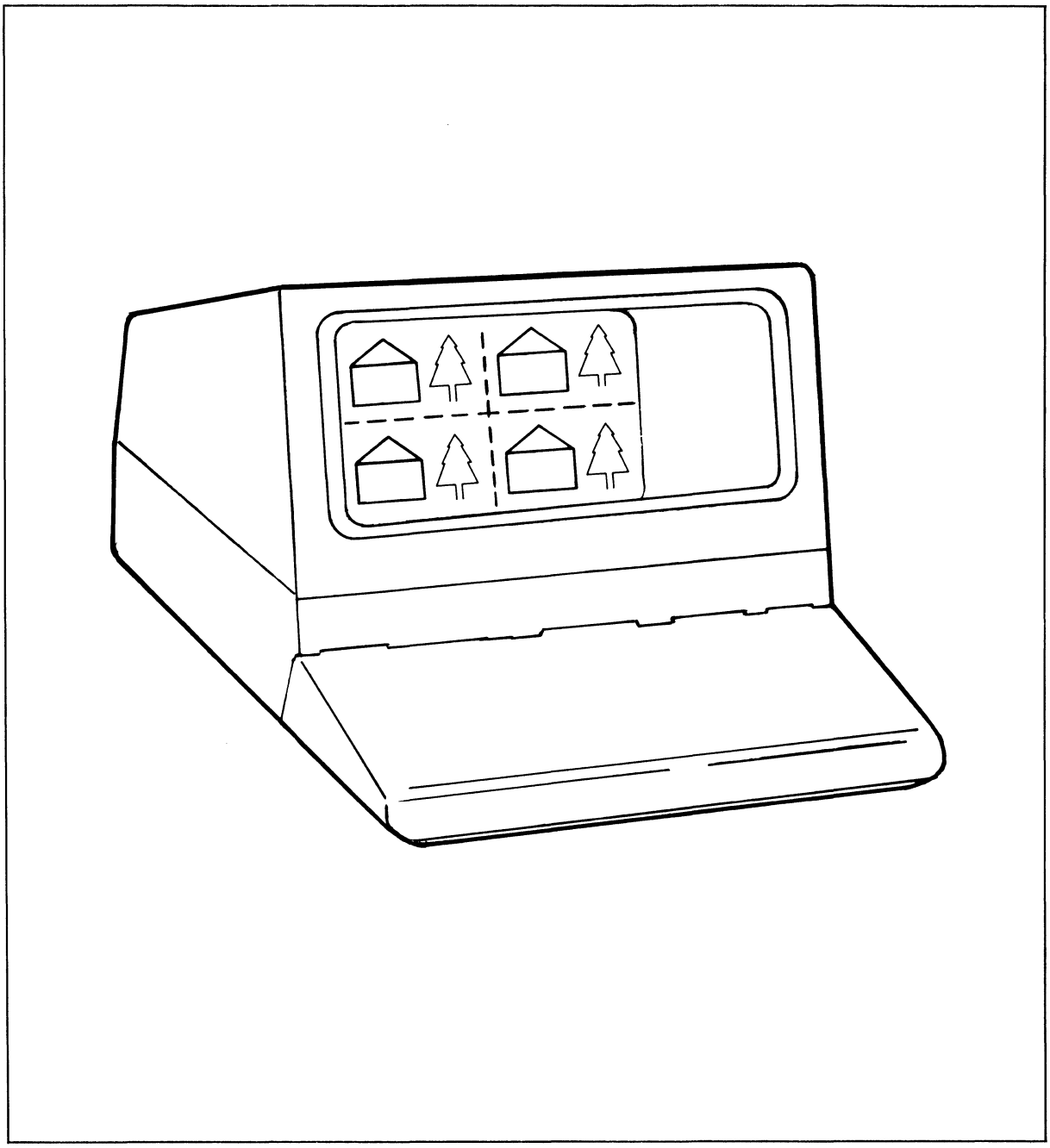


Figure 2-13. Multiple Viewports

Window

ZWIND (WXMIN,WXMAX,WYMIN,WYMAX) Set the window

ZWIND specifies the portion of the world coordinate system that maps onto the viewport. Setting the window allows the application program to define which portion of the world coordinate space is to be viewed. This provides the application program with the flexibility of working in units that are relevant to the application. Since the window is in the same coordinate space in which objects are defined, the bounds of the window can affect the size of the image displayed. The larger the limits of the window, the smaller the object's image. If, however, the window is specified as having smaller limits than the object, unpredictable, device-dependent results occur, since the DGL system does not perform software clipping.

The window is defined in units of the world coordinate system. The window's aspect ratio should be the same as the aspect ratio of the viewport if distortion is not desired. In general, setting the window by calculating its dimensions as a function of the viewport is a good way to prevent distortion. In this way, no assumption is made about a previously set viewport, helping to ensure that the visual results will be those desired.

The initial window has the following limits, expressed in world coordinate units:

(WXMIN=-1.0, WXMAX=1.0, WYMIN=-1.0, WYMAX=1.0)

The default aspect ratio of the window is 1.0, which is the same as the aspect ratio of the initial viewport.

Conversion of Units

ZDPMM (WX,WY,MMX,MMY) Convert a world coordinate point to millimeters for the physical graphics display device.

The DGL system provides the user with the capability to determine where a coordinate point in the world coordinate system will be displayed on a graphics display surface. A subroutine call to ZDPMM returns the location that the world coordinate system point (WX,WY) is mapped to on the graphics display. The coordinate returned in (MMX,MMY) is expressed as the millimeters offset from the origin of the physical graphics device. The location of the origin is device dependent. Refer to the *Device Handlers Manual* for the position of the origin on a specific device.

INPUT PRIMITIVES

General

The DGL system has a class of functions which enable a user to interact with an application program. These functions, or input primitives, provide a method for receiving input. Each primitive is implemented by an input device. A DGL application program can receive input from the following devices:

1. Button - returns an integer
2. Keyboard - returns alphanumeric text
3. Locator - returns an (X,Y) point
4. Valuator - returns a real value between 0.0 and 1.0

Data can be obtained from these devices in two different ways: *requesting* and *sampling*. A device is said to request data when it waits for an operator response, such as striking the carriage return, before returning information. Data can be requested from every input device. Sampling means that the current value of the device is returned without waiting for any operator response. Sampling may only be performed on locator and valuator devices.

One DGL application program can receive input from at most one of each of the four input devices. Some input devices are not supported directly by a physical device, but are simulated by the DGL software. As an example, the valuator device is simulated on the HP 2623 Graphics Terminal by using either the X or Y coordinate of the graphics cursor.

Another important feature is echoing. Echoing gives an operator instantaneous feedback during an input operation. Some echoes reflect the status of the input operation so as to allow refinement before completion. A familiar example of echoing is the characters that appear on a terminal screen when a key is pressed. In most systems, these characters have actually been echoed by the computer after it has received them from the keyboard. By seeing what is typed, an operator may correct any typographical errors before terminating the input operation by pressing return.

The type of echoing depends on the type of input being requested and the properties of the device being used. The echo itself begins when the input operation is initiated and lasts only until the operation is terminated.

There are two types of echoes: those on the input device itself and those on the graphics display device. All input functions may be echoed on the input device (depending, of course, on the device's echoing capabilities). Locator input may also be echoed on the graphics display device. These echoes begin at a user-defined point called the locator echo position. Some display device echoes also utilize this point throughout the echo. For

example, when a rubber band line echo is used with locator input, the fixed end of the rubber band line begins at the locator echo position.

Button

ZBUTN (ECHO, BUTTON) Request button input

The button device returns an integer value associated with the button pressed. When ZBUTN is called the application program waits until the user presses a button and an integer is returned in BUTTON. The integer ranges from 1 to a device-dependent maximum (it will never be greater than 255). Zero will be returned if an invalid button is activated.

Typical button devices are the numeric keys of an interactive keyboard or special function keys. Echoing is specified by the value of ECHO. A value of 0 indicates that no echo should be performed. Possible echoes include turning on a light or beeping when a button is activated.

```
      INTEGER BUTTON                    * Buttons are always integers.
        .
        .
   50 CONTINUE
      CALL ZBUTN(0,BUTTON)              * Branch based on the value of
      IF (BUTTON.EQ.1) GOTO 100         * the button.
      IF (BUTTON.EQ.2) GOTO 200
      IF (BUTTON.EQ.3) GOTO 300         * If out of range,
      GOTO 50                           * try again.
        .
        .
        .
```

Keyboard

ZKYBD (ECHO, MAX, ACTUAL, STRING) Request keyboard input

The keyboard device returns an alphanumeric text string to the calling program. A call to ZKYBD waits until a line of text has been entered on the keyboard device and is terminated (e.g. carriage return) before the text is returned in the array STRING. The characters are returned in Packed ASCII format. The termination character is not returned in STRING. A MAX number of characters may be entered, with the restriction that MAX be less than or equal to 132. The actual number of characters entered is returned in ACTUAL.

Echoing, as specified by the ECHO parameter, depends upon the capabilities of the physical device being used. In all cases, 0 indicates that echoing should not be performed. Possible echoes include sounding a bell or displaying the input as it is entered. Refer to the *Device Handlers Manual* for echoes supported by a particular device.

```
INTEGER ACTUAL,STRING(20)          * 40 characters
      .
      .
      .
CALL ZKYBD(1,40,ACTUAL,STRING)     * Request up to 40 characters
CALL ZTEXT(40,STRING)              * with echo 1 and output them
      .                            * on the graphics display.
      .
      .
```

Locator

GENERAL

The locator device returns an (X,Y) point in the world coordinate system. This position may either be sampled or requested. Typical locator devices are digitizers and graphics cursors on CRTs. At least one type of echoing can always be performed. The number and type of echoes available is dependent upon the device being used, as well as the method of inputting data.

LOCATOR LIMITS

ZLLIM (XMIN, XMAX, YMIN, YMAX, IERR)	Set the locator limits.
ZLPMM (WY, WY, MMX, MMY)	Convert a world coordinate point to a point on the physical locator device.

Just as a portion of the display surface was selectable for graphics output, portions of the locator surface may be selected for input. This is done by setting the limits of the locator surface. The limits of the locator are initially defined as the addressable area of the locator device. ZLLIM allows the user to set these limits to a different region on the locator surface. The pairs (XMIN,YMIN) and (XMAX,YMAX) define the corner points of this rectangle in terms of millimeters offset from the origin of the device. (The exact position of the device origin is device dependent and is documented in the *Device Handlers Manual*.)

ZLLIM does not affect the virtual coordinate system. It only affects the mapping from the virtual coordinate system to the locator surface.

If the locator and the graphics display are both the same physical device (i.e., HP 2623 display and HP 2623 cursor), then the locator limits and the display limits must be identical. Specifically, when they refer to the same surface, the interaction between ZDLIM and ZLLIM is as follows:

1. The locator limits are initialized to the same values as the logical display limits.

2. A call to ZDLIM implicitly calls ZLLIM with the same values.

3. Explicit calls to ZLLIM are ignored.

The logical locator limits always map directly to the view surface, therefore, distortion may result in the mapping between the logical locator and the display when the logical locator limits and the view surface have different aspect ratios. If this distortion is not desired it can be avoided by assuring that the logical locator limits maintain the same aspect ratio as that of the view surface.

ZLPMM converts a world coordinate point to millimeters on the locator device. (WX,WY) can be any world coordinate point. It does not have to be in the window or map to a point on the true physical limits of the locator. (MMX,MMY) is expressed in millimeters offset from the origin of the locator device. The position of this origin is device dependent. Refer to the *Device Handlers Manual* for its location on a particular device.

SAMPLING THE LOCATOR

ZSLOC (ECHO,WX,WY) Sample locator and report value

ZSLOC returns the current position of the locator in world coordinates (WX,WY) without waiting for an operator request.

The number of echoes supported by a device and the relationship between the value of ECHO and the echo performed is device dependent (refer to the *Device Handlers Manual*). A value of 0 always specifies that echo should not be performed.

REQUESTING LOCATOR INPUT

ZWLOC (ECHO, LBUTN, WX, WY) Request locator input, report value
ZLOCP (WX, WY) Set the locator echo position

A call to ZWLOC causes the application program to wait until a locator button is pressed. The value of the selected button and a world coordinate point (WX,WY) are then returned to the calling program. Each locator device has its own set of buttons which may or may not be the same as those that comprise the button device. If an invalid button is pressed, LBUTN will be returned as 0; otherwise, LBUTN will contain the value of the button activated.

Several different types of echoing can be performed. Some echoes are performed only on the locator device. Possible echoes include blinking a light or sounding a bell each time a point is entered. The number and relationship between the value of ECHO and the echo performed is dependent upon the locator device being used (refer to the Device Handlers Manual).

Some locator echoes are performed on a graphics display device. All locator echoes on the graphics display begin at a world coordinate point called the locator echo position. Some echoes may also use the locator echo position as a reference point. For example, many devices support a rubber band line echo. The fixed end of the rubber band line will be at the locator echo position. ZLOCP sets the value of this world coordinate point to (WX,WY). Note that the locator echo position is not relevant to ZSLOC, only to ZWLOC.

The point (WX,WY) specified in the ZLOCP call must be displayable to be able to use echoing. Therefore, it cannot be specified outside the current world coordinate limits or an error will result and the call will be ignored. The default locator echo position is in the center of the window. When a call is made which changes either the viewing transformation or the mapping between the display surface and locator surface, the locator echo position is reset to its default value. The calls which do this are ZASPK, ZDLIM, ZLLIM, ZDINT, ZLINT, ZWIND and ZVIEW.

The following program section shows a use of these calls. The locator device is used to let the operator set the locator limits interactively by selecting the lower left corner first, then the upper right corner.

```
          .
          .
          .
 CALL ZWLOC(0,LBUTN,WXMIN,WYMIN)          *Request locator input with no
                                          *echo.
 CALL ZLOCP(WXMIN,WYMIN)                  *Set the locator echo position.
 CALL ZWLOC(4,LBUTN,WXMAX,WYMAX)          *Request locator input using
                                          *rubber band echo.
 CALL ZLPMM(WXMIN,WYMIN,MXMIN,MYMIN)      *Convert both points to
 CALL ZLPMM(WXMAX,WYMAX,MXMAX,MYMAX)      *millimeters on the locator.
 CALL ZLLIM(MXMIN,MXMAX,MYMIN,MYMAX,IERR) *Set the locator limits to them.
          .
          .
          .
```

Valuator

GENERAL

A valuator returns a single real value ranging from 0.0 to 1.0. Some valuator devices are composed of multiple subvaluators. The subvaluator specifies which valuator will be used. For example, on the HP 2623 Graphics Terminal, subvaluators can be simulated by using either the X coordinate or the Y coordinate of the graphics cursor as the valuator. If the subvaluator equals 1, the X coordinate of the cursor is returned with 0.0 representing the extreme left side of the display and 1.0 representing the extreme right side. If the subvaluator equals 2, the Y coordinate of the cursor is returned, with 0.0 represented as the bottom of the display and 1.0 represented as the top of the display.

Echoing can be performed on the valuator device when either sampling data or requesting data. The number and type of supported echoes are device dependent, but the majority of valuators support at least one type of echoing. Possible echoes include beeping and displaying the value each time the valuator is sampled. A value of 0 for the echo specifies that echoing will not be performed.

SAMPLING A VALUATOR

ZSVAL (ECHO,SUBVAL,VALUE) Sample the valuator and report value

ZSVAL returns the current value of the valuator in the variable VALUE without waiting for operator response. VALUE will always range from 0.0 to 1.0. The program specifies the desired echo in ECHO and the subvaluator in SUBVAL. Valuators are sampled in applications that require changing some attribute of the program in real time.

REQUESTING A VALUATOR

ZWVAL (ECHO,SUBVAL,VBUTN,VALUE) Request valuator input and report value

A call to ZWVAL causes the application program to wait until a valuator button is pressed. The value of the selected button is then returned in VBUTN and the value of the specified valuator is returned in VALUE. Each valuator device has its own set of buttons which may or may not be the same as those that comprise the button device. If an invalid button is pressed, VBUTN will be returned as 0; otherwise, the integer value of the button that was pressed will be returned.

VALUE will always range from 0.0 to 1.0, and the desired echo and subvaluator are specified in the respective variables ECHO and SUBVAL. Refer to the *Device Handlers Manual* for the echoes and subvaluators supported on a particular device.

CONTROL

General

This section discusses initialization and termination, inquiry, timing and escape functions.

System Initialization and Termination

ZBEGN Initialize the DGL system
ZEND Terminate the DGL system

The DGL system provides one subroutine to initialize the DGL system (ZBEGN) and another to terminate it (ZEND). The system must be initialized before

any other DGL subroutine is called. A subroutine call to ZBEGN must be the first DGL call made by the application program. When the DGL system is initialized, all system maintained values are set to their initial values.

ZEND terminates the DGL system. Termination includes making the picture current as well as disabling and terminating all currently enabled devices.

ZEND should be the last DGL call in an application program.

Device Control

GENERAL

The DGL system is centered around the concepts of logical devices and a work station. A logical device is a hypothetical device which can perform input and/or output uniformly to several different physical devices.

Each logical device (e.g., button, graphics display device) from which any input will be requested or output will be sent must be enabled first. Likewise, every device must be disabled before the DGL system is terminated. The DGL system has one subroutine to enable and one to disable each device.

A work station does not have to support every logical device. The devices available on a particular work station are determined by the logical device handlers loaded with the application program. The logical devices supported by a work station can be inquired from the DGL system at any time after the DGL system is initialized.

One application program can sequentially perform graphics I/O to several different devices of the same type. After the logical device has been disabled, it can be reassociated with the same device type merely by making a call to reinitialize with a different value of the I/O unit descriptor.

The I/O unit descriptor is a name or a number by which the operating system identifies the device. See the system supplement for details.

In the subroutine calls in this chapter, the term I/O unit descriptor appears in italics to indicate that it is not a literal name; it must be replaced by the parameter or parameters appropriate to the system.

```
INTEGER CONTRL,IERR
REAL XHOUS,YHOUS,XTREE,YTREE
CONTRL=0

CALL ZDINT(I/O unit descriptor,CONTRL,IERR)
     .                              * Initialize HP 2623 graphics display.
     .
     .
CALL HOUS(XHOUS,YHOUS)              * Draw the image on the HP 2623.
CALL TREE(XTREE,YTREE)
     .
     .
CALL ZDEND                          * Disable the HP 2623.
     .
     .
CALL ZDINT(I/O Unit Descriptor,CONTRL,IERR)
                                    * Initialize graphics display:
     .                                a different HP 2623.
     .
CALL HOUS(XHOUS,YHOUS)              * Draw the same image on the same
CALL TREE(XTREE,YTREE)              * type of graphics display device.
     .
     .
CALL ZDEND                          * Disable the HP 2623.
```

GRAPHIC DISPLAY DEVICE

ZDINT (*I/O unit descriptor*,CONTRL,IERR) Enable the graphics display device
ZDEND Disable the graphics display device

ZDINT performs device initialization and enables the graphics display for output.

The variable CONTRL can be used to specify how output will be sent to the device. For example, the program can specify whether graphics data can be outspooled, and whether the display will be cleared when it is initialized.

ZDEND disables the graphics display device. In disabling the device, the DGL system makes the picture current, releases all resources being used by the device, and performs any termination sequences required by the device. It does not clear the display surface.

OTHER LOGICAL DEVICES

ENABLE	DISABLE	FUNCTION
ZAINT(*I/O UNIT DESCRIPTOR*,IERR)	ZAEND	Alphanumeric device
ZBINT(*I/O UNIT DESCRIPTOR*,IERR)	ZBEND	Button
ZKINT(*I/O UNIT DESCRIPTOR*,IERR)	ZKEND	Keyboard
ZLINT(*I/O UNIT DESCRIPTOR*,IERR)	ZLEND	Locator
ZVINT(*I/O UNIT DESCRIPTOR*,IERR)	ZVEND	Valuator

With the exception of the graphics display the DGL system enables and disables all devices in a similar manner; each has one call to enable and one to disable the device.

Every initialization call ensures that the picture is made current before attempting to initialize a device. Logical devices must be enabled before they are used for input or output. A logical input device cannot be enabled on the same physical device as that of an outspooled graphics display device. If an application program attempts to enable a logical device that is currently enabled, the enabled device will be terminated and the call will be continued.

Disabling a logical device causes any required termination sequence to be performed and all resources allocated to the device to be released. An application program should disable every logical device used before terminating DGL.

Clearing the Display Surface

ZNEWF — Clear the graphics display device

A subroutine call to ZNEWF makes the picture current and then performs a new-frame-action. A new-frame-action clears the graphics display of all graphics output. A new-frame-action has different connotations for each graphics display device. The screen will be erased on CRT devices such as graphics terminals. Plotters with page advance will advance the paper. A call to ZNEWF may only make the picture current on devices such as drum plotters or fixed page plotters.

Controlling the Timing Modes

ZBMOD (OPCODE) — Set the timing mode

The immediacy by which output primitives are sent to the graphics display may be controlled. Whether graphics commands are immediately sent to a device or whether they are placed in a buffer before being transmitted has a large effect on the efficiency of the DGL system and the immediacy of visual changes to the graphics display. The user can specify two timing modes, immediate visibility and system buffering.

When in immediate visibility mode, graphics commands will be sent to the display device before returning from the DGL subroutine called. Any requested change is displayed as it is made. This type of output is inefficient, and can cause noticable system degradation. It should only be used in applications which require any picture changes made to the graphics display device occur as they are generated.

The alternative to immediate visibility mode, which should be used in most applications, is to use system buffering mode, and explicitly make the picture current only at times when the picture must be current. When operating in this mode, requested picture changes will be placed in a buffer by the DGL system. The buffer is automatically flushed by DGL when the buffer is full. Graphics throughput will be improved since the number of data transfers is reduced significantly. The same information is sent to the device regardless of the timing mode used, but the information is sent in larger blocks when in system buffering mode so that fewer I/O transfers are required.

On slow devices, such as plotters, the actual time it takes to plot a picture may not be noticeably reduced when using system buffering mode rather than immediate visibility mode. However, the amount of CPU time required will be less when using system buffering mode and will therefore increase the overall throughput of a computer system used in a multi-user or multitasking environment.

Making the Picture Current

ZMCUR — Make the picture current

The graphics display device surface can be made current at any time with ZMCUR. This ensures that all previously generated primitives have been sent to the graphics display device. ZMCUR does not change the timing mode.

Many DGL functions implicitly make the picture current. For example, each of the input routines use it to ensure that the image is fully defined before the operator requests input.

