

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

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

Copyright (c) 1983 by HEWLETT-PACKARD COMPANY

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

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

Part I

System-Dependent Topics

Operating Environment.....	1
I/O Unit Descriptors.....	1
Programming Languages.....	1
FORTRAN.....	1
Pascal.....	3
C.....	6
Library Names.....	8
Linking DGL Programs.....	8
Color Table Size Change.....	9
Polygon Intercept Buffer Size Change.....	10
Performance.....	11
Outspooling Graphics.....	11
Keyboard Input.....	13
Special Keys.....	13
Type-Ahead.....	13
Outside-World Signals.....	14

Part II

System-Dependent Calls

Introduction.....	1
ZAINT.....	2
ZBINT.....	4
ZDINT.....	6
ZIWS