Inquiry Functions

ZIACS (DWIDE,DHIGH,AWIDE,AHIGH)	Actual size of graphics text
ZICOL (COLOR,COLP1,COLP2,COLP3)	Color table entry
ZIPST (PINDEX,DENSTY,ORIENT,EDGE)	Polygon table entry
ZIWS (OPCODE, *machine-dependent parameters*, ISIZE, RSIZE, *machine-dependent parameters*, ILIST,RLIST,IERR)	Other system information

A graphics application program can use inquiry calls in various ways. Inquiry calls can help debug a program, or help make it adaptable to different input and output devices. For example, the program can ask whether the work station supports a logical locator device. If not, the program can simulate the locator device by asking the user to type the current locator position on a keyboard device. Thus the program stays device independent and flexible.

ZIACS returns, in world coordinates, the actual character-cell width and height (AWIDE and AHIGH) that correspond to a desired width and height (DWIDE and DHIGH). This shows how closely the size of a displayed graphics character matches the requested size, given the capabilities of the graphics display device. When an exact match to the requested character size is not possible, ZIACS returns the nearest available size according to the "smaller best fit" rule (see ZCSIZ in Chapter 3).

Since both the both the desired and the actual sizes are in world coordinates, the physical size of text characters, as drawn or displayed, also depends on:

- The size of the CRT screen or plotting surface,
- The current world coordinate system, and
- The viewing transformations in effect.

ZICOL returns the color model values (COLP1, COLP2, COLP3) assigned to the specified index (COLOR) in the color table of the graphics device. ZDCOL can change the values of COLP1, COLP2 and COLP3. The interpretation of COLP1, COLP2 and COLP3 depends on the color model, which can be changed by ZCOLM. The subsection "Color" in this chapter talks about ZDCOL, ZCOLM, color tables, and color models.

ZIPST returns the polygon style values (DENSTY, ORIENT, EDGE) assigned to the specified index (PINDEX) in the polygon style table of the graphics device. ZDPST can change the values of DENSITY, ORIENT and EDGE. The section in this chapter, "Polygon Style", discusses ZDPST and the meanings of DENSTY, ORIENT, and EDGE.

ZIWS returns data on the current state of the DGL system. For details, refer to the section on ZIWS in Chapter 3 of this manual. One more inquiry call, ZIESC, is described in the next subsection.

Escape Functions

ZIESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)	Input escape access
ZOESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)	Output escape access

In addition to providing a wide variety of device-independent functions, the DGL system provides a mechanism to access some capabilities of devices that are not supported in a device independent manner. For example, the HP 7580 Drafting Plotter has a built-in circle generator. The mechanism by which these hardware features are accessed is called escape functions. There are two types of escape functions. Output escape functions, implemented by ZOESC, provide access to special features of the graphics display device. ZIESC allows a program to inquire about the capabilities of the device. The feature to be accessed is specified by the value of OPCODE. Since the value of OPCODE may specify a different function for each device, refer to the *Device Handlers Manual* for the functions supported by a specific graphics display device and the parameters required. If the OPCODE specified is not supported by a particular device an error will be returned and the call will be ignored.

Escape functions can be used anywhere in an application program after the graphics display has been initialized. Inquiry escape functions may not be requested from a device that is being outspooled.

Escape functions bypass the DGL system and interact with a device directly. The DGL system does not keep track of the use made of the escape functions. Escape functions may cause the DGL system to lose track of the state of the graphics display device. Therefore, they should be used very carefully. Because escape functions are by nature device dependent, they should be used sparingly in order to preserve the device independence of application programs.

Chapter 3
Detailed Description of DGL Subroutines

SYSTEM-SPECIFIC CALLS

The following system-specific calls are described in the appropriate system supplement:

ZAINT	ZDINT	ZKINT	ZVINT
ZBINT	ZIWS	ZLINT	

DEFINITION FORMAT

Each subroutine definition contains the following information: purpose, calling sequence, definition of parameters, commentary, and error conditions. The parameters in DGL calls are always arranged so that input parameters precede output parameters. Input parameters are defined as being parameters passed to the DGL system, whereas output parameters are returned from DGL. Output parameters are underlined in the calling sequence for easy reference. A parameter is never used as both an input and output parameter. DGL will not change the value of any input parameters.

All integer parameters in DGL are treated as single precision one word integers and all reals are treated as single precision two word reals.

DGL does a minimal amount of error reporting. The conditions which can cause errors are documented for every subroutine. Some subroutines (e.g., ones which initialize devices, inquire information, and change the display limits) return an error parameter which indicates whether the operation was performed successfully. In the following subroutine descriptions the error conditions are listed in the order in which they are tested.

ZAEND

PURPOSE: To disable the enabled alphanumeric device.

CALLING SEQUENCE: CALL ZAEND { no parameters }

ZAEND terminates and disables the alphanumeric device. It transmits any required termination sequence to the device and releases all resources being used by the device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0.

ZAEND is the complementary routine to ZAINT.

If an alphanumeric device is used, ZAEND should be called before the application program is terminated.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Alphanumeric device not currently enabled.
   ACTION: Call ignored. A call to ZAINT should be made prior to this call.

ZALPH

PURPOSE: To output a text string to the alphanumeric device.

CALLING SEQUENCE: CALL ZALPH (NCHARS, STRING)

NCHARS	[INTEGER; Input] The number of characters from STRING to be output to the alphanumeric device. (1 <= NCHARS <= 132).
STRING	[INTEGER; Array; Input] The text string to be output in Packed ASCII format.

ZALPH allows the application program to send non-graphics text and data to the enabled alphanumeric device. This call can be used to send prompts, give status, or print error messages.

Characters 1 through NCHARS in STRING are sent to the alphanumeric device. If NCHARS is less than 1 or greater than 132, the call is ignored.

Some devices can display only N characters, where N<132. If an application program attempts to output an NCHARS number of characters, where NCHARS is greater than N, only the first N characters of array STRING will be displayed. A program can use the subroutine ZIWS to inquire the maximum number of characters that can be output on each line of the alphanumeric device. ZIWS can also be used to determine the maximum number of lines viewable at any one time on the alphanumeric device.

A carriage-return line-feed (CRLF) is normally appended to the end of STRING when it is sent to the device. To supress the CRLF, an underscore must be the last character in STRING. The underscore will not be displayed but must be included in the character count, NCHARS. If NCHARS is greater than the number of characters a device can display and an underscore is the last character in the string, the underscore will be ignored and CRLF will be appended to the string when it is sent to the device.

STRING must be dimensioned large enough to contain the number of characters defined by NCHARS.

ZALPH

ZALPH insures that the picture is current before any output is sent to the alphanumeric device.

Before ZALPH is called the alphanumeric device must be enabled for output with a call to ZAINT. It is disabled with a call to ZAEND.

If the alphanumeric device is physically the same device as the graphics display device, there are some restrictions as to what should be passed in STRING. Any device commands which are sent to the alphanumeric device that affect the graphics display could destroy the integrity of the DGL system by putting the device in an unknown state.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The alphanumeric device is not currently enabled.
   ACTION: Call ignored.

3) NCHARS < 1 or NCHARS > 132.
   ACTION: Call ignored.

4) NCHARS > Maximum number of characters which can be displayed at one time.
   ACTION: First N characters will be displayed, where N = maximum number of characters that can be displayed at once.

ZASPK

PURPOSE: To redefine the aspect ratio of the virtual coordinate system.

CALLING SEQUENCE: CALL ZASPK (XSIZE, YSIZE)

XSIZE [REAL; Input]
The width of the virtual coordinate system in dimensionless units. (XSIZE > 0.0)

YSIZE [REAL; Input]
The height of the virtual coordinate system in dimensionless units. (YSIZE > 0.0)

ZASPK sets the aspect ratio of the virtual coordinate system, and hence the view surface, to be YSIZE divided by XSIZE. A ratio of 1.0 defines a square virtual coordinate system, a ratio greater than 1.0 specifies it to be higher than it is wide; and a ratio less than 1.0 specifies it to be wider than it is high. Since XSIZE and YSIZE are used to form a ratio, they may be expressed in any units as long as they are the same units.

DGL calculates the range of the coordinates of the virtual coordinate system based on the value of the aspect ratio. The coordinates of the longer axis are always set to range from 0.0 to 1.0 and those of the shorter axis from 0.0 to a value that achieves the specified aspect ratio. ZASPK redefines the limits of the virtual coordinate system as shown in Table 3-1.

Table 3-1. Aspect Ratio

ASPECT RATIO (AR)	X LIMITS	Y LIMITS
AR<1.0	(0.0, 1.0)	(0.0, 1.0*AR)
AR=1.0	(0.0, 1.0)	(0.0, 1.0)
AR>1.0	(0.0, 1.0/AR)	(0.0, 1.0)

ZASPK

When a call to ZASPK is made, the DGL system sets the viewport equal to the limits of the virtual coordinate system. If the viewport is reset with a call to ZVIEW, the viewport limits must be within the X and Y limits of the virtual coordinate system. A program could display an image on the entire logical display surface in the following manner:

```
INTEGER IDUMMY,IERR
REAL ASPK(2)
   .
   .
   .
CALL ZIWS (254,0,2,IDUMMY,ASPK,IERR)      * Inquire the aspect ratio
                                          * of the logical limits.
CALL ZASPK (1.0,ASPK(2))                  * Set the aspect ratio.
   .
   .
   .
```

The initial value of the aspect ratio is 1.0, setting the virtual coordinate system to be a square. This square is mapped to the largest inscribed square on any display surface, so that the viewable area is maximized. As a result, the initial virtual coordinate system limits range from 0.0 to 1.0 in both the X and Y directions. A program can access the largest inscribed rectangle on the logical display surface by modifying the value of the aspect ratio. The exact placement of the rectangle on the logical display surface is device dependent, but is normally centered on the CRT and justified in the lower left-hand corner of the plotter (see the *Device Handlers Manual*).

The starting position is not altered by this call. Since this call redefines the viewing transformation, the starting position may no longer represent the last world coordinate position. A call to ZMOVE should therefore be made after this call to update the starting position.

If the logical locator device is associated with the same physical device as the graphics display, then a call to ZASPK will implicitly set the logical locator limits equal to the new limits of the virtual coordinate system.

Since the window is not affected by the ZASPK call, distortion may result in the window to viewport mapping if the window does not have the same aspect ratio as the virtual coordinate system (see ZWIND).

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) XSIZE or YSIZE is less than or equal to zero.
   ACTION: Call ignored.

ZBEGN

PURPOSE: Initialize the DGL system.

CALLING SEQUENCE: CALL ZBEGN { no parameters }

ZBEGN initializes the DGL system. It must be the first DGL call made by the application program. Any call other than ZBEGN which is made while the system is not initialized will be ignored. ZBEGN performs the following operations:

- Sets the timing mode to system buffering mode.
- Sets the aspect ratio of the view surface to 1.0.
- Sets the virtual coordinate and viewport limits to range from 0.0 to 1.0 in the X and Y directions.
- Sets the starting position to (0.0,0.0) in world coordinate system units.
- Sets all attributes equal to their default values.

The DGL system is terminated with a call to ZEND.

ERROR CONDITIONS:

1) The DGL system is already initialized.
   ACTION: Call ignored.

ZBEND

PURPOSE: To disable the enabled button device.

CALLING SEQUENCE: CALL ZBEND { no parameters }

ZBEND disables and terminates the button device. It transmits any termination sequence required by the device and releases all resources being used by the device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0.

ZBEND is the complementary routine to ZBINT.

If the button device is used, ZBEND should be called before the application program is terminated.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Button device not currently enabled.
   ACTION: Call ignored.

ZBMOD

PURPOSE: To select the timing mode for graphics output.

CALLING SEQUENCE: CALL ZBMOD (OPCODE)

OPCODE [INTEGER; Input]
The code which determines the timing mode to be used with the graphics display.

OPCODE = 0 - Immediate visibility mode

= 1 - System buffering mode

DGL timing modes are provided to control DGL graphics throughput and picture update timing. Picture update timing refers to the immediacy of visual changes to the graphics display surface. Regardless of the timing mode used, the same final picture is sent to the graphics display. ZBMOD only controls when a picture appears on the graphics display, not what appears.

The DGL system supports two timing modes:

Immediate visibility mode

Requested picture changes will be sent to the graphics display device before control is returned to the calling program. Due to operating system delays there may be a delay before the picture changes are visible on the graphics display device.

The immediate visibility mode is less efficient than the system buffering mode. It should only be used in those applications that require picture changes to take place as soon as they are defined, even if the finished picture takes longer to create. When changing the timing mode to immediate visibility mode the picture is made current.

System buffering mode

Requested picture changes will be buffered by the DGL system. This means that graphic output will not be immediately sent to the display device. This allows the DGL system to send several graphics commands to the graphics display device in one data transfer, therefore, reducing the number of transfers. System buffering is the initial timing mode.

The following routines implicitly make the picture current:

ZALPH, ZBUTN, ZKYBD, ZSLOC, ZSVAL, ZWLOC, ZWVAL, ZIESC, ZDEND, ZAINT, ZBINT, ZKINT, ZLINT and ZVINT.

An alternative to immediate visibility that will solve many application needs is the use of system buffering together with the ZMCUR call. With this method, an application program places graphics commands into the output buffer and flushes the buffer (see ZMCUR) only at times when the picture must be fully displayed.

A call to ZMCUR can be made at any time within an application program to insure that the image is fully defined. ZMCUR flushes the output buffer but does not modify the timing mode.

Before performing any non-DGL input or output such as a FORTRAN, READ or WRITE, the DGL buffer must be empty. If the buffer is not flushed (via immediate visibility or ZMCUR) prior to a non-DGL I/O call, the resulting image may contain some "garbage" such as escape functions or invalid graphics data.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Invalid OPCODE specified.
   ACTION: Call ignored.

ZBUTN

PURPOSE: To return a button value from the enabled button device.

CALLING SEQUENCE: CALL ZBUTN (ECHO, BUTTON)

ECHO [INTEGER; Input]
The type of input echoing. Possible values are:

ECHO = 0 - No echoing

= 1 to 255 - Echo on the button device

BUTTON [INTEGER; Output]
The integer button value returned by the button device in the range 0 to 255.

ZBUTN waits for a button to be selected on the currently enabled button device. When a button is selected, that button's value is returned in BUTTON.

ZBUTN normally returns a positive integer from 1 to 255 as a function of the button or key selected. A zero is returned if an invalid button is activated on the button device. The number of buttons supported by a button device is device dependent. The correlation between the buttons or keys supported on a device and the value returned in BUTTON is also device dependent.

The number of echoes supported on a button device and the correlation between ECHO and the type of echoing performed is device dependent. Most button devices support at least one type of echo. Possible echoes are turning on a light, beeping, or displaying the button activated. Refer to the *Device Handlers Manual* to determine the number and type of echoing performed by a specific button device. If ECHO is larger than the number of echoes supported by the enabled button device, then the call will be treated as if ECHO = 1.

ZBUTN implicitly makes the picture current before requesting input from the button device.

The button device should be enabled before ZBUTN is called (see ZBINT). If the button device is not enabled, the call will be ignored. The button device is terminated with a call to ZBEND.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The logical button device is not enabled.
   ACTION: Call ignored.

3) ECHO is out of range.
   ACTION: Treated as if ECHO = 1.

4) The button activated is out of range.
   ACTION: VALUE set to 0.

ZCOLM

PURPOSE: To choose the color model.

CALLING SEQUENCE: CALL ZCOLM (MODEL)

MODEL [INTEGER; Input]
The color model DGL will use to interpret subsequent changes to or inquires of entries in the color table.

MODEL = 1 - The RGB (Red-Green-Blue) color cube; the default model.

= 2 - The HSL (Hue-Saturation-Luminosity) color cylinder.

The RGB physical model (see Figure 2-5) is a color cube with the primary additive colors, red, green and blue, as its axes. With this model, a call to ZDCOL specifies a point within the color cube that has a red intensity value (X coordinate), a green intensity value (Y coordinate) and a blue intensity value (Z coordinate). Each value ranges from zero (no intensity) to one.

The HSL perceptual model (see Figure 2-6) is a color cylinder in which:

- The angle about the axis of the cylinder, in fractions of a circle, is the hue (red is at 0, green is at 1/3 and blue is at 2/3).
- The radius is the saturation -- ranging from white through grey to black along the center axis (S=0) to saturated hues, with no apparent whiteness, at the cylinder's surface (S=1).
- The height along the center axis is the luminosity (the intensity or brightness per unit area); black is at the bottom of the cylinder (L=0), and the brightest colors are at the top of the cylinder (L=0) with white at the center top.

Hue (Angle), saturation (radius), and luminosity (height) all range from zero to one. Using this model, a call to ZDCOL specifies a point within the color cylinder that has a hue value, a saturation value, and a luminosity value.

ZCOLM

When a call to ZCOLM switches color models, parameter values in subsequent calls to ZDCOL then refer to the new model. Switching models does not affect color definitions that were previously made using another model. Note that when the value of a color table entry is inquired, it is returned in the current model, which may not be the model in which it was originally specified.

Not all color specifications can be displayed on every graphics device, since the devices which DGL supports differ in their capabilities. If a color specification is not available on a device, DGL will use the closest available color.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The value of MODEL is not 1 or 2.
   ACTION: Call ignored.

ZCOLR

PURPOSE: To set the color attribute for all primitives except polygon interior fill.

CALLING SEQUENCE: CALL ZCOLR (COLOR)

COLOR	[INTEGER; Input] Desired color to be applied to all primitive output except for polygon interior fill. COLOR is the index to an entry in a device-dependent color table. (0 < = COLOR < = 32767)

ZCOLR chooses the color index to be used for the following primitives:

- o Lines
- o Polylines
- o Graphics text
- o Markers
- o Polygon set edges

The user chooses the color entry to be used for polygon interiors using ZPICL.

Every graphics display device handler in DGL has its own default color table. The size of the color table will vary with the device. Every color table has at least one entry at index 1. Display devices for which there is a background color have an entry at index 0 which is the background color. Color 1 is the default color set at DGL initialization and display initialization for all primitives.

The user who does not wish to use the default color 1 may choose other entries in the color table to be used instead. Any entry in the color table may be chosen by ZCOLR. If ZCOLR specifies Color 0, the background color, results will be device dependent. For example, on plotters, the pen will be put away. On some raster displays, lines will be drawn in the background color allowing the user to erase existing primitives by drawing over them.

It is possible to define an entry in a device color table which the device is not capable of displaying. When such an entry is chosen using ZCOLR, DGL attempts to match the requested color with the closest color of which the device is capable. The resulting color may duplicate the color of some other entry in the table.

ZCOLR

On some devices, the ability of the device to display a color will depend on the type of primitive being displayed. This will frequently happen on color displays which are capable of filling polygon interiors in a wide range of colors but which have a narrow selection for all other primitives. In this case, line color, for example, may only approximate the color of the entry selected by ZCOLR while polygon set interior color may be much closer to the color selected with ZPICL.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) The value of COLOR is outside the range of the system (0 to 32767).
   ACTION: Color index 1 is used.

4) The value of COLOR is outside the range supported by the graphic display device.
   ACTION: Color index 1 is used.

ZCSIZ

PURPOSE: Sets the character size attribute for hardware text.

CALLING SEQUENCE: CALL ZCSIZ (WIDTH, HEIGHT)

WIDTH	[REAL; Input] Requested hardware character cell width in world coordinate units.
HEIGHT	[REAL; Input] Requested hardware character cell height in world coordinate units.

ZCSIZ sets the character size for subsequently output graphics text. WIDTH and HEIGHT specify the world coordinate size of a character cell. Therefore, the actual physical size of the characters output is determined by applying the current viewing transformations to the world coordinate units specification. Note that a consequence of this is that changing the window specification (or other viewing components) can alter the apparent size of characters on the graphics display surface.

The actual character size used is device dependent since not all devices support continuously variable character sizes. If the actual character cell size requested is not available on the graphics display, then the "smaller best fit" size available will be used. The smaller-best-fit character size is defined as:

- The size of the largest character whose cell height is less than or equal to the requested height and whose cell width is less than or equal to the requested width, or
- The smallest available size, if the size requested is smaller than all available sizes.

ZIACS can be used to inquire about the actual character size that will be used by the graphics display device without setting the character size attribute.

ZIWS can be used to inquire the actual character size that is set by a call to ZCSIZ.

The character size attribute has no effect on alphanumeric text output using ZALPH.

The default character size (set by ZDINT and ZBEGN) is determined as follows:

1. Height = .05 x (current height of the world coordinate system).

2. Width = .035 x (current width of the world coordinate system).

3. The "smaller best fit" rule is then applied.

If a change is made to the viewing transformation (by ZWIND, ZVIEW, ZDLIM, or ZASPK), the value of the character size attribute will not be changed, but the actual size of the characters generated may be modified.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Graphics display not currently enabled.
   ACTION: Call ignored.

ZDCOL

PURPOSE: To redefine an entry in the color table.

CALLING SEQUENCE: CALL ZDCOL (COLOR,COLP1,COLP2,COLP3)

COLOR [INTEGER; Input]
The entry in the color table which is to be redefined. (0 <= COLOR <= 32767)

COLP1,COLP2,COLP3 [REAL; Input]
New color model values for the entry selected by COLOR. (0.0 <= COLP1,COLP2,COLP3 <= 1.0)

Every graphics display device handler in DGL has its own default color table. Every default color table has at least an entry at index 1 and display devices for which there is a background color have an entry at index 0. Color 1 is the default color for all primitives and Color 0 is the background color of the display device. The other entries in the default color table are usually selected to match the basic colors which may be displayed on the device.

The color model parameters set by ZDCOL, and the current color model set by ZCOLM, define a new color. This color definition can then be chosen by calling ZCOLR or ZPICL. Note that ZDCOL is ignored if the graphic output device does not permit its color table to be redefined or if color modelling capabilities were suspended by replacing color modelling modules with dummy modules (HP 1000 only). Call ZIWS to inquire whether the current device permits its color table to be redefined.

If entry redefinition is supported, a new color value for the entry specified in parameter COLOR is set according to the values of the parameters COLP1, COLP2, and COLP3. When the value of a color entry is redefined through ZDCOL, any subsequently output primitives which use that entry will reflect the newly defined value. The effect of redefinitions on previously output primitives is device dependent. On most devices, previously output primitives which used the redefined entry will not change color. However, on some devices the redefinition may be retroactive.

The values of COLP1, COLP2, and COLP3 are interpreted according to one of two color models, RGB (Red-Green-Blue) or HSL (Hue-Saturation-Luminosity). DGL uses the latest model specified in the ZCOLM call, or the default RGB model if no call to ZCOLM has been made. See Table 3-2 for an example of the RGB sample colors and Table 3-3 for an example of the HSL sample colors.

Table 3-2. Sample Colors Using the RGB Color Model

COLP1 RED INTENSITY	COLP2 GREEN INTENSITY	COLP3 BLUE INTENSITY	RESULTING COLOR
0.0	0.0	0.0	Black
0.5	0.5	0.5	Gray
1.0	1.0	1.0	White
1.0	0.0	0.0	Red
1.0	1.0	0.0	Yellow
0.0	1.0	0.0	Green
0.0	1.0	1.0	Cyan
0.0	0.0	1.0	Blue
1.0	0.0	1.0	Magenta

Table 3-3. Sample Colors Using the HSL Color Model

COLP1 HUE	COLP2 SATURATION	COLP3 LUMINOSITY	RESULTING COLOR
Any	Any	0.0	Black
Any	0.0	0.5	Gray
Any	0.0	1.0	White
0.0 or 1.0	1.0	1.0	Red
1/6 = 0.17	1.0	1.0	Yellow
2/6 = 0.33	1.0	1.0	Green
3/6 = 0.50	1.0	1.0	Cyan
4/6 = 0.67	1.0	1.0	Blue
5/6 = 0.83	1.0	1.0	Magenta

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) The color table cannot be changed.
   ACTION: Call ignored.

4) The value of COLOR is outside the range of the system (0 to 32767).
   ACTION: Call ignored.

5) The value of COLOR is outside the range supported by the graphic display device handler.
   ACTION: Call ignored.

6) The value of COLP1, COLP2 or COLP3 is outside the range 0.0 to 1.0.
   ACTION: Call ignored.

ZDEND

PURPOSE: To disable the enabled graphics display device.

CALLING SEQUENCE: CALL ZDEND { no parameters }

ZDEND terminates the device enabled as the graphics display in the work station. ZDEND completes all remaining display device operations and disables the logical graphics display. It makes the picture current and releases all resources being used by the device. ZDEND does not send a new-frame-action to the graphics display device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0. If there is an enabled locator which is the same physical device as the graphics display device that is being disabled, the locator limits will be reset to the default limits.

The graphics display device should be disabled before the termination of the application program. ZDEND is the complementary routine to ZDINT.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical graphics display not currently enabled.
   ACTION: Call ignored.

ZDLIM

PURPOSE: To define the logical display limits of the graphics display.

CALLING SEQUENCE: CALL ZDLIM (XMIN, XMAX, YMIN, YMAX, IERR)

XMIN [REAL; Input]
The left side of the logical display limits is offset from the origin of the physical graphics display device (distance in millimeters).

XMAX [REAL; Input]
The right side of the logical display limits is offset from the origin of the physical graphics display device (distance in millimeters).

YMIN [REAL; Input]
The bottom of the logical display limits is offset from the origin of the physical graphics display device (distance in millimeters).

YMAX [REAL; Input]
The top of the logical display limits is offset from the origin of the physical graphics display device (distance in millimeters).

IERR [INTEGER; Output]
Return code indicating whether (XMIN,YMIN) and (XMAX,YMAX) are within the physical display limits.

IERR = 0 (XMIN,YMIN) and (XMAX,YMAX) are within the physical display limits.

= 1 XMIN >= XMAX or YMIN >= YMAX. Call ignored.

= 2 (XMIN,YMIN) or (XMAX,YMAX) is outside the physical display limits. Call ignored.

ZDLIM

ZDLIM allows an application program to specify the region of the display surface where the image will be displayed. This region is called the logical display surface. Upon display device initialization, the DGL system sets the logical display limits to be equal to a device-dependent portion of the specified physical device. By changing the logical display limits, any portion of the physical graphics display can be addressed.

The pairs (XMIN,YMIN) and (XMAX,YMAX) define the corner points of the new logical display limits in terms of millimeters offset from the origin of the physical display. The exact position of the physical display origin is device dependent. (Refer to the *Device Handlers Manual*).

ZDLIM sets the view surface to be the maximum rectangle within the logical display limits having the current aspect ratio of the virtual coordinate system. ZDLIM does not effect the virtual coordinate system limits, but it may reset the view surface limits; therefore, it may have the effect of scaling and translation. A call to ZDLIM also sets the locator echo position to its default value, the center of the world coordinate system. This applies in all cases, not just when the display and location are the same device.

The starting position is not altered by this call. Since this call redefines the viewing transformation, the starting position may no longer represent the last world coordinate position. A call to ZMOVE, ZPOLY, ZPGDD, or ZPGDI should therefore be made after this call to update the starting position.

If the logical display and logical locator are associated with the same physical device, a call to ZDLIM will set the logical locator limits equal to the new view surface limits.

ZDLIM should only be called while the graphics display is enabled.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The graphics display is not enabled.
   ACTION: Call ignored.

3) XMIN >= XMAX or YMIN >=YMAX
   ACTION: IERR set to 1 and the rest of call ignored.

4) One of the offsets specified is outside of the physical display limits.
   ACTION: IERR set to 2 and the rest of call ignored.

ZDPMM

PURPOSE: To convert from world coordinates to millimeters on the graphics display.

CALLING SEQUENCE: CALL ZDPMM (WX, WY, MMX, MMY)

WX [REAL; Input]
The X coordinate of a point to be converted from world coordinates to millimeters on the graphics display.

WY [REAL; Input]
The Y coordinate of a point to be converted from world coordinates to millimeters on the graphics display.

MMX [REAL; Output]
The number of millimeters that the world coordinate point (WX,WY) is from the origin along the X axis on the graphics display.

MMY [REAL; Output]
The number of millimeters that the world coordinate point (WX,WY) is from the origin along the Y axis on the graphics display.

ZDPMM converts the world coordinate point (WX,WY) to the point (MMX,MMY), expressed in millimeters from the origin of the graphics display device. The location of this origin is device dependent. Refer to the *Device Handlers Manual* for the position of the origin on a specific device.

Since the origin of the world coordinate system need not correspond to the origin of the physical graphics display, converting the point (0.0,0.0) in the world coordinate system may not result in the value (0.0,0.0) offset from the physical display device's origin.

ZDPMM accepts any world coordinate point, inside or outside the current window, and converts to a point offset from the physical display device's origin.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The graphics display is not enabled.
   ACTION: Call ignored.

ZDPST

PURPOSE: To redefine the polygon style of an entry in the polygon style table.

CALLING SEQUENCE: CALL ZDPST (PINDEX,DENSTY,ORIENT,EDGE)

PINDEX [INTEGER; Input]
The index to the entry in the polygon style table. (1 <= PINDEX <= 255)

DENSTY [REAL; Input]
The desired density of polygon interior fill. (-1.0 <= DENSTY <= 1.0)

ORIENT [REAL; Input]
The desired orientation of polygon interior fill lines. This is the angle (degrees) that the fill lines will make with the horizontal axis of the view surface. For a cross-hatched fill, the angular value of ORIENT refers to one set of fill lines. The second set is perpendicular to the first. (-90.0 <= ORIENT <= 90.0)

EDGE [INTEGER; Input]
Specification controlling whether display of the edge of the polygon set is permitted.

EDGE = 0 - No polygon edge display.

= 1 - Polygon edge display permitted. Display of particular edge segments depends on the OPCODE parameter in the call that generates the polygon set: ZPGDD or ZPGDI.

Every graphics device handler in DGL has a default polygon style table. The number of entries in the table and their values depend on the graphics device. Default definitions are designed to suit each graphics device. The first 16 entries are the same for all devices (see Figure 2-9). For some devices, entries can be added to or deleted from the table before program execution.

When the value of a polygon style entry is redefined through ZDPST, any subsequently output polygon sets which use that entry will reflect the newly defined value. The effect of redefinitions on previously output polygons sets is device dependent. On most devices, previously output polygon sets which used the redefined entry will not change styles. However, on some devices the redefinitions may be retroactive.

DENSTY defines the ratio of filled area (area covered by fill lines) to total area in the interiors of polygon sets. For completely filled polygon sets, make DENSTY = $\pm$ 1.0; for no fill, make DENSTY = 0.0. Valid densities other than $\pm$ 1.0 and 0.0 describe partially filled interiors.

A positive density requests parallel fill lines in one direction only. A negative density requests cross-hatching with two sets of fill lines at right angles. For a given density, the distance between two adjacent parallel lines is greater with cross hatching than in the case of pure parallel filling.

Fill line width is always the thinnest width possible on the graphic display device. Also, the distance between fill lines, hence density, does not change with a change of scale caused by a viewing transformation.

The actual percentage of the area covered by lines can differ from the specified density. Density values assume unbroken fill lines. A difference will occur when the current interior linestyle, set by ZPILS, calls for gaps in the hatch lines. Device-dependent fills produced by ZPGDD may also deviate from the requested density.

Note: Problems with rasterization, device resolution and numerical roundoff may occasionally cause gaps when density = $\pm$ 1. This most commonly occurs on raster devices when ORIENT is not equal to -90.0, 0.0, or +90.0 degrees.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) Dummy polygon routines were loaded.
   ACTION: Call ignored.

4) The value of PINDEX is outside the range permitted by DGL (1 to 255).
   ACTION: Call ignored.

5) The value of PINDEX is outside the range supported by the graphic display device.
   ACTION: Call ignored.

6) The value of DENSTY is outside the range -1.0 to 1.0.
   ACTION: Call ignored.

7) The value of ORIENT is outside the range -90.0 to 90.0.
   ACTION: Call ignored.

8) The value of EDGE is not 0 or 1.
   ACTION: Call ignored.

ZDRAW

PURPOSE: To draw a line from the starting position to the world coordinate specified.

CALLING SEQUENCE: CALL ZDRAW (WX, WY)

WX	[REAL; Input] Ending X coordinate of the line to be drawn in the world coordinate system.
WY	[REAL; Input] Ending Y coordinate of the line to be drawn in the world coordinate system.

A line is drawn from the starting position to the world coordinate specified by (WX,WY). The starting position is updated to (WX,WY) at the completion of this call.

The DGL system does not perform any software clipping on output primitives. If either the starting position or the point (WX,WY) is outside of the current view surface, then the resulting picture may not be well defined because each device handles out-of-range data in a different manner.

Drawing to the starting position generates the shortest line possible. Depending on the nature of the linestyle, nothing may appear on the graphics display surface. See ZLSTL for a complete description of how linestyle affects a particular point or vector.

The primitive attributes of linestyle, color, highlighting and linewidth apply to lines drawn using ZDRAW. However, the lines will appear with these attributes only if the graphics device is capable of applying them to lines. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The graphics display is not enabled.
   ACTION: Call ignored.

ZEND

PURPOSE: To terminate the DGL system.

CALLING SEQUENCE: CALL ZEND { no parameters }

ZEND terminates the DGL system. Termination includes making the picture current as well as disabling and terminating all currently enabled devices. ZEND should be the last DGL call in an application program.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored.

ZHIGH

PURPOSE: Sets the highlighting attribute for subsequently output primitives.

CALLING SEQUENCE: CALL ZHIGH (HLIGHT)

HLIGHT	[INTEGER; Input] Highlighting value. (1 <= HLIGHT <= 255)

A call to ZHIGH allows an application program to specify the value of the highlighting attribute. Highlighting is a means of emphasizing portions of the displayed image. Highlighting is not performed when the highlighting attribute equals 1 (the initial value). A device may perform highlighting by blinking or intensifying subsequently generated output primitives.

The type of highlighting available is device dependent. The mapping between the values of HLIGHT and the supported highlighting on a device is also device dependent. Highlighting of a polygon set affects both the interior and exterior of the set.

If the requested highlighting is not available on the graphics display device, then the highlighting attribute will be set equal to the initial value of 1.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) Graphics display not currently enabled.
   ACTION: Call ignored.

3) The highlighting attribute specified by HLIGHT is not in the valid range of 1 to 255.
   ACTION: Graphics primitives will be output using highlight 1.

4) The highlighting attribute specified by HLIGHT is not supported by the graphics display.
   ACTION: Graphics primitives will be output using highlight 1.

ZIACS

PURPOSE: Given a desired character size, it returns the actual character size that will be used by the graphics display.

CALLING SEQUENCE: CALL ZIACS (DWIDE, DHIGH, AWIDE, AHIGH)

DWIDE [REAL; Input]
Requested character cell width in world coordinate units.

DHIGH [REAL; Input]
Requested character cell height in world coordinate units.

AWIDE [REAL; Output]
Actual width of the character cell, expressed in world coordinate units, which will be used to display graphics text.

AHIGH [REAL; Output]
Actual height of the character cell, expressed in world coordinate units, which will be used to display graphics text.

Given a desired character size, ZIACS returns the actual character size that will be used by the graphics display without setting the character size attribute. The actual character size used is device dependent since not all devices support continuous character sizes. If the desired character cell size requested is not available on the graphics display device, then the "smaller best fit" size available will be returned. The smaller best fit character size is defined to be:

A) The size whose cell height is <= the desired height and whose cell width is <= the desired width.

B) If the size requested is smaller than all available sizes then the smallest available size will be used.

ZCSIZ is used to set the character size attribute.

ERROR CONDITIONS:

1) The DGL system is not initialized.
ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The graphics display is not enabled.
ACTION: Call ignored.

ZICOL

PURPOSE: To inquire the color of an entry in the color table.

CALLING SEQUENCE: CALL ZICOL (COLOR,COLP1,COLP2,COLP3)

COLOR [INTEGER; Input]
The index to the color table entry.
(0 <= COLOR <= 32767)

COLP1,COLP2,COLP3 [REAL; Output]
Current color model values for the entry assigned to COLOR. (0.0 <= COLP1,COLP2,COLP3 <= 1.0)

ZICOL returns the color value of the indexed entry in the device-dependent color table.

COLP1, COLP2 and COLP3 define the color for the table entry indexed by COLOR, in accordance with the currently selected color model. If the RGB model is currently in use, COLP1 gives the red intensity, COLP2 the green intensity, and COLP3 the blue intensity. If the HSL model is currently used, COLP1 gives the hue, COLP2 the saturation and COLP3 the luminosity.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) The color table cannot be changed.
   ACTION: Call ignored.

4) The value of COLOR is outside the range of the system (0 to 32767).
   ACTION: Call ignored.

5) The value of COLOR is outside the range supported by the graphic display device.
   ACTION: Call ignored.

ZIESC

PURPOSE: To perform a device-dependent inquiry escape function from the graphics display device.

CALLING SEQUENCE: CALL ZIESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)

OPCODE [INTEGER; Input]
The opcode of the device-dependent inquiry escape function being invoked. The opcodes supported for a given device are described in the *Device Handlers Manual*.

ISIZE [INTEGER; Input]
The number of integer parameters to be returned in array ILIST by the escape function.

RSIZE [INTEGER; Input]
The number of real parameters to be returned in array RLIST by the escape function.

ILIST [INTEGER; Array; Output]
The array in which zero or more integer parameters are returned by the escape function. If ISIZE equals 0, ILIST is a dummy parameter.

RLIST [REAL; Array; Output]
The array in which zero or more real parameters are returned by the escape function. If RSIZE equals 0, RLIST is a dummy parameter.

IERR [INTEGER; Output]
Code indicating whether the inquiry escape function was performed. Possible values and their meanings are:

IERR = 0 Inquiry escape function successfully completed.

= 1 OPCODE not supported by the graphics display device. Call ignored.

= 2 ISIZE is not equal to the number of integer parameters to be returned. Call ignored.

= 3 RSIZE is not equal to the number of real parameters to be returned. Call ignored.

= 4 Graphics display is being outspooled. Therefore no input operation can be performed. Call ignored.

ZIESC allows application programs to inquire about special hardware features on a graphics display device. The desired inquiry escape function is specified by OPCODE. Possible inquiry escape functions may return the status or the options supported by a particular graphics display device. Refer to the *Device Handlers Manual* for the inquiry escape functions supported by a specific graphics display device. If the opcode specified is not supported by a particular device or if ISIZE or RSIZE are specified incorrectly an error code will be returned and the call will be ignored.

The type of information returned from the graphics display device is determined by the value of OPCODE. The thousands digit of OPCODE specifies the number of integer values returned in ILIST; the hundreds digit specifies the number of real values returned in RLIST. It is the application program's responsibility to insure that ILIST is dimensioned at least as large as specified by ISIZE, and that RLIST is dimensioned at least as large as specified by RSIZE. If ISIZE or RSIZE are not exactly the size required by a particular inquiry escape function, the call will be ignored and an error code returned in IERR.

Inquiry escape functions only apply to graphics display devices. ZIESC can be called anywhere in an application program while the graphics display device is enabled. ZIESC implicitly makes a picture current before performing the escape function.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The logical graphics display is not enabled.
   ACTION: Call ignored.

3) Graphics display device is being outspooled.
   ACTION: IERR set to 4 and rest of call ignored.

4) The specified OPCODE is not supported by the graphics display device.
   ACTION: IERR set to 1 and rest of call ignored.

5) ISIZE is not equal to the number of integer parameters returned by the inquiry escape function.
   ACTION: IERR set to 2 and rest of call ignored.

6) RSIZE is not equal to the number of real parameters returned by the inquiry escape function.
   ACTION: IERR set to 3 and rest of call ignored.

ZIPST

PURPOSE: To inquire the polygon style of an entry in the polygon style table.

CALLING SEQUENCE: CALL ZIPST (PINDEX,DENSTY,ORIENT,EDGE)

PINDEX [INTEGER; Input]
The index to the entry in the polygon style table. (1 <= PINDEX <= 255)

DENSTY [REAL; Output]
The density of the polygon interior fill. (-1.0 <= DENSTY <= 1.0)

ORIENT [REAL; Output]
The orientation of polygon interior fill. This is the angle (degrees) that fill lines make with the horizontal axis of the view surface. (-90.0 <= ORIENT <= 90.0)

EDGE [INTEGER; Output]
Specification controlling whether display of the edge of the polygon set is permitted.

EDGE = 0 - No polygon edge display.

= 1 - Polygon edge display permitted. Display of particular edge segments depends on a parameter within the call (ZPGDD or ZPGDI) used to draw the polygon.

ZIPST returns the device-dependent contents of the polygon style table entry specified in PINDEX.

For a full explanation of polygon style, see ZDPST in this chapter.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) Dummy polygon routines were loaded.
   ACTION: Call ignored.

4) The value of PINDEX is outside the range permitted by DGL (1 to 255).
   ACTION: Call ignored.

5) The value of PINDEX is outside the range supported by the graphic display device.
   ACTION: Call ignored.

ZKEND

PURPOSE: To disable the enabled keyboard device.

CALLING SEQUENCE: CALL ZKEND { no parameters }

ZKEND terminates and disables the enabled keyboard device. It transmits any termination sequence required by the device and releases all resources being used by the device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0.

ZKEND is the complementary routine to ZKINT.

If a keyboard device is used, ZKEND should be called before the application program is terminated.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The logical keyboard device is not currently enabled.
   ACTION: Call ignored.

ZKYBD

PURPOSE: To return a string from the enabled keyboard device.

CALLING SEQUENCE: CALL ZKYBD (ECHO, MAX, ACTUAL, STRING)

ECHO	[INTEGER; Input] The level of input echoing. Possible values are: ECHO = 0 - No echoing = 1 to 255 - Echo on the keyboard device
MAX	[INTEGER; Input] The maximum number of characters to be returned in the array, STRING. (1 <= MAX <= 132)
ACTUAL	[INTEGER; Output] The actual number of characters read from the keyboard device and returned in array STRING. (0 <= ACTUAL <= MAX)
STRING	[INTEGER; Array; Output] The array containing the entered character string returned to the application program. The string is in Packed ASCII format.

ZKYBD reads text information from the enabled keyboard device. Following the call to ZKYBD, the operator enters text information on the enabled keyboard device and activates the keyboard input terminator (e.g., carriage return) to complete the call.

The terminator is not returned in STRING nor is it included in the character count returned in ACTUAL. If the terminator is activated without entering any text, ACTUAL will be returned as zero. If more than MAX characters are entered, only the first MAX characters will be returned to the application program.

The input characters are returned in Packed ASCII format and filled with trailing blanks up to MAX characters.

Note that MAX is the maximum number of characters to be returned in the array, STRING. If MAX is less than 1 or greater than 132, the call will be ignored. It is the calling program's responsibility to insure that STRING is dimensioned large enough to hold MAX characters.

ZKYBD

The number of echoes supported by a keyboard device and the correlation between ECHO and the type of echoing performed is device dependent. Most keyboard devices support at least one form of echoing. Possible echoes are displaying the input as it is entered or beeping. Refer to the *Device Handlers Manual* to determine the number and type of echoes supported by a specific keyboard device. If ECHO is not supported by the enabled keyboard device, then the call will be treated as if ECHO = 1.

ZKYBD insures that the picture is current before requesting input from the keyboard device.

The keyboard device should be enabled for input by calling ZKINT before ZKYBD. The keyboard device is disabled by calling ZKEND.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The keyboard device is not enabled.
   ACTION: Call ignored

3) MAX out of range.
   ACTION: Call ignored.

4) ECHO is out of range.
   ACTION: Treated as if ECHO = 1.

ZLEND

PURPOSE: To disable the enabled locator device.

CALLING SEQUENCE: CALL ZLEND { no parameters }

ZLEND terminates and disables the enabled locator device. It transmits any termination sequence required by the device and releases all resources being used by the device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0.

ZLEND is the complementary routine to ZLINT.

If a locator device is used, ZLEND should be called before the application program is terminated.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical locator device not currently enabled.
   ACTION: Call ignored.

ZLLIM

PURPOSE: To define the logical locator limits of the locator device.

CALLING SEQUENCE: CALL ZLLIM (XMIN, XMAX, YMIN, YMAX, IERR)

XMIN [REAL; Input]
The left side of the logical locator limits is offset from the origin of the physical locator (distance in millimeters).

XMAX [REAL; Input]
The right side of the logical locator limits is offset from the origin of the physical locator (distance in millimeters).

YMIN [REAL; Input]
The bottom of the logical locator limits is offset from the origin of the physical locator (distance in millimeters).

YMAX [REAL; Input]
The top of the logical locator limits is offset from the origin of the physical locator (distance in millimeters).

IERR [INTEGER; Output]
Return code indicating whether an error occurred.

IERR = 0 (XMIN,YMIN) and (XMAX,YMAX) are within the physical locator limits.

= 1 XMIN >= XMAX or YMIN >= YMAX Call ignored.

= 2 (XMIN,YMIN) or (XMAX,YMAX) is outside the physical locator limits. Call ignored.

= 3 The locator and graphics display are mapped to be the same physical device, so the logical locator limits may not be set independently but must correspond to the current view surface limits. Call ignored.

ZLLIM can be used to specify the portion of the physical locator device that should be used to perform locator functions. When the logical locator device is enabled (via ZLINT) the logical device limits are set to a device-dependent portion of the physical locator device. With a call to ZLLIM, the user can reset the logical locator limits by specifying a new area within the physical locator limits.

The pairs (XMIN,YMIN) and (XMAX,YMAX) define the corner points of the new logical locator limits in terms of millimeters offset from the origin of the physical locator. The exact position of the physical locator origin is device dependent (refer to the *Device Handlers Manual*).

ZLLIM should only be called while the graphics locator is enabled. ZLLIM sets the locator echo position to the default value (see ZLOCP).

If a logical locator and a logical display are associated with the same physical device, then the logical locator limits must be the same as the view surface limits. Specifically, the effects of the association with the same physical device are as follows:

1. The logical locator limits are set to be the same as those of the view surface.

2. Any call which redefines the virtual coordinate system limits will also redefine the logical locator limits. These calls are: ZASPK, ZDINT, ZDLIM, ZVIEW, ZLINT and ZWIND.

3. The logical locator limits cannot be defined by a call to ZLLIM. If the user attempts to do so IERR is set equal to 3 and the rest of the call is ignored.

By changing the logical locator limits any portion of the locator device can be addressed, with the restrictions stated above.

The logical locator limits always map directly to the view surface, therefore, distortion may result in the mapping between the logical locator and the display when the logical locator limits and the view surface have different aspect ratios. If this distortion is not desired it can be avoided by assuring that the logical locator limits maintain the same aspect ratio as that of the view surface.

ZLLIM

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The graphics locator is not enabled.
   ACTION: Call ignored.

3) XMIN >= XMAX or YMIN >= YMAX
   ACTION: IERR set to 1 and rest of call ignored.

4) One of the offsets specified is outside of the physical locator limits.
   ACTION: IERR set to 2 and rest of call ignored.

5) The logical locator and the logical display are the same physical device. The logical locator limits cannot be redefined explicitly (they must correspond to the view surface limits).
   ACTION: IERR set to 3 and rest of call ignored.

ZLOCP

PURPOSE: To define the locator echo position on the graphics display.

CALLING SEQUENCE: CALL ZLOCP (WX, WY)

WX	[REAL; Input] X coordinate of the locator echo position in world coordinates.
WY	[REAL; Input] Y coordinate of the locator echo position in world coordinates.

ZLOCP allows a programmer to define the position of the locator echo position to be used when echoing on the graphics display device. The locator echo position is the starting point for all locator echoes on the graphics display. For some of these echoes the locator echo position is also used as a fixed reference point. For example, it is used as the anchor point when a rubber band echo is performed. With this echo, the graphics cursor is initially turned on at the locator echo position. From that time on, the cursor reflects the position of the locator and a line extends from the locator echo position to the locator as it moves around the graphics display. The sample locator function, ZSLOC, does not use the locator echo position.

ZLOCP should only be called while the graphics display and locator are initialized. If the point specified is outside of the window limits, the call is ignored.

The default locator echo position is the center of the window. When the locator is initialized, the locator echo position is set to the default value. When a call is made which affects the viewing transformations or the logical locator limits, the locator echo position is set to the default value. The calls which cause this are ZASPK, ZDINT, ZDLIM, ZLINT, ZLLIM, ZWIND and ZVIEW.

Once the locator echo position is set, it retains this value until the next call to ZLOCP or until a call is made which resets it to the default value.

ZLOCP

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The graphics display is not enabled.
   ACTION: Call ignored.

3) The locator device is not enabled.
   ACTION: Call ignored.

4) (WX,WY) is outside of the limits of the window.
   ACTION: Call ignored.

ZLPMM

PURPOSE: To convert from world coordinates to millimeters on the locator surface.

CALLING SEQUENCE: CALL ZLPMM (WX, WY, MMX, MMY)

WX [REAL; Input]
The X coordinate of the point to be converted from world coordinates to millimeters on the physical locator.

WY [REAL; Input]
The Y coordinate of the point to be converted from world coordinates to millimeters on the physical locator.

MMX [REAL; Output]
The offset of the world point (WX,WY) in millimeters from the origin of the physical locator device.

MMY [REAL; Output]
The offset of the world point (WX,WY) in millimeters from the origin of the physical locator device.

ZLPMM converts the world coordinate point (WX,WY) to the point (MMX,MMY), expressed in millimeters from the origin of the physical locator device. The location of this origin is device dependent. Refer to the *Device Handlers Manual* for the position of the origin on a specific device.

Since the origin of the world coordinate system need not correspond to the origin of the physical locator device, converting the point (0.0,0.0) in the world coordinate system may not result in the value (0.0,0.0) offset from the physical locator device's origin.

ZLPMM will take any world coordinate point, inside or outside the current window, and convert it to a point offset from the physical locator's origin.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The graphics locator is not enabled.
   ACTION: Call ignored.

ZLSTL

PURPOSE: To set the linestyle attribute for all primitives except polygon interior fill and markers.

CALLING SEQUENCE: CALL ZLSTL (LSTYLE)

LSTYLE [INTEGER; Input]
Desired linestyle to be applied to all subsequent output primitives except polygon interior fill and markers. LSTYLE is the index to an entry in a device-dependent linestyle table. (1 <= LSTYLE <= 255)

Every graphics device handler in DGL has a linestyle table that cannot be changed. The size of the linestyle table will vary with the device. Every linestyle table has at least an entry at index 1 which specifies solid lines. Linestyle 1 is the default linestyle set at DGL initialization and display initialization.

The user who does not wish to use the default linestyle 1 may choose other entries in the linestyle table to be used instead. Refer to the *Device Handlers Manual* for more information on supported linestyles.

ZLSTL chooses the linestyle index to be used for all primitives except markers and polygon interior fill lines. Markers are always drawn with solid lines. The user chooses the linestyle for polygon interior fill lines using ZPILS.

A linestyle is a recurring pattern of dots, blanks, short line segments and long line segments. DGL can draw any linestyle in any of three ways: start-adjusted, continuous, or vector-adjusted. However, the capability to draw in any or all of these linestyles is device dependent. On some devices, for instance, all linestyles are drawn in a continuous manner.

START-ADJUSTED linestyles always start the drawing cycle at the beginning of the vector. Thus if the linestyle starts with a pattern, each vector drawn will start with that pattern. Likewise, if the current linestyle starts with a space and then a dot, each vector will be drawn starting with a space and then a dot. In this case if the vectors are short, they might not appear at all.

CONTINUOUS linestyles are generated such that the pattern will be started with the first vector drawn. Subsequent vectors will be continuations of the pattern. Thus, it may take several vectors to complete one cycle of the pattern. This type of linestyle is useful for drawing smooth curves, but does not necessarily draw either end point of a vector. A side effect of this type of linestyle is if a vector is small enough it might be composed only of the space between points or dashes in the linestyle. In that case, the vector may not appear on the graphics display at all.

VECTOR-ADJUSTED linestyles treat each vector individually. Individual treatment guarantees that a solid component of the dash pattern will be generated at both ends of the vector. Thus, the endpoints of each vector will be clearly identifiable. This type of linestyle is good for drawing rectangles. The integrity of the linestyle will degenerate with very small vectors; since some component of the dash pattern must appear at both ends of the vector, the entire vector for a short vector will often be drawn as solid.

Figure 2-7 illustrates how one pattern would be displayed using various different linestyle types.

Note that an entire vector might not be drawn if it is smaller than the blank spaces in the selected linestyle.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) The value of LSTYLE is outside the range permitted by DGL (1 to 255).
   ACTION: Linestyle 1 is used.

4) The value of LSTYLE is outside the range supported by the graphic display device.
   ACTION: Linestyle 1 is used.

ZLWID

PURPOSE: Sets the linewidth attribute for all primitives except polygon interior fill and markers.

CALLING SEQUENCE: CALL ZLWID (LWIDTH)

LWIDTH	[INTEGER; Input] Desired linewidth to be applied to all subsequent output primitives except polygon interior fill and markers. LWIDTH is the index to an entry in a device-dependent linewidth table. (1 <= LWIDTH <= 255)

Every graphics device handler in DGL has a linewidth table that cannot be changed. The size of the linewidth table will vary with the device. Every linewidth table has at least an entry at index 1 which specifies the thinnest linewidth available. Linewidth 1 is the default linewidth set at DGL initialization and display initialization.

The user who does not wish to use the default linewidth 1 may choose other entries in the linewidth table to be used instead. For devices that support multiple linewidths, the linewidth increases as LWIDTH does until the device supported maximum is reached. For example, LWIDTH = 1 specifies the thinnest, LWIDTH = 2 specifies the next wider linewidth, and so on. Refer to the *Device Handlers Manual* for more information on supported linewidths.

Markers and polygon interior fill are always drawn with the thinnest linewidth available.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) Graphics display is not enabled.
   ACTION: Call ignored.

3) LWIDTH not supported by the graphics display.
   ACTION: Linewidth attribute set to 1 (thinnest).

4) LWIDTH out of range (1-255).
   ACTION: Linewidth attribute set to 1 (thinnest).

ZMARK

PURPOSE: To display a marker symbol at the current position.

CALLING SEQUENCE: CALL ZMARK (MARKNO)

MARKNO [INTEGER; Input]
An integer that identifies a marker, in the range 1-255. There are 19 device-independent markers. They are:

1 - .	7 - *rectangle*	13 - 3
2 - +	8 - *diamond*	14 - 4
3 - *	9 - *rectangle with cross*	15 - 5
4 - O	10 - 0	16 - 6
5 - X	11 - 1	17 - 7
6 - *triangle*	12 - 2	18 - 8
		19 - 9

Marker numbers 20 to 255 are device dependent.

ZMARK displays a marker designated by MARKNO centered on the starting position. The starting position is left unchanged at the completion of this call.

If the marker number (MARKNO) is greater than the number of distinct marker symbols supported by the graphic display device, then marker 1 (".") is used. Call ZIWS to find out the number of markers available on the graphics device. Refer to the *Device Handlers Manual* for special markers supported by specific devices. Depending on the capabilities of a device, DGL uses either hardware or software to generate a marker symbol. Since the DGL system does not do software clipping, placing a marker partly or completely outside the window limits will produce results that are device dependent.

The size and orientation of markers are fixed and not affected by viewing transformations. The size of markers is device dependent and cannot be changed. Typically, marker size is the smallest character size available on a device.

Only the attributes of color and highlighting apply to markers. However, the markers will appear with the attributes only if the graphics device is capable of applying them to markers. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ZMARK

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to initialize a device.

3) The value of MARKNO is outside the range permitted by DGL (1 to 255).
   ACTION: Marker 1 (".") is used.

4) The value of MARKNO is outside the range supported by the graphic display device.
   ACTION: Marker 1 (".") is used.

ZMCUR

PURPOSE: To make the picture current.

CALLING SEQUENCE: CALL ZMCUR { no parameters }

The graphics display surface can be made current at any time with a call to ZMCUR. This insures that all previously generated primitives have been sent to the graphics display device. Due to operating system delays, all picture changes may not have been displayed on the graphics display upon return to the calling program. ZMCUR is most often used in system buffering mode (see ZBMOD) to make sure that all output has been sent to the graphics display device when required.

Before performing any non-DGL input or output, such as a FORTRAN READ or WRITE statement to a DGL device, it is essential that all of the previously generated output primitives be sent to the device. If immediate visibility is the current timing mode, all primitives will be sent to the device before completion of the call to generate them, but if system buffering is used, ZMCUR should be called before performing any non-DGL I/O.

When changing the timing mode to immediate visibility, the picture is made current. The following routines implicitly make the picture current - ZALPH, ZBUTN, ZKYBD, ZSLOC, ZSVAL, ZWLOC, ZWVAL, ZIESC, ZDEND, ZAINT, ZBINT, ZKINT, ZLINT and ZVINT.

A call to ZMCUR can be made at any time within an application program to insure that the image is fully displayed. ZMCUR does not modify the current timing mode.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Graphics display not enabled.
   ACTION: Call ignored.

ZMOVE

PURPOSE: To set the starting position to the world coordinate position specified.

CALLING SEQUENCE: CALL ZMOVE (WX, WY)

WX	[REAL; Input] X coordinate of the new starting position in world coordinates.
WY	[REAL; Input] Y coordinate of the new starting position in world coordinates.

ZMOVE specifies a new starting position. It does this by setting the value of the starting position to the world coordinate system point specified by (WX,WY). This point will then be the starting position of the next output primitive.

Since the DGL system does not perform software clipping, setting the starting position to a point outside of the current view surface limits will produce device-dependent results.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The logical graphics display is not currently enabled.
   ACTION: Call ignored.

ZNEWF

PURPOSE: Performs a new-frame-action on the graphics display.

CALLING SEQUENCE: CALL ZNEWF { no parameters }

The DGL system provides the capability to make the picture current and then clear the graphics display of all output primitives. This new-frame-action has different connotations for each graphics display surface. On CRT devices such as graphics terminals, the screen will be erased. Plotters with page advance will advance the paper. On devices such as drum plotters or fixed page plotters, a call to ZNEWF only makes the picture current.

The buffered new-frame-action bit on the ZDINT call can affect the timing of the clearing of the graphics display (see ZDINT for use of this bit and the *Device Handlers Manual* for its effect on a particular device).

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The logical graphics display is not currently enabled.
   ACTION: Call ignored.

ZOESC

PURPOSE: To perform a device-dependent output escape function on the graphics display device.

CALLING SEQUENCE: CALL ZOESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)

OPCODE [INTEGER; Input]
The opcode of the output escape function to be performed. The opcodes supported for a given device are described in the *Device Handlers Manual*.

ISIZE [INTEGER; Input]
The number of integer parameters required by the selected output escape function.

RSIZE [INTEGER; Input]
The number of real parameters required by the selected output escape function.

ILIST [INTEGER; Array; Input]
An array containing an ISIZE number of integer elements. If ISIZE = 0, ILIST is a dummy parameter.

RLIST [REAL; Array; Input]
An array containing an RSIZE number of real elements. If RSIZE = 0, RLIST is a dummy parameter.

IERR [INTEGER; Output]
Code indicating whether the output escape function was performed. Possible values and their meanings are:

IERR = 0 Output escape function successfully sent to the device.

= 1 OPCODE not supported by the graphics display device. Call ignored.

= 2 ISIZE is not equal to the number of required integer parameters. Call ignored.

= 3 RSIZE is not equal to the number of required real parameters. Call ignored.

ZOESC allows an application program to access special hardware features on a graphics display device. The desired output escape function is specified by OPCODE. Possible functions may change the hardware character set or draw a circle using a device's circle generator. Refer to the *Device Handlers Manual* for the output escape functions supported by a specific graphics display device and the parameters required. If the OPCODE specified is not supported by a particular device or if RSIZE or ISIZE are specified incorrectly, an error code will be returned and the call will be ignored.

The type of information sent to the graphics display device is determined by the value of OPCODE. The thousands digit of OPCODE specifies the number of integer values returned in ILIST; the hundreds digit specifies the number of real values returned in RLIST. It is the application program's responsibility to insure that ILIST is dimensioned at least as large as specified by ISIZE, and that RLIST is dimensioned at least as large as specified by RSIZE. If ISIZE or RSIZE is not exactly the size required by a particular output escape function then the call will be ignored and an error will be returned in IERR.

Output escape functions only apply to graphics display devices. Output escape functions can be used anywhere in an application program while the graphics display device is enabled. Due to the fact that the DGL system does not check the parameters passed by ZOESC, the user must insure that the data passed to the device is valid.

The starting position may be altered by a call to ZOESC. Refer to the *Device Handlers Manual* for the effect of various output escape functions on a particular device.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The logical graphics display is not enabled.
   ACTION: Call ignored.

3) The specified OPCODE is not supported by the graphics display device.
   ACTION: IERR set to 1 and rest of call ignored.

4) ISIZE is not equal to the required number of integer parameters.
   ACTION: IERR set to 2 and rest of call ignored.

5) RSIZE is not equal to the required number of real parameters.
   ACTION: IERR set to 3 and rest of call ignored.

ZPGDD

PURPOSE: To display a polygon set in a device-dependent manner.

CALLING SEQUENCE: CALL ZPGDD (NPOINT,XVEC,YVEC,OPCODE)

NPOINT [INTEGER; Input]
The number of values in each array: XVEC, YVEC and OPCODE.

XVEC,YVEC [REAL; Array; Input]
XVEC and YVEC are real arrays that sequentially specify the vertices of all the members of a polygon set. The vertices for the first member polygon must be at the start of these arrays, followed by the vertices for the second member polygon, and so on.

OPCODE [INTEGER; Array; Input]
Control values that define member polygons and specify which segments of their edges will be displayed.

OPCODE(i) = 0 - Do not display the edge segment from the (i-1)th vertex to the ith vertex of the polygon set.

1 - Display the edge segment from the (i-1)th vertex to the ith vertex.

2 - The ith vertex is the first vertex of a member polygon.

No edge segments are displayed if EDGE = 0 in the polygon style.

Note: OPCODE (1) must be equal to 2 otherwise DGL will ignore the call.

ZPGDD draws polygon sets in the current polygon style as accurately as possible, using the hardware capabilities of the graphics device. ZPGDI should be used to request that polygon sets be produced in a device-independent manner. Generally, ZPGDD draws polygon sets with greater speed and more economical use of memory, and ZPGDI draws them with greater accuracy and uniformity.

Polygon fill capabilities can vary widely between devices. A device may have no filling capabilities at all, may be able to perform only solid fill, or may be able to fill polygons with different fill densities and at different fill line orientations. Some devices may be able to apply their capabilities to single rectangles only, while some may handle the full range of concave and convex polygon sets. ZPGDD tries to match the device capabilities to the request. If the device cannot fill the request at all (e.g., the device only handles rectangles and the request is for a filled complex polygon set), the polygon will not be filled.

In the case where the polygon style specifies nondisplay of edges, to provide some visible output, ZPGDD will outline the polygon using the polygon interior fill attributes. However, only those edge segments specified as displayable by OPCODE will be drawn. Refer to the *Device Handlers Manual* for a description of the polygon capabilities of a particular device. DGL automatically closes each member polygon with a nonvisible edge segment if its last vertex does not coincide with its first. To display the closing edge segment, the first and last vertices must be coincident and OPCODE for the last vertex must equal 1.

The DGL system does not perform any software clipping on output primitives. If any of the points specified in XVEC and YVEC are outside of the current view surface, then the resulting picture may not be well defined because each device handles out of range data in a different manner.

At the completion of this call the starting position is set to the first vertex of the last member polygon in the set.

The attributes of color, polygon interior color, highlighting, linestyle, polygon interior linestyle, polygon style and linewidth apply to polygons. However, the polygons will appear with these attributes only if the graphics device is capable of applying them to polygons. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) On the HP 1000, dummy polygon routines were loaded.
   ACTION: Call ignored.

4) NPOINT is a non-positive integer.
   ACTION: Call ignored.

5) More vertices are specified in XVEC and YVEC than the current graphic display device can handle.
   ACTION: The polygon set is outlined.

6) OPCODE(1) does not equal 2.
   ACTION: Call ignored.

7) OPCODE(i) (i > 1) does not equal 0, 1, or 2.
   ACTION: The polygon set is drawn but the results are unpredictable.

8) An interior fill line crosses the edge of the polygon set too many times. See the system dependent supplement for a description of applicable limits and the *Device Handlers Manual* for the display being used.
   ACTION: The fill line is not drawn.

ZPGDI

PURPOSE: To display a polygon set in a device-independent manner.

CALLING SEQUENCE: CALL ZPGDI (NPOINT,XVEC,YVEC,OPCODE)

NPOINT [INTEGER; Input]
The number of values in each array: XVEC, YVEC and OPCODE.

XVEC,YVEC [REAL; Array; Input]
XVEC and YVEC are real arrays that sequentially specify the vertices of all the members of a polygon set. The vertices for the first member polygon must be at the start of these arrays, followed by the vertices for the second member polygon, and so on.

OPCODE [INTEGER; Array; Input]
Control values that define member polygons and specify which segments of their edges will be displayed.

OPCODE(i) = 0 - Do not display the edge segment from the (i-1)th vertex to the ith vertex of the polygon set.

1 - Display the edge segment from the (i-1)th vertex to the ith vertex.

2 - The ith vertex is the first vertex of a member polygon.

No edge segments are displayed if EDGE = 0 in the polygon style.

Note: OPCODE (1) must be equal to 2 otherwise DGL will ignore the call.

ZPGDI

ZPGDI draws polygon sets in a device-independent manner. This is done in software unless the device is capable of producing polygon sets in the entire polygon style range described below. ZPGDD draws polygon sets using the hardware capabilities of the device. Generally, ZPGDI draws polygon sets with greater accuracy and uniformity, and ZPGDD draws them with greater speed and more economical use of memory.

DGL may place a limit on the number of polygon vertices which the user may specify (see the system dependent supplement). If the number of vertices specified is greater than the number that can be filled, the polygon will not be filled. In the case where the polygon style specifies nondisplay of edges, to provide some visible output, ZPGDI will outline the polygon using the polygon interior fill attributes. However, only those edge segments specified as displayable by OPCODE will be drawn.

DGL automatically closes each member polygon with a non-visible edge segment if its last vertex does not coincide with its first. To display the closing edge segment, the first and last vertices must be coincident and OPCODE for last vertex must equal 1.

The DGL system does not perform any software clipping on output primitives. If any of the points specified in XVEC and YVEC are outside of the current view surface, then the resulting picture may not be well defined because each device handles out of range data in a different manner.

At the completion of this call, the starting position is set to the first vertex of the last member polygon in the set.

The attributes of color, polygon interior color, highlighting linestyle, polygon interior linestyle and linewidth apply to polygon primitives. However, the polygons will appear with these attributes only if the graphics device is capable of applying them to polygons. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) On the HP 1000, dummy polygon routines were loaded.
ACTION: Call ignored.

4) NPOINT is a non-positive integer.
ACTION: Call ignored.

5) More vertices were specified in XVEC than can be filled. See the system dependent supplement for a description of applicable limits.
ACTION: The polygon set is outlined.

6) OPCODE(1) does not equal 2.
ACTION: Call ignored.

7) OPCODE(i) (i > 1) does not equal 0, 1, or 2.
ACTION: The polygon set is drawn but the results are unpredictable.

8) An interior fill line crosses the edge of the polygon set too many times. See the system dependent supplement for a description of applicable limits and the *Device Handlers Manual* for the display being used.
ACTION: The fill line is not drawn.

ZPICL

PURPOSE: To set color attribute for polygon interior fill.

CALLING SEQUENCE: CALL ZPICL (COLOR)

COLOR	[INTEGER; Input] Desired color for the interior fill of polygon sets. COLOR is the index to an entry in a device-dependent color table. (0 <= COLOR <= 32767)

ZPICL sets the color index for the interior fill of subsequent polygon sets. ZCOLR sets the color index for all other primitives, including the edges of polygon sets.

Every graphics display device handler in DGL has its own default color table. The size of the color table will vary with the device. Every default color table has at least one entry at index 1. Display devices for which there is a background color have an entry at index 0 which is the background color. Color 1 is the default color set at DGL initialization and display initialization for all primitives.

The user who does not wish to use the default color 1 for polygon interior fill may choose other entries in color table to be used instead. Any entry in the color table may be chosen by ZPICL. If ZPICL specifies Color 0, the background color, results will be device dependent. For example, on plotters, the pen will be put away. On raster displays, lines will be drawn in the background color allowing the user to erase existing primitives by drawing over them.

It is possible to define an entry in a device color table which the device is not capable of displaying. When such an entry is chosen using ZPICL, DGL attempts to match the requested color with the closest color of which the device is capable. The resulting color may duplicate the color of some other entry in the table.

On some devices, the ability of the device to display a color will depend on the type of primitive being displayed. This will frequently happen on color displays which are capable of filling polygon interiors in a wide range of colors but which have a narrow selection for all other primitives. In this case, line color, for example, may only approximate the color of the entry selected by ZCOLR while polygon set interior color may be much closer to the color selected with ZPICL

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) On the HP 1000, dummy polygon routines were loaded.
   ACTION: Call ignored.

4) The value of COLOR is outside the range of the system (0 to 32767).
   ACTION: Color index 1 is used.

5) The value of COLOR is outside the range supported by the graphic display device.
   ACTION: Color index 1 is used.

ZPILS

PURPOSE: To set the linestyle attribute for polygon interior fill.

CALLING SEQUENCE: CALL ZPILS (LSTYLE)

LSTYLE	[INTEGER; Input] Desired linestyle for the interior fill lines of polygon sets. LSTYLE is the index to an entry in a device-dependent linestyle table. (0 <= LSTYLE <= 255)

Every graphics device handler in DGL has a linestyle table that cannot be changed. The size of the linestyle table will vary with the device. Every linestyle table has at least an entry at index 1 which specifies solid lines. Linestyle 1 is the default linestyle set at DGL and display initialization for all primitives.

The user who does not wish to use the default linestyle 1 may choose other entries in the linestyle table to be used instead. Refer to the *Device Handlers Manual* for more information on supported linestyles.

ZPILS sets the linestyle index for interior fill lines of subsequent polygon sets. ZLSTL chooses the linestyle index for all other primitives except markers, including the edges of polygon sets.

A linestyle is a recurring pattern of dots, blanks, short line segments and long line segments. DGL can draw any linestyle in any of three ways: start-adjusted, continuous or vector-adjusted. However, the capability to draw in any or all of these linestyles is device dependent. On some devices, for instance, all linestyles are drawn in a continuous manner.

START-ADJUSTED linestyles always start the drawing cycle at the beginning of the vector. Thus, if the linestyle starts with a pattern, each vector drawn will start with that pattern. Likewise, if the current linestyle starts with a space and then a dot, each vector will be drawn starting with a space and then a dot. In this case if the vectors are short, they might not appear at all.

CONTINUOUS linestyles are generated such that the pattern will be started with the first vector drawn. Subsequent vectors will be continuations of the pattern. Thus, it may take several vectors to complete one cycle of the pattern. This type of linestyle is useful for drawing smooth curved but does not necessarily draw either end point of a vector. A side effect of this type of linestyle is if a vector is small enough it might be composed only of the space between points or dashes in the linestyle. In that case, the vector may not appear on the graphics display at all.

VECTOR-ADJUSTED linestyles treat each vector individually. Individual treatment guarantees that a solid component of the dash pattern will be generated at both ends of the vector. Thus, the endpoints of each vector will be clearly identifiable. This type of linestyle is good for drawing rectangles. The integrity of the linestyle will degenerate with very small vectors; since some component of the dash pattern must appear at both ends of the vector, the entire vector will often be drawn as solid.

Figure 2-7 illustrates how one pattern would be displayed using various different linestyle types.

Note that an entire vector might not be drawn if it is smaller than the blank spaces in the selected linestyle.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) On the HP 1000, dummy polygon routines were loaded.
   ACTION: Call ignored.

4) The value of LSTYLE is outside the range permitted by DGL (1 to 255).
   ACTION: Linestyle index 1 is used.

5) The value of LSTYLE is outside the range supported by the graphic display device. ACTION: Linestyle index 1 is used.

ZPOLY

PURPOSE: To draw a connected line sequence starting at the specified point.

CALLING SEQUENCE: CALL ZPOLY (NPTS, XVEC, YVEC)

NPTS	[INTEGER; Input] The number of points in the connected line sequence. NPTS = (number of lines) + 1.
XVEC	[REAL; Array; Input] The array of X coordinate values for each line in the sequence.
YVEC	[REAL; Array; Input] The array of Y coordinate values for each line in the sequence.

The subroutine ZPOLY provides the capability to draw a series of connected lines starting at the specified point. A complete object can be drawn by making one call to this subroutine. This call first sets the starting position to be the first elements in the XVEC,YVEC arrays. The line sequence begins at this point and is drawn to the second element in each array, then to the third and continues until NPTS-1 lines are drawn.

This call is equivalent to the following sequence of calls:

```
CALL ZMOVE (XVEC(1),YVEC(1))
CALL ZDRAW (XVEC(2),YVEC(2))
CALL ZDRAW (XVEC(3),YVEC(3))
             :
             :
CALL ZDRAW (XVEC(NPTS),YVEC(NPTS))
```

The DGL system does not perform software clipping on output primitives. If any of the points specified in XVEC and YVEC are outside of the current view surface limits, the resulting image will be device dependent.

The starting position is set to (XVEC(NPTS), YVEC(NPTS)), at the completion of this call.

Specifying only one element, or NPTS equal to 1, causes a move to be made to the world coordinate point (XVEC(1),YVEC(1)).

It is the application program's responsibility to insure that XVEC, and YVEC are all dimensioned to at least NPTS and that at least NPTS values are contained in each array.

Depending on the nature of the current linestyle nothing may appear on the graphics display. See ZLSTL for a complete description of how linestyle affects a particular point or vector.

The primitive attributes of color, highlighting, linestyle, and linewidth apply to polylines. However, the polyline will appear with these attributes only if the graphics device is capable of applying them to polylines. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Graphics display is not enabled.
   ACTION: Call ignored.

3) NPTS <= 0
   ACTION: Call ignored.

ZPSTL

Purpose: To set the polygon style for polygon sets.

CALLING SEQUENCE: CALL ZPSTL (PINDEX)

PINDEX [INTEGER; Input]
Desired style for polygon sets. PINDEX is an index into a device-dependent polygon style table.
(1 <= PINDEX <= 255)

Every graphics device handler in DGL has a polygon style table. ZPSTL sets the style attribute for subsequent polygon sets. The first 16 styles are the same for all devices (see Figure 2-9). See ZDPST for an explanation of polygon style.

1) The DGL system is not initialized.
ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) No graphic display device is enabled.
ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) On the HP 1000, dummy polygon routines were loaded.
ACTION: Call ignored.

4) The value of PINDEX is outside the range permitted by DGL (1 to 255).
ACTION: Polygon style 1 is used.

5) The value of PINDEX is outside the range supported by the graphic display device.
ACTION: Polygon style 1 is used.

ZSLOC

PURPOSE: To sample the enabled locator device and return the locator point without waiting for operator response.

CALLING SEQUENCE: CALL ZSLOC (ECHO, WX, WY)

ECHO — [INTEGER; Input]
The level of input echoing. Legal values are:

ECHO = 0 - No echo.

= 1 to 255 - Echo on the locator device.

WX, WY — [REAL; Output]
The world coordinate point returned from the enabled locator. This point may be outside the current logical locator limits.

ZSLOC returns the current world coordinate value of the locator in (WX,WY) without waiting for an operator response. To wait for an operator response before returning the value of the locator, see ZWLOC. ZSLOC implicitly makes the picture current before sampling the locator. The locator device should be enabled prior to calls to ZSLOC (see ZLINT). The locator device is disabled by calling ZLEND.

The number of echoes supported by a locator device and the correlation between ECHO and the type of echoing performed is device dependent. Most locator devices support at least one form of echoing. Possible echoes are beeping, or displaying the point sampled. Refer to the *Device Handlers Manual* to determine the number and type of echoes supported by a specific locator device. If ECHO is larger than the number of echoes supported by the enabled locator device, then ECHO 1 will be used.

Locator echoing can only be performed on the locator device. The locator echo position is not used in conjunction with any echoes performed while sampling a locator.

ZSLOC

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical locator device not currently enabled.
   ACTION: Call ignored.

3) ECHO out of range.
   ACTION: Treated as if ECHO = 1.

ZSVAL

PURPOSE: To sample the enabled valuator device and return the value of the subvaluator specified without waiting for operator response.

CALLING SEQUENCE: CALL ZSVAL (ECHO, SUBVAL, VALUE)

ECHO [INTEGER; Input]
The level of input echoing. Legal values are:

ECHO = 0 - No echo.

1 to 255 - Echo on the valuator device.

SUBVAL [INTEGER; Input]
The subvaluator which is to be sampled on the valuator device. (1 <= SUBVAL <= 255)

VALUE [REAL; Output]
The value returned from the subvaluator specified by SUBVAL on the enabled valuator device. (0.0 <= VALUE <= 1.0)

ZSVAL returns the current value of the subvaluator in the variable VALUE without waiting for an operator response. VALUE will always range from 0.0 to 1.0 inclusive. To wait for an operator response before returning the value of the valuator, see ZWVAL.

A valuator device is considered to be a collection of one or more subvaluator devices. SUBVAL is used to specify the subvaluator to be used when sampling the valuator.

For example, the valuator device is simulated on the HP 9111 Data Tablet by reading the position of the stylus on the platen. If SUBVAL is set to 1, the X position of the stylus is returned, with 0.0 represented as the extreme left side of the platen, and 1.0 represented as the extreme right side. If SUBVAL is set to 2, the Y position of the cursor is returned, with 0.0 represented as the bottom of the platen, and 1.0 represented as the top.

ZSVAL

The number of subvaluators supported on a valuator device is device dependent but all valuators support at least one subvaluator. If SUBVAL is greater than the number of supported subvaluators on the enabled valuator device, then the call will proceed as if SUBVAL were equal to 1. ZIWS with OPCODE equal to 7051 can be used to determine the number of valuators supported.

The number of echoes supported by a valuator device and the correlation between ECHO and the type of echoing performed is device dependent. Most valuator devices support at least one form of echoing. Possible echoes are beeping or displaying the value sampled. Refer to the *Device Handlers Manual* to determine the number and type of echoes supported by a specific valuator device. If ECHO is larger than the number of echoes supported by the enabled valuator device, then ECHO 1 will be used. Valuator echoing can only be performed on the valuator device.

ZSVAL implicitly makes the picture current before sampling valuator. The logical valuator device should be enabled prior to calls to ZSVAL (see ZVINT). The valuator is disabled by calling ZVEND.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical valuator device not currently enabled.
   ACTION: Call ignored.

3) ECHO out of range.
   ACTION: Treated as if ECHO = 1.

4) SUBVAL not supported by the valuator device.
   ACTION: Treated as if SUBVAL = 1.

ZTEXT

PURPOSE: To output graphics text on the graphics display.

CALLING SEQUENCE: CALL ZTEXT (NCHARS, STRING)

NCHARS [INTEGER; Input]
The number of characters to be output.
(1 <= NCHARS <= 132).

STRING [INTEGER; Array; Input]
Contains the characters to be output in Packed ASCII format.

ZTEXT generates text on the graphics display. Text will be produced by a device's hardware character generator, when present. Otherwise, text is simulated in software.

A call to ZTEXT will output the characters contained in the array STRING to the graphics display. When the text string is output, the starting position is initially at the lower left-hand corner of the first character in STRING. Text is normally output from left to right and is printed vertically with no slant. Some devices support other text directions and character slants which can be accessed through escape functions (see ZOESC). Refer to the *Device Handlers Manual* for additional text functions which may be supported by a specific graphics display device.

After completion of this call, the starting position is left in a device-dependent location. Refer to the *Device Handlers Manual* for the effect of subsequent text primitives. A call to ZMOVE, ZPOLY, ZPGDI or ZPGDD should be made after this call to update the starting position.

The actual character size used to generate text is device dependent since not all devices support continuously variable character sizes. If the actual character cell size requested is not available on the graphics display, then the "smaller best fit" size available will be used. The smaller best fit character size is defined to be:

- The largest character whose cell height is less than or equal to the requested height and whose cell width is less than or equal to the requested width, or,
- If the size requested is smaller than all available sizes, then the smallest available size will be used.

Since the DGL system does not perform software clipping, any portion of the string that lies either partially or entirely outside of the limits of the window will generate device-dependent results.

The attributes of color, highlighting linestyle, linewidth and character size apply to text primitives. However, the text will appear with these attributes only if the graphics device is capable of applying them to text. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical graphics display not currently enabled.
   ACTION: Call ignored.

3) NCHARS out of range (1 to 132).
   ACTION: Call ignored.

ZVEND

PURPOSE: To disable the enabled valuator device.

CALLING SEQUENCE: CALL ZVEND { no parameters }

ZVEND terminates and disables the enabled valuator device. It transmits any termination sequence required by the device and releases all resources being used by the device. The device name is set to the default device name (see the *Device Handlers Manual*), the device status is set to 0 (not enabled) and the I/O unit descriptor of the device is set to 0.

ZVEND is the complementary routine to ZVINT.

If the valuator is used, ZVEND should be called before the application program is terminated.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Logical valuator device not currently enabled.
   ACTION: Call ignored.

ZVIEW

PURPOSE: To set the boundaries of the viewport in the virtual coordinate system.

CALLING SEQUENCE: CALL ZVIEW (VXMIN, VXMAX, VYMIN, VYMAX)

VXMIN [REAL; Input]
The minimum boundary in the X direction expressed in virtual coordinates.

VXMAX [REAL; Input]
The maximum boundary in the X direction expressed in virtual coordinates.

VYMIN [REAL; Input]
The minimum boundary in the Y direction expressed in virtual coordinates.

VYMAX [REAL; Input]
The maximum boundary in the Y direction expressed in virtual coordinates.

ZVIEW sets the limits of the viewport in the virtual coordinate system. The viewport must be within the limits of the virtual coordinate system; otherwise the call will be ignored.

The initial viewport is:

(VXMIN=0.0,VXMAX=1.0,VYMIN=0.0,VYMAX=1.0)

The initial viewport is set by ZBEGN. This initial viewport is mapped onto the maximum visible square within the logical display limits. This area is called the view surface. The placement of the view surface within the logical display limits is dependent upon the device being used (see the *Device Handlers Manual*). It is generally centered on CRT displays and tubes and is usually placed in the lower left-hand corner of plotters.

By changing the limits of the viewport, an application program can display an image in several different positions on the same graphics display device. A program can make a call to ZVIEW anytime the DGL system is initialized.

The starting position is not altered by this call. Since this call redefines the viewing transformation, the starting position may no longer represent a known world coordinate position. A call to ZMOVE, ZPOLY, ZPGDD, or ZPGDI should be made after this call to update the starting position.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) XMIN >= XMAX or YMIN >= YMAX.
   ACTION: Call ignored.

3) Either of the points (XMIN, YMIN) or (XMAX, YMAX) is not within the current virtual coordinate system boundary.
   ACTION: Call ignored.

ZWIND

PURPOSE: To define the boundaries of the window.

CALLING SEQUENCE: CALL ZWIND (WXMIN, WXMAX, WYMIN, WYMAX)

WXMIN	[REAL; Input] The minimum boundary in the X direction expressed in world coordinates, i.e., the left window border.
WXMAX	[REAL; Input] The maximum boundary in the X direction expressed in world coordinates, i.e., the right window border.
WYMIN	[REAL; Input] The minimum boundary in the Y direction expressed in world coordinates, i.e., the bottom window border.
WYMAX	[REAL; Input] The maximum boundary in the Y direction expressed in world coordinates, i.e., the top window border.

ZWIND defines the limits of the window. All positional information sent to and received from the DGL system is specified in world coordinate units. This allows the application program to specify coordinates in units related to the application.

The window is linearly mapped onto the viewport specified by ZVIEW. This is done by mapping WXMIN to the minimum X viewport boundary, WXMAX to the maximum X viewport boundary, WYMIN to the minimum Y viewport boundary, and WYMAX to the maximum Y viewport boundary. If distortion of the graphics image is not desired, the aspect ratio of the window boundaries should be equal to the aspect ratio of the viewport.

Since the DGL system does not perform software clipping, the results generated by specifying coordinates outside of the defined window are device dependent.

The default window limits range from -1.0 to 1.0 on both the X and Y axis. ZBEGN sets the window to its default limits.

The starting position is not altered by this call. Since this call redefines the viewing transformation, the starting position may no longer represent a known world coordinate position. A call to ZMOVE, ZPOLY, ZPGDD, or ZPGDI should therefore be made after this call to update the starting postion.

ZWIND can be called at any time while the DGL system is initialized.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) XMIN = XMAX or YMIN = YMAX.
   ACTION: Call ignored.

ZWLOC

PURPOSE: To wait until activation of the locator button and then read from the enabled locator device.

CALLING SEQUENCE: CALL ZWLOC (ECHO, LBUTN, WX, WY)

ECHO [INTEGER; Input]
The level of input echoing. Possible values are:

0 - No echo.

1 - Echo on the locator device.

2 - Track the position of the locator on the graphics display device. The initial position of the cursor is at the locator echo position. The point returned is the locator position.

3 - Designate the position of the locator on the graphics display device with two intersecting lines. One line is horizontal with a length equal to the width of the graphics display surface. The other line is vertical with a length equal to the height of the graphics display device. The initial point of intersection is at the current locator echo position. The point returned is the locator position.

4 - Designate the endpoints of a line. One end is fixed at the locator echo position; the other endpoint is designated by the current locator position. This is usually referred to as a rubber band line. The point returned is the locator position.

5 - Designate a horizontal line. One endpoint of the line is fixed at the locator echo position; the other endpoint has the world Y coordinate of the locator echo position and the world X coordinate of the current locator position. The point returned will have the X coordinate of the locator position and the Y coordinate of the locator echo position.

6 - Designate a vertical line. One endpoint of the line is fixed at the locator echo position; the other endpoint will have the world X coordinate of the locator echo position and the world Y coordinate of the current locator position. The point returned will have the X coordinate of the locator echo position and the Y coordinate of the locator position.

7 - Designate a horizontal/vertical line. One endpoint of the line is fixed at the locator echo position. The other endpoint will have the world X coordinate of the locator echo position and the world Y coordinate of the current locator position or the world X coordinate of the current locator and the world Y coordinate of the locator echo position, depending on which point defines a longer line. If both lines are of equal length, a horizontal line will be used. As a result this echo defines either a horizontal or vertical line depending on whether the line from the locator echo position to the current locator position is closer to the horizontal or the vertical. The point returned is the endpoint of the echoed line.

8 - Designate a rectangle. The diagonal of the rectangle is the line from the locator echo position to the current locator position. The returned point will be the locator position.

9 - 255 - Echo on the locator device.

LBUTN [INTEGER; Output]
The integer value of the button used to activate the locator input. (0 <= LBUTN <= 255).

WX,WY [REAL; Output]
The world coordinate point returned from the enabled locator. The point may be outside of the current logical locator limits.

ZWLOC waits until the locator button is activated and then returns the value of the selected button and the world coordinate point (WX,WY) of the locator. Each locator device has its own set of buttons that may or may not be the same as those that comprise the button device. If an invalid button is pressed, LBUTN will be returned as 0; otherwise LBUTN will contain the value of the button pushed.

Several different types of echoing can be performed. Some echoes are performed on the locator device while others use the graphics display device. When ECHO is in the range of 2 through 8, the graphics display device will be used. All of the echoes on the graphics display start at a point on the graphics display called the locator echo position (see ZLOCP). For some of these echoes the locator echo position is also used as a fixed reference point. For example, many devices support a rubber band echo. The fixed end of the rubber band line will be at the locator echo position. Not all graphics display devices can fully perform locator echoes 2 through 8. When an echo cannot be performed it will be simulated as closely as possible. For example the HP 2623 Graphics Terminal does not support a full screen cursor echo so it is simulated using the standard graphics cursor. The actual echo performed is dependent upon the locator and graphics display devices being used. Refer to the *Device Handlers Manual* for device specific information. If ECHO is in the range 2 through 8 and the graphics display device is disabled ECHO 1 will be used.

ECHO values of 1 and 9 through 255 specify that the echo is to be performed on the locator device. Most locator devices support at least one form of echoing. Possible forms include beeping, displaying the value entered, or blinking a light each time a point is entered. If the specified echo is not supported on the enabled locator device, ECHO 1 will be used.

ZWLOC implicitly makes the picture current before requesting locator input. The logical locator device should be enabled prior to calls to ZWLOC (see ZLINT). The locator is disabled by calling ZLEND.

In some configurations of some devices the starting position may be left at a device-dependent location after this call. Refer to the *Device Handlers Manual* for details.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) The locator device is not currently enabled.
   ACTION: Call ignored.

3) ECHO is out of range.
   ACTION: Treated as if ECHO = 1.

4) ECHO requires a graphics display device and the logical graphics display is not enabled.
   ACTION: Treated as if ECHO=1.

5) Locator button activated is not valid.
   ACTION: LBUTN set to 0 and call proceeds.

ZWVAL

PURPOSE: To wait until activation of the valuator button and then read from the specified subvaluator of the enabled valuator device.

CALLING SEQUENCE: CALL ZWVAL (ECHO, SUBVAL, VBUTN, VALUE)

ECHO — [INTEGER; Input]
The level of input echoing. Possible values are:

ECHO = 0 - No echo.

= 1 to 255 - Echo on the valuator device.

SUBVAL — [INTEGER; Input]
The subvaluator which is to be read from the enabled valuator device. (1 <= SUBVAL <= 255)

VBUTN — [INTEGER; Output]
The integer value of the button used to activate the valuator input.
(0 <= VBUTN <= 255).

VALUE — [REAL; Output]
The REAL value returned from the subvaluator specified by SUBVAL on the enabled valuator device.
(0.0 <= VALUE <= 1.0)

ZWVAL waits until the valuator button is activated and then returns the value of the selected button and the value of the specified subvaluator. Each valuator device has its own set of buttons which may or may not be the same as those that comprise the button device. If an invalid button is pressed, VBUTN will be returned as 0; otherwise, the VBUTN will contain the value of the button pushed. VALUE will always range from 0.0 to 1.0. The desired echo and subvaluator are specified in the respective variables ECHO and SUBVAL.

A valuator device is considered to be a collection of one or more subvaluators. SUBVAL is used to specify which subvaluator on the enabled valuator will be used to perform the valuator function.

For example, the valuator device is simulated on the HP 9111 Data Tablet by reading the position of the stylus on the platen. If the specified value of SUBVAL is 1, the X position of the stylus is returned, with 0.0 represented as the extreme left side of the platen, and 1.0 represented as the extreme right side. If SUBVAL is specified as 2, the Y position of the cursor is returned, with 0.0 represented as the bottom of the platen, and 1.0 represented as the top.

The number of subvaluators supported on a valuator device is device dependent but all valuators support at least one subvaluator. If SUBVAL is greater than the number of supported subvaluators on the enabled valuator device, then the call will proceed as if SUBVAL were equal to 1. ZIWS can be used to determine the number of subvaluators supported.

The number of echoes supported by a valuator device and the correlation between ECHO and the type of echoing performed is device dependent. Most valuator devices support at least one form of echoing. Possible echoes are beeping, or displaying the value sampled. Refer to the *Device Handlers Manual* to determine the number and type of echoes supported by a specific valuator device. If ECHO is larger than the number of echoes supported by the enabled valuator device, then ECHO 1 will be used.

ZWVAL implicitly makes the picture current before requesting valuator input. The logical valuator device should be enabled prior to calls to ZWVAL (see ZVINT). The valuator is disabled by calling ZVEND.

In some configurations of some devices the starting position may be left at a device-dependent location after this call. Refer to the *Device Handlers Manual* for details.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The valuator device is not enabled.
   ACTION: Call ignored.

3) ECHO out of range.
   ACTION: Treated as if ECHO = 1.

4) Valuator button activated is not valid.
   ACTION: VBUTTN set to 0 and call proceeds.

Glossary

Alphanumeric display device

An output device to which non-graphics text and data may be directed. It is most often used to send prompts, give status and print other messages.

Aspect ratio

The ratio of the height to width of an area (e.g., the area of a display device, the window, the viewport or the view surface).

Attribute

See Primitive attribute.

Button input function

An input function that returns an integer value corresponding to a button or key that is activated.

Character cell

An imaginary box placed around a character which defines its dimensions. The character size attribute determines the size of the character cell.

Clipping

The elimination from view of all visible primitives or parts of primitives which lie outside the given limits.

Color model

A system within which displayable colors are defined. See also, HSL Model and RBG Model. Colors are defined by a triplet of real numbers in a range (0.0 to 1.0).

Color table

A table defining colors currently available for a graphical display device. Colors are defined by a triplet of real numbers in a range (0.0 to 1.0) using the current color model.

Default

See Initial value.

Eavesdrop device

A device connected between the host computer and a terminal which "listens" to the data sent from the host computer. When a special escape sequence is sent, the device accepts all subsequent data and does not pass it on the terminal. When a second special escape sequence is sent, the device ignores all data and passes it on the terminal.

Echoing

A mechanism for reflecting the status of an input function. Echoing is manifested in several ways as a function of the different input functions and the different physical devices being used.

Edge, member polygon

See Polygon edge.

Edge, polygon set

See Polygon edge.

Edge display

The physical expression of a polygon edge, drawn on the graphics display device; an edge has the attributes of highlighting, color, linestyle and linewidth in effect at the time of polygon generation.

Enable a device

Enabling a device makes it available for use. A device must be enabled before requests are directed to it.

Escape function

A facility within DGL which allows access to device dependent functions of a graphics display device.

Graphics display device

A peripheral which displays graphics output.

Highlighting

A primitive attribute used to emphasize the primitive of an image. Some devices highlight by blinking or by increasing line intensity, others ignore the highlighting attribute.

HSL model

A perceptual color model which represents a color space as a cylinder whose angle about its axis is the hue, height is the luminosity, and radius is the saturation. See also, Color model.

Image

A particular view of one or more objects or parts of objects. Images comprise the picture on a graphics display device.

Immediate visibility

A DGL timing mode in which picture changes requested by a DGL routine are sent to the graphics display device before the routine is exited. This is the least efficient timing mode in terms of system resource usage.

Initial value

The value of an attribute, viewing component, or characteristic of a work station which is in effect when DGL is initialized.

Inquiry

User request for the current status, value, or characteristics of the graphics environment.

Interior, polygon

See Polygon interior.

Interior Density

See Polygon interior density.

Interior Orientation

See Polygon interior orientation.

Keyboard input function

An input function which returns a text string of zero or more characters. The typical keyboard device is the alphanumeric keyboard. Keyboard input is terminated when a termination character (e.g. carriage return) is detected.

Line

A vector drawn from the current position to a specified point.

Linestyle

An output primitive attribute which controls the pattern with which lines, polylines, polygon edges and text primitives are drawn.

Linewidth

An output primitive attribute which controls the thickness with which lines, polylines, polygon edges and text primitives are drawn.

Locator input function

An input function which returns a virtual coordinate point corresponding to a location on a locator device.

Logical device

An abstraction of a typical graphics device, defined in terms of the type of data input or output. The logical devices supported by DGL are button, keyboard, locator, valuator, alphanumeric display and graphics display.

Logical display limits

The bounds of the logical display surface.

Logical display surface

The portion of a graphics display device where all output will appear.

Make-picture-current

Action which forces all currently buffered updates to be sent immediately to the display device.

Mapping

The transformation of data from one coordinate system to another.

Marker

An output primitive that is a center oriented symbol used to identify a point in the world coordinate system.

Member polygon

A polygon which is an element of a polygon set. See Polygon.

Move

Moving the starting position to a specified point without generating a line.

New-frame-action

The elimination from view of all primitives. On most devices this means the screen will be erased. On chart advance plotters the paper is advanced.

Object

The conceptual graphics entity in the application program. Objects are defined in terms of output primitives and primitive attributes. Their units are the units of the world coordinate system.

Output primitive

The basic element of an object. The output primitives which DGL supports are: move, line, polyline, polygon, marker and text. Values of the primitive attributes determine aspects of the appearance of output primitives.

Packed ASCII Format

The format in which input characters are returned or given: ASCII left-justified, one character per byte and blank filled.

Picture

A collective reference to all the images on a display device.

Pixel

The smallest element of a display surface that can be independently assigned a color or intensity; a picture element.

Polygon

An output primitive that defines a multi-sided closed figure, including the surface defined by its boundary.

Polygon boundary

The outer limit of a polygon surface. The interior of the polygon is defined to include the boundary.

Polygon edge

A characteristic of the polygon style attribute. For each entry in the polygon style table, a polygon edge is specified as present or absent. If an edge is present, it is superimposed on the boundary of the polygon and is generated with the current attributes of linestyle, highlighting, linewidth and color.

Polygon interior

The area defined by the boundary of a polygon, including the boundary itself.

Polygon interior color

A polygon attribute that defines the color of the interior of subsequently generated polygons. This attribute is an index into a device-dependent color table.

Polygon interior density

A characteristic of the polygon style attribute. For each entry in the polygon style table the interior density is specified as the ratio of filled area to total area in a polygon's interior (based on a solid linestyle).

Polygon interior orientation

One characteristic of the polygon style attribute. For each polygon style entry in the polygon style table, an interior orientation is specified. It defines whether the pattern which fills the interior of the polygon is to be hatching or cross-hatching. It also defines the angle of orientation of the fill lines to the view surface.

Polygon style

A polygon attribute that defines the density of the polygon interior, the orientation of the polygon interior, the presence or absence of a polygon edge. This attribute is an index into a work station dependent polygon style table.

Polygon style table

A device dependent table of polygon styles. Each table entry defines the polygon interior density, the polygon interior orientation, and the polygon edge flag.

Polyline

An output primitive which defines a connected sequence of lines.

Primitive

See Output primitive.

Primitive attribute

A characteristic of an output primitive, such as color, linestyle, or text font.

Requested input

An input operation which requests data and completes when an operator enters a termination command.

RGB Model

A color model which represents a color spatially as a cube that has as its three axes the primary additive colors: red, green and blue. See also, Color model.

Sampled input

An input operation which does not require operator intervention; the routine returns with the current value as soon as the input device can respond.

System buffering

A DGL timing mode in which picture changes requested by a DGL routine are subject to buffering.

Timing mode

Used to control DGL execution efficiency and the immediacy of visual changes to a graphics display device. DGL has two timing modes: immediate visibility and system buffering.

Valuator input function

An input function which returns a numeric value within the range (0.0 to 1.0) inclusive. The typical valuator is the control dial (potentiometer), but multi-position switches and toggle switches can be utilized as low-precision valuators.

Viewing operation

See Viewing transformation.

Viewing transformation

An operation which maps positions in the world coordinate system to positions in device coordinates, thereby transforming objects into images.

Viewport

The rectangular region of the view surface onto which the window will be mapped.

View surface

The largest rectangle within the logical display limits having the same aspect ratio as the virtual coordinate system.

Virtual coordinate system

A two-dimensional coordinate system representing an idealized display device. Virtual coordinates are always in the range $0.0 \leq (X,Y) \leq 1.0$.

Window

A rectangular region in world coordinates which is linearly mapped onto the viewport.

Work station

A collection of graphics devices that can be treated as a unit.

World coordinate system

The two-dimensional left-handed Cartesian coordinate system in which objects are described by the user program.

Index

A

B

C

D

E

G

H

I

K

L

M

N

P

R

S

T

V

W

Z

READER COMMENT SHEET

Device-independent Graphics Library
Programmer Reference Manual

97084-90000 June 1983

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?	**Yes [ ] No [ ]**	(If no, explain under Comments, below.)
Are the concepts and wording easy to understand?	**Yes [ ] No [ ]**	(If no, explain under Comments, below.)
Is the format of this manual convenient in size, arrangement and readability?	**Yes [ ] No [ ]**	(If no, explain or suggest improvements under Comments, below.)

Comments:

Date: _____

FROM:

Name _____

Company _____

Address _____

_____

_____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO, CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager
Hewlett-Packard Company
Engineering Productivity Division
11000 Wolfe Road
Cupertino, California 95014

FOLD FOLD

MANUAL PART NO. 97084-90000
Printed in U.S.A.
E0683

HEWLETT-PACKARD COMPANY
Engineering Productivity Division
11000 Wolfe Road
Cupertino, California 95014

The attributes of color, polygon interior color, highlighting linestyle, polygon interior linestyle and linewidth apply to polygon primitives. However, the polygons will appear with these attributes only if the graphics device is capable of applying them to polygons. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) NPOINT is a non-positive integer.
   ACTION: Call ignored.

4) The number of vertices specified in IXVEC, IYVEC exceeds 500.
   ACTION: The polygon set is outlined.

5) OPCODE(1) does not equal 2.
   ACTION: Call ignored.

6) OPCODE(i) (i > 1) does not equal 0, 1, or 2.
   ACTION: The polygon set is drawn but the results are unpredictable.

7) An interior fill line crosses the edge of the polygon set too many times. See Section I ("Polygon Intercept Buffer Size Changes").
   ACTION: The fill line is not drawn.

Device-independent Graphics Library Supplement for HP-UX Systems

HEWLETT-PACKARD COMPANY
Engineering Productivity Division
19420 Homestead Road
Cupertino, California 95014

MANUAL PART NO. 97084-90001
Printed in U.S.A. Nov 1983
E1183

Printing History

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates. However, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

First Edition	June, 1983
Second Edition	Nov, 1983

PREFACE

Who needs to use this manual?

The *DGL Supplement for HP-UX Systems* complements the *DGL Programmer Reference Manual* and is intended for users of Hewlett-Packard's Device-independent Graphics Library (DGL). The Supplement presents DGL reference information which is specific to the HP-UX operating system.

How is it organized?

The *DGL Supplement for HP-UX Systems* is divided into three parts. Part I contains HP-UX Operating System topics such as programming requirements, user-modifiable graphics tables, and other HP-UX system-specific topics. Part II describes calls which are operating system dependent. Part III describes graphics integer interface calls.

Calls in Parts II and III have the same format as those in the *DGL Programmer Reference Manual*. You may insert these calls in the Reference Manual (in alphabetical order) or use the Supplement as is.

Table of Contents

DGL SUBROUTINES AND PARAMETERS

ZAEND (no parameters)
ZAINT (IODLEN,IODES,IERR)
ZALPH (NCHARS,STRING)
ZASPK (XSIZE,YSIZE)
ZBEGN (no parameters)
ZBEND (no parameters)
ZBINT (IODLEN,IODES,IERR)
ZBMOD (OPCODE)
ZBUTN (ECHO,BUTTON)
ZCOLM (MODEL)
ZCOLR (COLOR)
ZCSIZ (WIDTH, HEIGHT)
ZDCOL (COLOR,COLP1,COLP2,COLP3)
ZDEND (no parameters)
ZDINT (IODLEN,IODES,CONTRL,IERR)
ZDLIM (XMIN,XMAX,YMIN,YMAX,IERR)
ZDPMM (WX,WY,MMX,MMY)
ZDPST (PINDEX,DENSTY,ORIENT,EDGE)
ZDRAW (WX,WY)
ZEND (no parameters)
ZHIGH (HLIGHT)
ZIACS (DWIDE,DHIGH,AWIDE,AHIGH)
ZICOL (COLOR,COLP1,COLP2,COLP3)
ZIDRW (IX,IY)
ZIESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)
ZIMOV (IX,IY)
ZIPGD (NPTS,IXVEC,IYVEC,OPCODE)
ZIPGI (NPTS,IXVEC,IYVEC,OPCODE)
ZIPLY (NPTS,IXVEC,IYVEC)
ZIPST (PINDEX,DENSTY,ORIENT,EDGE)
ZIWS (OPCODE,SSIZE,ISIZE,RSIZE,SLIST,ILIST,
RLIST,IERR)

ZKEND (no parameters)
ZKINT (IODLEN,IODES,IERR)
ZKYBD (ECHO,MAX,ACTUAL,STRING)
ZLEND (no parameters)
ZLINT (IODLEN,IODES,IERR)
ZLLIM (XMIN,XMAX,YMIN,YMAX,IERR)
ZLOCP (WX,WY)
ZLPMM (WX,WY,MMX,MMY)
ZLSTL (LSTYLE)
ZLWID (LWIDTH)
ZMARK (MARKNO)
ZMCUR (no parameters)
ZMOVE (WX,WY)
ZNEWF (no parameters)
ZOESC (OPCODE,ISIZE,RSIZE,ILIST,RLIST,IERR)
ZPGDD (NPOINT,XVEC,YVEC,OPCODE)
ZPGDI (NPOINT,XVEC,YVEC,OPCODE)
ZPICL (COLOR)
ZPILS (LSTYLE)
ZPOLY (NPTS,XVEC,YVEC)
ZPSTL (PINDEX)
ZSLOC (ECHO,WX,WY)
ZSVAL (ECHO,SUBVAL,VALUE)
ZTEXT (NCHARS,STRING)
ZVEND (no parameters)
ZVIEW (VXMIN,VXMAX,VYMIN,VYMAX)
ZVINT (IODLEN,IODES,IERR)
ZWIND (WXMIN,WXMAX,WYMIN,WYMAX)
ZWLOC (ECHO,LBUTN,WX,WY)
ZWVAL (ECHO,SUBVAL,VBUTN,VALUE)

Part 1
System-Dependent Topics

Operating Environment

DGL is supported on the HP-UX operating system.

I/O Unit Descriptors

The I/O unit descriptor specified in the initializatin calls is a device file. To create this file, use the `mknod` command described in the *HP-UX System Administrator* manual and the *HP-UX Reference*.

Programming Languages

Fortran

Usless otherwise specified, the FORTRAN compiler defaults all integers to 342 bits, but DGL requires 16 bit integers. To assign 16 bit integers, use one of the following methods:

- Use the compiler option, `$OPTION SHORT INTEGERS`.
- Use the `-I2` option in the run string for the compiler.

Characters must be passed to DGL in integer arrays. Do this with data statements, as shown in the program example on the following page.

A file, /usr/lib/graphics/fortran/fdgl1.h, is supplied with DGL containing aliases to the DGL routine names which are more English-like. The contents of this file may be copied to a local directory and modified as needed. To use these names, include this file in a DGL program as follows:

```
      $INCLUDE '/usr/lib/graphics/fortran/fdgl1.h'
```

The following FORTRAN example is complete. It may be typed into the computer and executed, or it may be modified to include many of the program fragments which are listed in the *DGL Programmer Reference Manual*.

```
$OPTION SHORT INTEGERS
      PROGRAM USER
C
C..DGL user program to draw a line.
C  The graphics output is hard coded to /dev/crtgraphics
C
C..Set up variables
C
      INTEGER*2 IODES(8), IODLEN, IERR, CNTRL
      DATA IODES/2H/d,2Hev,2H/c,2Hrt,2Hgr,2Hap,2Hhi,2Hcs/
      DATA IODLEN/16/
C
C..Initialize the system
C
      CALL ZBEGN
C
C..Initialize the graphics display device
C  at /dev/crtgraphics with no control bits set.
C

      CNTRL = 0
C
      CALL ZDINT(IODLEN, IODES, CNTRL, IERR)
C
C
C..Everything between the comment lines may be changed as needed.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C..Use default attributes and viewing transformation to draw line
C  diagonally across the display.
C
      CALL ZMOVE (-1.0,-1.0)
      CALL ZDRAW (1.0,1.0)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C..Disable the graphics display device, end the DGL system,
C  and terminate this program.
C
      CALL ZDEND
      CALL ZEND
      END
```

Pascal

In general, the difficulties which arise when accessing DGL routines are those which arise when accessing *any* non-Pascal routine from a Pascal program. Therefore, a review of that topic in the Pascal Manual may be useful.

Parameters (param1, param2, ...) passed to DGL routines must be declared VAR, or "call-by-reference". Note that any external routine may modify the actual parameters passed to it. However, DGL does not modify input parameters. All DGL routines should be declared as external procedures as follows:

```
PROCEDURE DGL_routine
   (VAR param1 : type1;
    VAR param2 : type2;
           :
           :
    VAR paramN : typeN);
EXTERNAL;
```

The following files are supplied with DGL:

```
/usr/lib/graphics/pascal/pdgl1.h
/usr/lib/graphics/pascal/pdgl2.h
```

These files provide standard data types and external declarations for inclusion in all application programs. The file pdgl2.h contains declarations for the standard DGL rountine names as well as aliases to a set of more English-like names. The contents of these files may be copied to a local directory and modified as needed.

The application program must use the data types which DGL expects. Unfortunately, Pascal's and FORTRAN's data types are not always equivalent. The Pascal user should therefore declare and use appropriate data types for both the formal and actual parameters of DGL routines. One example of this incompatibility is the data type INTEGER. Pascal represents integers in 32 bits, while DGL's FORTRAN code always assumes them to be 16 bits.

The following declarations cause the variable "graphics_id" to be stored in a 16 bit integer:

```
TYPE
     INT  = -32768..32767;

VAR
     graphics_id : INT;
```

The following table summarizes correspondences between DGL parameters and matching Pascal data types. A file containing standard data-type declarations for Pascal is supplied with the DGL product.

DGL expects	Pascal data type
Integer	-32768..32767
Character string of n characters	PACKED ARRAY[1..n] OF CHAR or ARRAY [1..n] of CHAR
Real	REAL
Array of n integers	ARRAY [1..n] OF -32768..32767

A PASCAL version of program USER is shown below.

```
$PASCAL
PROGRAM user;   { DGL user program to draw a line }

{ The following include files are part of the DGL product }
$INCLUDE '/usr/lib/graphics/pascal/pdgl1.h'$

var
  ierr : INT;
  control : CONTROL_WORD;
  iodes : STRING132;
  iodlen : INT;
  xmin, xmax, ymin, ymax : REAL;
$INCLUDE '/usr/lib/graphics/pascal/pdgl2.h'$

BEGIN  { Main of user }

{ Initialize variables }
  iodlen := 16;
  iodes := '/dev/crtgraphics';
  xmin := -1.0;
  xmax :=  1.0;
  ymin := -1.0;
  ymax :=  1.0;
  control := control_spooling_off;

{ Initialize the DGL system.                                    }
  zbegn;

{ Initialize a graphics display device at /dev/crtgraphics with }
{ no control bits set}                                          }
  zdint ( iodlen, iodes, control, ierr );

{ Everything between the dashes may be changed as needed        }
{ ------------------------------------------------------------ }

{ Using default attributes and viewing transformation, draw line}
{ diagonally across the display                                 }
  zmove ( xmin, ymin );
  zdraw ( xmax, ymax );

{ ------------------------------------------------------------ }

{ Disable the graphics display device and end the system        }
  zdend;
  zend;
END.  { Main of User}
```

C

In general, the difficulties which arise when accessing DGL routines are those which arise when accessing *any* FORTRAN routine from C. See the FORTRAN manual for a discussion of the FORTRAN/C interface.

The user must pass actual parameters (param1, param2, ...) to DGL routines by reference only. Therefore, all actual parameters passed to DGL must be pointers or variables prefixed with the address operator (&). All formal parameters returned from DGL must be pointer variables. All integers parameters should be declared as short integers and all reals as float.

DGL expects character parameters to be passed in integer arrays. C programmers can pass these in one of two ways:

1. use an array of short integers and a character pointer to the array; assign the characters individually using the character array (see C program below).

2. pass a double-quoted character string. For example, the zdint call in the C program could have been coded:

```
zdint (&iodlen,"/dev/crtgraphics",&control,&ierr);
```

In either case, the number of characters in the string passed must be at least the length specified. For character output, DGL requires a pointer to an array of short integers.

The following table summarizes correspondences between DGL parameters and matching C data types. User-supplied names are bracketed by <>. Characters are packed two per element in a short integer array and require (n+1)/2 elements to store n characters. For example, 15 characters could be stored in an array declared short string[8].

DGL expects	C data type
Integer	short
Character string of n characters (see note above)	short <arrayname> [(n+1)/2]
Real	float
Array of n integers	short <arrayname> [n]

A file, /usr/lib/graphics/c/cdgl1.h, is supplied with the DGL product. It contains macro defintions of more English-like names. The contents of this file may be copied to a local directory and modified as needed. To use these names, include this file in a DGL program as follows:

```
#include "/usr/lib/graphics/c/cdgl1.h"
```

A C version of program USER is shown below.

```
main ()
{     /* begin main of user*/

/*****************************************************************/
/*    DGL user program to draw a line.                           */
/*    The graphics output is hard-coded to /dev/crtgraphics.     */
/*****************************************************************/

      short   iodlen,iodes[8],control,ierr;
      float  xmin,xmax,ymin,ymax;
      char   *ioces = (char *)iodes;

/*****************************************************************/
/*    Initialize the DGL system                                  */
/*****************************************************************/

      zbegn();

/*****************************************************************/
/*    Initialize the graphics display device at                  */
/*    /dev/crtgraphics with no control bits set.                 */
/*****************************************************************/

      iodlen = 16;
      ioces[0] = '/';
      ioces[1] = 'd';
      ioces[2] = 'e';
      ioces[3] = 'v';
      ioces[4] = '/';
      ioces[5] = 'c';
      ioces[6] = 'r';
      ioces[7] = 't';
      ioces[8] = 'g';
      ioces[9] = 'r';
      ioces[10] = 'a';
      ioces[11] = 'p';
      ioces[12] = 'h';
      ioces[13] = 'i';
      ioces[14] = 'c';
      ioces[15] = 's';
      control = 0;
      zdint(&iodlen,iodes,&control,&irerr);

/*****************************************************************/
/*    Everything between the dashes may be changed as needed     */
/*---------------------------------------------------------------*/

/*****************************************************************/
/*    Use default attributes and viewing transformation          */
/*    to draw a line diagonally across the display.              */
/*****************************************************************/
```

```
      xmin = -1.0;
      xmax =  1.0;
      ymin = -1.0;
      ymax =  1.0;
      zmove (&xmin,&ymin);
      zdraw (&xmax,&ymax);

/*-------------------------------------------------------------*/
/*************************************************************/
/*    Disable graphics display device and end the DGL system  */
/*************************************************************/
      zdend();
      zend();
}     /*  end main of user */
```

LIBRARY NAMES

This supplement refers to libraries in the same way they are referenced using the -l option on the linker. By adding the appropriate prefix and suffix to these names, the files can be found in the /lib directory. For example, the file name for the D0031 library is /lib/libD0031.a and for DIDD, /lib/libDIDD.a.

LINKING DGL PROGRAMS

When linking DGL application programs, the user must be sure to search a device handler for each logical device accessed by the program. If the program uses the graphics display, it must be the last device handler searched. Search the DGL library (DIDD) after searching all the device handlers.

Because DGL is written in FORTRAN and C, the user must search the FORTRAN, C and math libraries for every link procedure. Shown below is the compile and link procedure for an application program which accesses the following:

the graphics display device on the HP 7470A Plotter,
the locator on the HP 7580B Drafting Plotter and
the button device on the HP 9111A Data Tablet.

Note that in these examples some libraries are automatically searched, and therefore, do not have to be specified in the run string.

For FORTRAN:

```
fc program.f -lB004 -lL006 -lD0031 -lDIDD
```

For Pascal:

```
pc program.p -lB004 -lL006 -lD0031 -lDIDD -lF77 -lI77
```

For C:

```
cc program.c -lB004 -lL006 -lD0031 -lDIDD -lF77 -lI77 -lm
```

See the HP-UX documentation for details of additional compiler and linker options.

If the application program references a logical device but does not explicitly load a device handler for the device, a dummy device handler is automatically loaded. For example, if the application program calls ZBINT but the load procedure does not search for a button device hander (Bxxxx), a dummy device handler is implicitly loaded from DIDD.

Special Information for Starbase Handlers

There are five handlers which communicate with another software product rather than communicating directly with a device. These handlers are:

- B0056
- L0056
- P0056
- V0056
- D0056

These function in the same way as other handlers except for two areas: program linking (using the pc or ld system commands) and device initialization (using the zdint or zbint DGL commands).

When calling the device initialization routines for the DGL to Starbase handler, there is a small change in the second parameter. Currently, ZBINT, ZDINT, ZLINT, ZVINT, and the AGP calls JDINT and JEDEV expect the second parameter to be the name of a device file where the device is located, such as /dev/tty. For the DGL to Starbase handlers **only** (D0056, B0056, L0056, P0056, V0056) the second parameter takes the form

```
filename device_driver.
```

The following table shows several path names and their associated lengths.

```
                              PATH NAME              LENGTH

Most DGL handlers:        /dev/tty                      8
D0056:                    /dev/tty hp98700             16
B0056:                    /dev/hil2 hp-hil             16
```

Figure I.1. The Second Parameter

The first parameter to the ZBINT or ZDINT call is the length of the entire second parameter. The length of the second parameter is the number of characters in the device file name, **plus** one for the space between the device file name and the Starbase device identifier, **plus** the number of characters in the Starbase device driver name. There must be exactly one space in the string or an error may be reported.

There is also a way to default the Starbase device driver name so that existing code which only specifies the path name will continue to run. If you wish to use the default name, the ZxINT call should specify only the device file name, the same way that all other device handlers require the name to be specified. The default Starbase driver name is specified by linking additional libraries. The library names have the form

```
d.<starbase id>
```

where <starbase id> is a shortened form of the Starbase `GOPEN` driver type parameter, usually shortened by removing the leading `hp`.

Some library names are:

```
d.98700      corresponds to hp98700
d.98710      corresponds to hp98710
d.98760      corresponds to hp98760
d.262x       corresponds to hp262x
d.2623       corresponds to hp2623
d.2627       corresponds to hp2627
d.hpgl       corresponds to hpgl
d.hil        corresponds to hp-hil
d.kbd        corresponds to keyboard
```

A complete list of libraries may be found by looking in the HP-UX directory `/lib`.

One of these default name libraries should be specified after each Starbase DGL handler in the link command. The default name may always be replaced by an explicit name specified in the ZxINT call.

The Starbase Device Identifiers and the device driver names are defined in the Starbase documentation.

To link a DGL program which uses a Starbase DGL handler, you must also include the proper Starbase graphics drivers and the Starbase library files. If you wish to specify a default Starbase driver name, the default name libraries must also be linked in.

As an example, assume a program which uses a Starbase display and button, and standard DGL locator and alpha devices. The button device will default to a hp-hil device and the display will default to a hp98700 device. The compile command could be:

```
fc -x -o demo demo.f -lA0000 -lL0004 \   <<< Alpha, Locator
      -lB0056 -ld.hil   -lddhil \        <<< Button, default, starbase
      -lD0056 -ld.98700 -ldd98700 \      <<< Display, default, starbase
      -lDIDD \                           <<< Device Independent
      -lsb1 -lsb2                        <<< Starbase Device Independent
```

Note the standard DGL rules are followed: link all devices except the display, then link the display, then link DIDD. With each Starbase DGL handler an optional default name may be specified. The same default name may be specified for more than one Starbase DGL handler. Each handler also requires that the Starbase driver be loaded. Each Starbase driver must be loaded once only. (If both the button and the locator are hp-hil devices, specify -lddhil only once; you may specify d.hil several times however.) Finally, link the device independent Starbase libraries.

Color Table Size Change

Every graphic display device has a default color table, which is loaded whenever the corresponding device handler is loaded. The default tables of specific devices are given in the *Device Handlers Manual.*

For some devices, the number of colors in the default table can be increased at link time. Consult the *Device Handlers Manual* for a specific device to determine if its color table can be resized. This number must never be made greater than 32767 (the system limit), or fewer than the default size for the device.

If the number of entries in the color table is increased, each new entry has the same default parameters as entry 1. If nothing more is done, the effect of referencing any entry greater than the default size will be the same with the modified table as with the default table. If the new entries are to permit the display of new colors, their contents must be redefined within the program using ZDCOL.

To change the size of the color table:

1. Make a copy of the source file: `/usr/lib/graphics/fortran/z1ctb.f`.
2. Modify the copy using the instructions documented in the source code.
3. Compile the source module with `fc -c z1ctb.f`.
4. Link the new color table in with the DGL program. Examples are shown for the various types of main programs.

- FORTRAN program:
  fc program.f z1ctb.o -lB0004 -lL0006 -lD0031 -lDIDD
- Pascal program:
  pc program.p z1ctb.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77

- C program:
  cc program.c z1ctb.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77 -lm

Polygon Intercept Buffer Size Change

Most device-independent polygon-filling routines compute all the locations where a given fill line intercepts the edge of the polygon set. If a polygon set edge has n segments, up to n ;interceptes can be generated for the worst case of a concave or self-intersecting polygon set. DGL stores the intercept locations in an intercept buffer. If the number of intercepts overflows the buffer, the fill line is not drawn. In this case, singel fill lines or portions of the polygon set will not be filled. The intercept buffer has a 100 point default capacity, whicfhcan be changed only by modifying the source code.

To change the size of the plygon intecept buffer:

1. Make a copy of the source file `/usr/lib/graphics/fortran/t1lint.f`.
2. Modify the copy using the instructrion documented in the source code.
3. Compile the source module using `fc -c t1lint.f`.

4. Link the new polygon intercept buffer in with the DGL program. Examples are shown for the various types of main programs.

   FORTRAN program:

   ```
   fc program.f t1int.o -lB0004 -lL0006 -lD0031 -lDIDD
   ```

   Pascal program:

   ```
   pc program.p t1int.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77
   ```

   C program:

   ```
   cc program.c t1int.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77 -lm
   ```

PERFORMANCE

Although several things can influence the performance of DGL application programs, the one the programmer can most easily control is the timing mode used to send graphics commands to a device. The default timing mode, system buffering, is the most efficient method because it buffers all graphics commands before they are transmitted; this minimizes the number of data transfers. Therefore, system buffering reduces the amount of time spent in the operating system as well as time spent performing input and output. When operating in immediate visibility mode, a data transfer occurs for each graphics command.

In system buffering mode, a program can update the image whenever necessary by performing an input operation or by calling ZMCUR. Both operations flush the DGL buffer and therefore update the image.

OUTSPOOLING GRAPHICS

DGL allows the user to outspool graphics commands but does not do the spooling itself. Some devices may not support spooling.

Outspooling is enabled by setting bit 0 of the control word in the ZDINT call. If bit 0 is not set, DGL checks to be sure the type of device the I/O unit descriptor points to is correct. If bit 0 is set (i.e., spooling is enabled), DGL does not check the type.

If the alphanumeric device and the graphics display device are identical, alphanumeric data and graphics data can be spooled simultaneously; but it is necessary to enable the graphics display before enabling the alphanumeric display.

DGL output can be outspooled to a disc file which can later be sent to the graphics/alphanumeric device. This is useful when identical copies of a picture are needed.

Once created, the file can be sent only to a physical device which is the same type supported by the device handler linked with the application program. For example, if a file was created using an HP 9872C device handler, the file can only be sent to an HP 9872C plotter.

KEYBOARD INPUT

Special Keys

When using ZKYBD the operator enters the desired text and terminates the operation by entering a carriage return. Some keys have a special meaning when entered by the operator and are not returned to the application program as shown below.

Key	Function
BACKSPACE	Deletes the last character typed. The alphanumeric cursor moves back one character and overwrites the character with a space.
LINE FEED	A line feed terminates the keyboard operation, the same as a carriage return.
CONTROL-D	Deletes the last character typed and terminates keyboard operation.

Type-Ahead

When using the tty (software) terminal interface, HP-UX allows the user to type characters at any time, even during output. During the ZDINT call, if the spooling bit is not set, any characters that have been typed ahead are cleared to allow DGL to identify the device. At any other time the user may type ahead responses that DGL expects to get from the device.

Anything that is typed ahead will be treated by DGL as if it had come directly from the device. Therefore, the content and order must be *exactly* as expected from the device or incorrect results may occur. For example, before requesting locator input from the HP 2623, if the user types in a point as '.23,.45' an invalid locator point is returned since that is not the format for a point that DGL expects from the device.

OUTSIDE-WORLD SIGNALS

Through DGL, the user may end a graphics application program with any of the following signals:

Hang up - signal #1
Interrupt - signal #2
Quit - signal #3
Software termination - signal #15

DGL uses these signals only if they are set to the system default action (which is termination). ZBEGN checks to see whether or not the user has changed the action. If the action has not been changed, DGL responds to a signal by attempting to restore devices to their normal state and terminating graphics (using ZEND) and the program.

DGL can only detect changes to a signal's action if the user program makes the change before calling ZBEGN. If the program has changed the action, DGL takes no action. If desired, the program can terminate just graphics (using ZEND) after it intercepts the signal and then continue to operate. For more information on the use of signals, refer to the appropriate HP-UX manual.

CAUTION: For devices that store commands in internal buffers (such as the HP 7580B Drafting Plotter), graphics output may continue for a while after the signal has been sent.

Part II
System-Dependent Calls

INTRODUCTION

The calls described in this supplement contain system-dependent information. These descriptions, together with the (system-independent) call descriptions in the *DGL Reference Manual* provide complete documentation for DGL subroutines.

The format of the following pages is the same as the format in the Reference Manual. You may insert these call descriptions in the *DGL Reference Manual* (in alphabetical order) or use this system-specific manual as it is.

ZAINT

PURPOSE: To enable the alphanumeric device.

CALLING SEQUENCE: CALL ZAINT (IODLEN, IODES, IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

IERR [INTEGER; Output]
A return code indicating whether the alphanumeric device was successfully enabled.

IERR = 0 Alphanumeric device successfully enabled.

= 1 The alphanumeric device is not supported on this work station. This occurs when the user loads a dummy device handler instead of the alphanumeric device handler. Call ignored.

= 2 The requested device is not available. This can occur for the following reasons:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O unit descriptor is not the same as the one specified in the device handler.
- o The IODLEN specified is less than one or greater than 132.

Call ignored.

ZAINT enables the alphanumeric device for output (ZAEND disables it). Call ZAINT before attempting to output alphanumeric data using ZALPH. ZAINT associates the logical device with a physical device; it implicitly makes the picture current and then initializes the device. The device name is then set to the name of the physical device and the device status is set to 1 (enabled).

If an alphanumeric device is currently enabled a call to ZAINT terminates the enabled device (using ZAEND) and then continues.

Alphanumeric data can be outspooled to the same I/O unit descriptor as that of a currently enabled graphics display device. To outspool to the alphanumeric device it must be enabled after the graphics display device has been enabled.

The alphanumeric device should be disabled before ending the application program.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) Alphanumeric device not supported on this work station.
   ACTION: IERR set to 1 and rest of call ignored.

3) Alphanumeric device already enabled.
   ACTION: Currently enabled device is terminated through an implicit call to ZAEND, and call continues.

4) Requested device not at the specified I/O unit descriptor.
   ACTION: IERR set to 2 and rest of call ignored.

ZBINT

PURPOSE: To enable the logical button device.

CALLING SEQUENCE: CALL ZBINT (IODLEN, IODES, IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

IERR [INTEGER; Output]
A return code indicating whether the button device was successfully enabled.

IERR = 0 Button device successfully enabled.

= 1 The button device is not supported on this work station. This occurs when the user loads a dummy device handler instead of the button device handler. Call ignored.

= 2 The requested device is not available. This can occur for the following reasons:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O unit descriptor is not the same as the one specified in the device handler.
- o The IODLEN specified is less than one or greater than 132.

Call ignored.

= 3 The specified physical device is an outspooled graphics display device. Call ignored.

ZBINT enables a logical button device for input (ZBEND disables it). ZBINT associates the logical device with a physical device; it implicitly makes the picture current and then initializes the device. The device name is then set to the name of the physical device and its status is set to 1 (enabled).

ZBUTN is used to perform the button input once the button device is enabled. If a button device is currently enabled, ZBINT terminates the enabled device (using ZBEND) and then continues.

Note that a logical device cannot be enabled for input if output is being spooled from the physical device (see ZDINT).

Disable the button device before ending the application program.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) Button device is not supported on this work station.
   ACTION: IERR is set to 1 and the rest of the call is ignored.

3) Button device already enabled.
   ACTION: Currently enabled button device is terminated and the call continues.

4) Attempt to enable input on the same physical device as an outspooled graphics display device.
   ACTION: IERR set to 3 and rest of call ignored.

5) The requested device is not at the I/O unit descriptor specified.
   ACTION: IERR set to 2 and rest of call ignored.

ZDINT

PURPOSE: To enable a graphics display device.

CALLING SEQUENCE: CALL ZDINT (IODLEN, IODES, CONTRL, IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

CONTRL [INTEGER; Input]
Controls the characteristics of the graphics display device. Bits should be set according to the following map (all unused bits set to 0).

D	D	D	D	D	D	D	D	X	X	0	0	0	0	0	X
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BIT 0 = 0 - Graphics data cannot be outspooled to the I/O unit descriptor.

= 1 - Graphics data can be outspooled to the I/O unit descriptor. DGL does not do the spooling; it only ensures that nothing is read from the device so the system can outspool the data.

BITS 1-5 - Currently unused. Should be set to 0.

BIT 6 = 0 - When ZNEWF is called, the graphics display is made current and then the display is cleared.

= 1 - When ZNEWF is called, the graphics display is made current but the instruction to clear the graphics display is stored in the DGL buffer; the instruction is then executed the next time the buffer is sent.

When the buffer is sent, the display will be cleared and whatever calls to DGL were put into the buffer after the ZNEWF call will take effect on the graphics display. Note that if immediate visibility is used, the action is the same as if bit 6 were 0 since the buffer is sent after every DGL call.

BIT 7 = 0 - ZDINT clears the graphics display. This action is not affected by the value of bit 6 (i.e. even if buffered new-frame-action is requested, ZDINT clears the screen immediately).

= 1 - The graphics display is not cleared during initialization. If the physical background color in effect is different from the initial value of color 0 (DGL's background color), the physical background color will be updated when the buffer is flushed or when ZNEWF is called, depending on the device. Also, if the device has retroactive color definition and the color table has been redefined, the colors of the image may change because DGL initializes the color table to default values.

BITS 8-15 - D = device-dependent bit. Set to 0 if not used.

IERR [INTEGER; Output]
A return code indicating whether the graphics display device was successfully initialized.

IERR = 0 The graphics device was successfully initialized.

= 1 The graphics device is not supported by the work station. This occurs when the user loads a dummy device handler instead of the graphics device handler. Call ignored.

= 2 The requested graphics display device is not available.

Possible reasons are:

- The incorrect I/O unit descriptor was specified.
- The specified device is down.
- The physical device at the I/O descriptor is not the same as the one specified in the device handler.
- The IODLEN specified is less than one or greater than 132. Call ignored.

= 3 An input device has previously been assigned to this physical device and the spooling option was specified, or the device does not support spooling. Call ignored.

Before the device is initialized the device status is 0, the I/O unit descriptor length is 0, and the device name is the default name. The default name is the last four characters of the display library loaded, followed by two blanks. For example, for the HP 7470A Plotter, the required library is D0031, and the default name is "0031 ". When a device is enabled the device status is set to 1 (enabled) and I/O unit descriptor of the device is connected. If the spooling option is used, the device name remains set to the default name; otherwise the device name is set to the device being used. See the *Device Handlers Manual*.

If the spooling option is not used (that is, bit 0 of CONTRL is set to 0), ZDINT checks whether the graphics display device is up. Then, if possible, DGL inquires the device name from the device at the I/O unit descriptor. Unpredictable results occur if there is no device at the I/O unit descriptor, if the device is not the intended graphics device, if the device is turned off or in a local mode, or if the connection between the device and the computer is bad.

If an input device is currently enabled on the same physical device specified when ZDINT is called and spooling is requested, ZDINT generates an error (IERR = 3).

Note that some devices cannot be outspooled. See the *Device Handlers Manual*.

Alphanumeric data can be outspooled to the same I/O unit descriptor as that of a currently enabled graphics display device. To do this, enable the alphanumeric device *after* enabling the graphics display device.

Initialization includes the following operations:

1. The graphics display surface may be cleared (see BIT 7 of CONTRL).
2. The starting position is set to a device-dependent location.
3. The logical display limits are set to the default limits for the device.
4. The aspect ratio of the virtual coordinate system is applied to the logical display limits to define the limits of the virtual coordinate system.
5. All primitive attributes are set to their default values.
6. The locator echo position is set to its default value.
7. If the display and the locator are the same physical device, the logical locator limits are set to the limits of the view surface.

Only one type of graphics display device can be accessed by a DGL application program. A single program, however, can send output to several devices of the same type. For example, after sending graphics output to an HP 7470, a program could disable it and then reassociate the graphics output with another HP 7470 by changing the value of the I/O unit descriptor.

If the graphics display device is currently enabled, a call to ZDINT terminates the enabled device (using ZDEND) and then continues.

After this call, the starting position should be established with a call to a primitive which sets but does not begin at the starting position; such calls place the physical pen or beam at a known location on the graphics display device. Examples are: ZMOVE, ZPOLY, ZPGDI, ZPGDD, ZIMOV, ZIPLY, ZIPGI, and ZIPGD.

No other program should send data to the I/O unit descriptor while it is initialized for use by DGL. There is no DGL error check for this occurrence and the results are unpredictable.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) The graphics display device is not supported on the work station.
   ACTION: IERR is set to 1 and the rest of the call is ignored.

3) A device is already enabled as the graphics display device.
   ACTION: The device is terminated and ZDINT continues.

4) The outspooling option is specified and an input device is already enabled on the specified physical device.
   ACTION: IERR is set to 3 and the rest of the call is ignored. Disable the logical input device.

5) Although the device does not support outspooling, outspooling was requested by the CONTRL parameter in ZDINT.
   ACTION: IERR set to 3 and rest of call ignored.

6) The graphics display device is not at the I/O unit descriptor specified.
   ACTION: IERR is set to 2 and the rest of the call is ignored.

ZIWS

PURPOSE: To inquire characteristics of the DGL system.

CALLING SEQUENCE: CALL ZIWS (OPCODE, SSIZE, ISIZE, RSIZE, SLIST, ILIST, RLIST, IERR)

OPCODE [INTEGER; Input]
The code specifying which characteristic of the DGL system system is being queried. The supported values of OPCODE are listed below.

SSIZE [INTEGER; Input]
Maximum number of characters that are to be returned in the array SLIST by the function specified by OPCODE.

ISIZE [INTEGER; Input]
The number of integer parameters that are returned in the array ILIST by the function specified by OPCODE.

RSIZE [INTEGER; Input]
The number of real parameters that are returned in the array RLIST by the function specified by OPCODE.

SLIST [INTEGER; Array; Output]
An array of characters that represent characteristics of the work station specified by the value of OPCODE. SLIST must be dimensioned large enough to hold SSIZE characters.

ILIST [INTEGER; Array; Output]
An array of integer values that represent the characteristics of the work station requested by OPCODE. ILIST must be dimensioned large enough to hold ISIZE characters.

RLIST [REAL; Array; Output]
An array of real values that represent the characteristics of the work station specified by the value of OPCODE. RLIST must be dimensioned large enough to hold RSIZE characters.

IERR [INTEGER; Output]
Return code indicating whether the inquiry was performed.

IERR = 0 Inquiry successfully performed.

= 1 OPCODE invalid. Call ignored.

= 2 ISIZE not equal to number of integer parameters returned by OPCODE. Call ignored.

= 3 RSIZE not equal to number of real parameters returned by OPCODE. Call ignored.

= 4 SSIZE not large enough to receive the character string requested by OPCODE or SSIZE should be 0 for the OPCODE. Call ignored.

The subroutine ZIWS returns current information about the DGL system to the application program. The type of information desired is specified by a unique value for OPCODE. OPCODE digits have the following meanings:

Ten thousands digit = 1 if a character string is returned in SLIST.
= 0 if no string returned.

Thousands digit = Number of integer values returned in ILIST.

Hundreds digit = Number of real values returned in RLIST.

Supported values of OPCODE and their meanings:

OPCODE MEANING

250 Actual cell size for a graphics text character, in world coordinates, as determined by the "smaller best fit" rule.

RLIST(1) = Character cell width.
(2) = Character cell height.

251 Marker size in world coordinates (valid only if a graphics display device is enabled).

RLIST(1) = Marker width.
(2) = Marker height.

252 Resolution of graphics display on a CRT in points/mm.

RLIST(1) = Resolution in X-direction.
(2) = Resolution in Y-direction.

253 Maximum dimensions in mm of display surface of graphics device (may be wrong if device not yet enabled).

RLIST(1) = Maximum size in X-direction.
(2) = Maximum size in Y-direction.

254 Aspect ratios.

RLIST(1) = Aspect ratio of virtual coordinate system.
(2) = Aspect ratio of logical display limits.

255 Resolution of locator device in points/mm.

RLIST(1) = Resolution in X-direction.
(2) = Resolution in Y-direction

256 Maximum dimensions of locator device in mm.

RLIST(1) = Maximum size in X-direction.
(2) = Maximum size in Y-direction.

257 Locator echo position in world coordinates.

RLIST(1) = X-coordinate of position.
(2) = Y-coordinate of position.

258 Virtual-coordinate limits.

RLIST(1) = Maximum X-coordinate.
(2) = Maximum Y-coordinate.

259 Starting position in world coordinates. (Undefined after a text call, an escape function call, changes to the viewing transformations, or the enabling of a graphics device.)

RLIST(1) = X-coordinate of position.
(2) = Y-coordinate of position.

450 Window limits in world coordinates.

RLIST(1) = Minimum X-coodinate of window.
(2) = Maximum X-coodinate of window.
(3) = Minimum Y-coodinate of window.
(4) = Maximum Y-coodinate of window.

451 Viewport limits in virtual coordinates.

RLIST(1) = Minimum X-coordinate of viewport.
(2) = Maximum X-coordinate of viewport.
(3) = Minimum Y-coordinate of viewport.
(4) = Maximum Y-coordinate of viewport.

1050 Does the graphics display device support hardware clipping?

ILIST(1) = 0 - No.
= 1 - Yes, to the view surface boundaries.
= 2 - Yes, but only to the physical limits of the display surface.

1051 Justification of the view surface within the logical display limits.

ILIST(1) = 0 - View surface is centered within the logic display limits.
= 1 - View surface origin is at the lower left corner of the logical display limits.

1052 Can the graphics display device draw using the background color?

ILIST(1) = 0 - No.
= 1 - Yes.

1053 Total number of non-dithered colors supported on the graphics display. The number returned does not include the background color (compare opcodes 1053, 1054, 1075).

ILIST(1) = Number of distinct colors supported (color palette).

1054 Number of non-dithered distinct colors which can appear on the graphics display at one time. The number returned does not include the background color (compare opcodes 1053, 1054, 1075.)

ILIST(1) = Number of distinct colors which can appear on the display at one time (color gamut).

1055 Number of types of highlighting supported on the graphics device.

ILIST(1) = Number of highlights.

1056 Number of linestyles supported on the graphics device.

ILIST(1) = Number of hardware linestyles.

1057 Number of linewidths supported on the graphics device.

ILIST(1) = Number of linewidths.

1058 Number of hardware character sizes supported on the graphics device.

ILIST(1) = Number of character sizes. If ILIST(1) = -1, the graphics device supports continuously varying character sizes.

1059 Number of markers supported on the graphics device.

ILIST(1) = Number of markers.

1060 Color index for line primitives.

ILIST(1) = Color index value.

1061 Highlighting attribute.

ILIST(1) = Highlight value.

1062 Linestyle index for all primitives excluding polygon interior fill.

ILIST(1) = Linestyle index value.

1063 Linewidth attribute.

ILIST(1) = Linewidth value.

1064 Timing mode for displaying graphics primitives.

ILIST(1) = 0 - Display is immediate.
= 1 - Display is delayed because DGL commands are buffered.

1065 Number of entries in the polygon style table. If this number is 0, the table cannot be changed because dummy polygon routines were loaded.

ILIST(1) = 0 - Polygon routines not available.
> 0 - Number of styles in the table.

1066 Color index for polygon interiors.

ILIST(1) = Color index value.

1067 Polygon style index.

ILIST(1) = Value of polygon style index.

1068 Number of polygon vertices supported by the graphics device.

```
ILIST(1) = 0                  - No hardware support.
         = n (0 < n < 32767)  - Number of vertices.
         = 32767              - No hardware limit on number of
                                vertices.
```

1069 Does the graphics device support immediate, retroactive change of polygon style for polygons already displayed?

```
ILIST(1) = 0 - No.
         = 1 - Yes.
```

1070 Does the graphics device support hardware generation of polygons using ZPGDD?

```
ILIST(1) = 0 - No.  Use ZPGDI instead.
         = 1 - Yes.
```

1071 Does the graphics device support immediate, retroactive color change for primitives already displayed?

```
ILIST(1) = 0 - No.
         = 1 - Yes.
```

1072 Can the background color of the graphics device be changed?

```
ILIST(1) = 0 - No.
         = 1 - Yes.
```

1073 Can entries in the color table of the graphics device be redefined using ZDCOL?

```
ILIST(1) = 0 - No.
         = 1 - Yes.
```

1074 Color model in use.

```
ILIST(1) = 1 - RGB.
         = 2 - HSL.
```

1075 Number of entries in the color table. The number returned does not include the background color (compare opcodes 1053, 1054, 1075).

```
ILIST(1) = 0 - Color table cannot be changed.
ILIST(1) > 0 - Number of entries in the table.
```

1076 Linestyle index for polygon interior fill.

```
ILIST(1) = Value of polygon interior linestyle index.
```

2050 For internal use only.

11050 Graphics display device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

11051 Keyboard device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

11052 Locator device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

11053 Button device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

11054 Alphanumeric device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

11055 Valuator device association.

SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 character maximum).

12050 Graphics display device information.

SLIST(1) = Character string containing device name, blank filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status
= -1 There is no graphics display device loaded.
= 0 Graphics display device is not enabled.
= 1 Graphics display device is enabled.

12051 Keyboard device information.

SLIST(1) = Character string containing device name, blank filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status
= -1 There is no keyboard device loaded.
= 0 Keyboard device is not enabled.
= 1 Keyboard device is enabled.

13052 Locator device information.

SLIST(1) = Character string containing device name, blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status.
= -1 There is no locator loaded.
= 0 Locator device is not enabled.
= 1 Locator device is enabled.
ILIST(3) = Number of buttons on the locator device (0 if there is no locator device loaded).

13053 Button device information.

SLIST(1) = Character string containing device name, blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status.
= -1 There is no button loaded.
= 0 Button device is not enabled.
= 1 Button device is enabled.
ILIST(3) = Number of buttons on the button device (0 if there is no button device loaded).

14054 Alphanumeric device information.

SLIST(1) = Character string containing device name, blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status.
= -1 There is no alphanumeric device loaded.
= 0 Device is not enabled.
= 1 Device is enabled.
ILIST(3) = Maximum number of lines displayable at once.
ILIST(4) = Maximum number of characters per line. -1 if underminable (e.g. lineprinter).

14055 Valuator device information.

SLIST(1) = Character string containing device name, blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
ILIST(2) = Status.
= -1 There is no valuator loaded.
= 0 Device is not enabled.
= 1 Device is enabled.
ILIST(3) = Number of buttons on the valuator device (0 if there is no valuator device loaded).
ILIST(4) = Number of sub-valuators supported.

ERROR CONDITIONS:

1) The DGL system is not initialized.
ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Invalid OPCODE specified.
ACTION: IERR is set to 1 and the rest of the call is ignored.

3) ISIZE is not equal to the number of integer values to be returned.
ACTION: IERR is set to 2 and the rest of the call is ignored.

4) RSIZE is not equal to the number of real values to be returned.
ACTION: IERR is set to 3 and the rest of the call is ignored.

5) SSIZE is not large enough to receive the character string requested or SSIZE should be 0 for the OPCODE.
ACTION: IERR is set to 4 and the rest of the call is ignored.

ZKINT

PURPOSE: To enable the keyboard device.

CALLING SEQUENCE: CALL ZKINT (IODLEN, IODES , IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

IERR [INTEGER; Output]
A return code indicating whether the keyboard device was successfully enabled.

IERR = 0 The keyboard was successfully enabled.

= 1 The keyboard device is not supported on this work station. This occurs when the user loads a dummy device handler instead of the keyboard device handler. Call ignored.

= 2 The requested device is not at the specified I/O unit descriptor. This can occur for the following reasons:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O unit descriptor is not the same as the one specified in the device handler.
- o The IODLEN specified is less than one or greater than 132.

Call ignored.

= 3 An input device has previously been assigned to this physical device and the spooling option was specified, or the device does not support spooling.

Call ignored.

ZKINT enables the logical keyboard device for input (disable with ZKEND) by associating the logical with the physical keyboard device and initializing

the device. (ZKINT implicitly makes the picture current before attempting to initialize the device.) The device name is set to the name of the physical device and the device status is set to 1 (enabled).

If the keyboard is currently enabled, ZKINT terminates the enabled device (using ZKEND) and continues.

Note that if a physical device is being used as an outspooled graphics display device, a logical input device cannot be enabled on it.

Call ZKYBD to return a string from the enabled keyboard.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) Keyboard device not supported on this work station.
   ACTION: IERR is set to 1 and the rest of the call is ignored.

3) Keyboard device already enabled.
   ACTION: Enabled keyboard device is terminated and the call continues.

4) Attempt to enable input on the same physical device as an outspooled graphics display device.
   ACTION: IERR set to 3 and rest of call ignored.

5) The specified I/O unit descriptor does not support a logical keyboard device.
   ACTION: IERR set to 2 and rest of call ignored.

ZLINT

PURPOSE: To enable the locator device for input.

CALLING SEQUENCE: CALL ZLINT (IODLEN, IODES, IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

IERR [INTEGER; Output]
A return code indicating whether the locator device was successfully enabled.

IERR = 0 Locator device was successfully enabled.

= 1 Locator device is not supported on this work station. This occurs when the user loads a dummy device handler instead of the locator device handler. Call ignored.

= 2 The requested device is not at the I/O unit descriptor. This can occur for the following reasons:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O unit descriptor is not the same as the one specified in the device handler.
- o The IODLEN specified is less than one or greater than 132.

Call ignored.

= 3 The specified physical device is an outspooled graphics display device. Call ignored.

ZLINT

ZLINT enables the logical locator device for input (ZLEND disables it) by associating the logical locator device with a physical device and initializing the device. (ZLINT implicitly makes the picture current before initializing the device.) The device name is set to the name of the physical device and the device status is set to 1 (enabled). The locator echo position is set to the default value (see ZLOCP).

If the specified I/O unit descriptor is not the same as that of an enabled display device, then the logical locator limits will be set to the default values for the particular locator used. If the I/O unit descriptor specified is the same as that of an enabled display device, then the logical locator limits are set to the current view surface limits. If the locator is currently enabled, ZLINT terminates the enabled device (using ZLEND) and continues. Be sure to disable the locator device before terminating the application program.

A logical input device cannot be enabled on the same physical device as an outspooled graphics display device (see ZDINT).

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) Locator device is not supported on this work station.
   ACTION: IERR set to 1 and rest of call ignored.

3) Locator device already enabled.
   ACTION: Enabled locator device is terminated and call continues.

4) Attempt to enable input on an outspooled graphics display device.
   ACTION: IERR set to 3 and rest of call ignored.

5) The locator device requested is not at the I/O unit descriptor specified.
   ACTION: IERR set to 2 and rest of call ignored.

ZVINT

PURPOSE: To enable the logical valuator device.

CALLING SEQUENCE: CALL ZVINT (IODLEN, IODES, IERR)

IODLEN [INTEGER; Input]
The number of characters in the device name stored in IODES.

IODES [INTEGER; Array; Input]
The name of the I/O unit descriptor, expressed as a character string.

IERR [INTEGER; Output]
A return code indicating whether the valuator device was successfully enabled.

IERR = 0 The valuator device was successfully enabled.

= 1 The valuator device is not supported on this work station. This occurs when the user loads a dummy device handler instead of the valuator device handler. Call ignored.

= 2 The requested device is not at the I/O unit descriptor. This can occur for the following reasons:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O unit descriptor is not the same as the one specified in the device handler.
- o The IODLEN specified is less than one or greater than 132.

Call ignored.

= 3 The specified physical device is an outspooled graphics display device. Call ignored.

ZVINT

ZVINT enables the logical valuator device for input (ZVEND disables it) by associating the logical valuator device with a physical device and initializing the device. (ZVINT implicitly makes the picture current before initializing the device.) The device name is set to the name of the physical device and the device status is set to 1 (enabled).

The valuator device should be enabled before it samples or requests data. If the valuator is currently enabled, ZVINT terminates the enabled device (using ZVEND) and continues.

A logical input device cannot be enabled on the same physical device as an outspooled graphics display device (see ZDINT).

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize the DGL system.

2) The valuator device is not supported on this work station.
   ACTION: IERR set to 1 and rest of call ignored.

3) The valuator device already enabled.
   ACTION: Enabled valuator device is terminated and call continues.

4) Attempt to enable input on an outspooled graphics display device.
   ACTION: IERR set to 3 and rest of call ignored.

5) The valuator device requested is not at the I/O unit descriptor specified.
   ACTION: IERR set to 2 and rest of call ignored.

Part III
Integer Interface

INTRODUCTION

The integer interface is a set of function calls which allow the user to specify primitives using integer coordinates. The integer and real interfaces may be used together; portions of the drawing may be done in integer and other portions in real.

The integer interface increases performance but is less accurate than the real interface. The difference between the results of these interfaces will be at most two device units.

ZIMOV

PURPOSE: To set the starting position to the world coordinate position specified.

CALLING SEQUENCE: ZIMOV (IX, IY)

IX [INTEGER; Input]
X-coordinate of the new starting position in integer world coordinates.

IY [INTEGER; Input]
Y-coordinate of the new starting position in integer world coordinates.

ZIMOV specifies a new starting position. It does this by setting the value of the starting position to the world coordinate system point specified by (IX,IY). This point will then be the starting position of the next output primitive.

Since the DGL system does not perform software clipping, setting the starting position to a point outside of the current view surface limits will produce device-dependent results.

ERROR CONDITIONS:

1) The DGL system is not initialized.
ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The logical graphics display is not currently enabled.
ACTION: Call ignored.

ZIDRW

PURPOSE: To draw a line from the starting position to the world coordinate position specified.

CALLING SEQUENCE: ZIDRW (IX, IY)

IX [INTEGER; Input]
Ending X-coordinate of the line to be drawn, in integer world coordinates.

IY [INTEGER; Input]
Ending Y-coordinate of the line to be drawn, in integer world coordinates.

A line is drawn from the starting position to the world coordinate specified by (IX,IY). The starting position is updated to (IX,IY) at the completion of this call.

The DGL system does not perform any software clipping on output primitives. If either the starting position or the point (IX,IY) is outside of the current view surface, then the resulting picture may not be well defined because each device handles out-of-range data in a different manner.

Drawing to the starting position generates the shortest line possible. Depending on the nature of the linestyle, nothing may appear on the graphics display surface. See ZLSTL for a complete description of how linestyle affects a particular point or vector.

The primitive attributes of linestyle, color, highlighting and linewidth apply to lines drawn using ZIDRW. However, the lines will appear with these attributes only if the graphics device is capable of applying them to lines. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) The graphics display is not enabled.
   ACTION: Call ignored.

ZIPLY

PURPOSE: To draw a sequence of lines beginning at a specified point.

CALLING SEQUENCE: ZIPLY (NPTS, IXVEC, IYVEC)

NPTS	[INTEGER; Input] Number of points in IXVEC, IYVEC.
IXVEC	[INTEGER ARRAY; Input] X-coordinates of the polyline points in integer world coordinates.
IYVEC	[INTEGER ARRAY; Input] Y-coordinates of the polyline points in integer world coordinates.

The subroutine ZIPLY provides the capability to draw a series of connected lines starting at a specified point. A complete object can be drawn by making one call to this subroutine. This call first sets the starting position to be the first elements in the IXVEC,IYVEC arrays. The line sequence begins at this point and is drawn to the second element in each array, then to the third and continues until NPTS-1 lines are drawn.

This call is equivalent to the following sequence of calls:

```
CALL ZIMOV (IXVEC(1),IYVEC(1))
CALL ZIDRW (IXVEC(2),IYVEC(2))
CALL ZIDRW (IXVEC(3),IYVEC(3))
             :
             :
CALL ZIDRW (IXVEC(NPTS),IYVEC(NPTS))
```

The DGL system does not perform software clipping on output primitives. If any of the points specified in IXVEC and IYVEC are outside of the current view surface limits, the resulting image will be device dependent.

The starting position is set to (IXVEC(NPTS), IYVEC(NPTS)) at the completion of this call.

If NPTS is set to 1, ZIPLY generates a move to the world coordinate point (IXVEC(1),IYVEC(1)).

It is the application program's responsibility to insure that IXVEC, and IYVEC are dimensioned to at least NPTS and that at least NPTS values are contained in each array.

Depending on the nature of the current linestyle, nothing may appear on the graphics display. See ZLSTL for a complete description of how linestyle affects a particular point or vector.

The primitive attributes of color, highlighting, linestyle, and linewidth apply to polylines. However, the polyline will appear with these attributes only if the graphics device is capable of applying them to polylines. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. A call to ZBEGN must be made to initialize the DGL system.

2) Graphics display is not enabled.
   ACTION: Call ignored.

3) NPTS <= 0 (Possibly NPTS is a real number.)
   ACTION: Call ignored.

ZIPGD

PURPOSE: To draw a device-dependent polygon.

CALLING SEQUENCE: ZIPGD (NPTS, IXVEC, IYVEC, OPCODE)

NPTS	[INTEGER; Input] Number of points in IXVEC, IYVEC.
IXVEC	[INTEGER ARRAY; Input] X-coordinates of the polygon.
IYVEC	[INTEGER ARRAY; Input] Y-coordinates of the polygon.
OPCODE	[INTEGER ARRAY; Input] Polygon opcodes for each vertex.

ZIPGD draws polygon sets in the current polygon style as accurately as possible, using the hardware capabilities of the graphics device. ZIPGI should be used to request that polygon sets be produced in a device-independent manner. Generally, ZIPGD draws polygon sets with greater speed and more economical use of memory, and ZIPGI draws them with greater accuracy and uniformity.

Polygon fill capabilities can vary widely between devices. A device may have no filling capabilities at all, may be able to perform only solid fill, or may be able to fill polygons with different fill densities and at different fill line orientations. Some devices may be able to apply their capabilities to single rectangles only, while some may handle the full range of concave and convex polygon sets. ZIPGD tries to match the device capabilities to the request.

ZIPGD limits the number of polygon vertices which the user may specify. For devices that do not support hardware filling, up to 500 polygon vertices will be accepted for scribing the outline of the polygon interior. ZIPGD may be further limited by the device in the number of vertices that can be filled. If the device does not support hardware filling, see the *Device Handlers Manual* for ZPGDD to determine how many vertices will be accepted for filling by the hardware. If the limit is exceeded, the polygon will not be filled. In the case where the polygon style specifies nondisplay of edges, to provide some visible output, ZIPGD will outline the polygon using the polygon interior fill attributes. However, only those edge segments specified as displayable by OPCODE will be drawn.

DGL automatically closes each member polygon with a non-visible edge segment if its last vertex does not coincide with its first. To display the closing edge segment, the first and last vertices must be coincident and OPCODE for the last vertex must equal 1.

The DGL system does not perform any software clipping on output primitives. If any of the points specified in IXVEC and IYVEC are outside of the current view surface, then the resulting picture may not be well defined because each device handles out of range data in a different manner.

At the completion of this call the starting position is set to the first vertex of the last member polygon in the set.

The attributes of color, polygon interior color, highlighting, linestyle, polygon interior linestyle, polygon style and linewidth apply to polygons. However, the polygons will appear with these attributes only if the graphics device is capable of applying them to polygons. Check the *Device Handlers Manual* for the exact capabilities of the display device used to determine the effect of all primitive attributes.

ERROR CONDITIONS:

1) The DGL system is not initialized.
   ACTION: Call ignored. Call ZBEGN to initialize DGL.

2) No graphic display device is enabled.
   ACTION: Call ignored. A call to ZDINT must be made to enable a device.

3) NPOINT is a non-positive integer.
   ACTION: Call ignored.

4) More vertices are specified in IXVEC and IYVEC than the current graphic display device can handle.
   ACTION: The polygon set is outlined.

5) OPCODE(1) does not equal 2.
   ACTION: Call ignored.

6) OPCODE(i) (i > 1) does not equal 0, 1, or 2.
   ACTION: Polygon set is drawn but results are unpredictable.

ZIPGI

PURPOSE: To draw a device-independent polygon.

CALLING SEQUENCE: ZIPGI (NPTS, IXVEC, IYVEC, OPCODE)

NPTS	[INTEGER; Input] Number of points in IXVEC, IYVEC.
IXVEC	[INTEGER ARRAY; Input] X-coordinates of the polygon.
IYVEC	[INTEGER ARRAY; Input] Y-coordinates of the polygon.
OPCODE	[INTEGER ARRAY; Input] Polygon opcodes for each vertex.

ZIPGI draws polygon sets in a device-independent manner. This is done in software unless the device is capable of producing polygon sets in the entire polygon style range described below. ZIPGD draws polygon sets using the hardware capabilities of the device. Generally, ZIPGI draws polygon sets with greater accuracy and uniformity, and ZIPGD draws them with greater speed and more economical use of memory.

ZIPGI limits the number of polygon vertices which the user may specify (500 maximum). If the limit is exceeded, the polygon will not be filled. In the case where the polygon style specifies nondisplay of edges, to provide some visible output, ZIPGI will outline the polygon using the polygon interior fill attributes. However, only those edge segments specified as displayable by OPCODE will be drawn.

DGL automatically closes each member polygon with a non-visible edge segment if its last vertex does not coincide with its first. To display the closing edge segment, the first and last vertices must be coincident and OPCODE for last vertex must equal 1.

The DGL system does not perform any software clipping on output primitives. If any of the points specified in IXVEC and IYVEC are outside of the current view surface, then the resulting picture may not be well defined because each device handles out-of-range data in a different manner.

At the completion of this call, the starting position is set to the first vertex of the last member polygon in the set.

Part No. 97084-90001
E1183

Printed in U.S.A.
Second Edition
November 1983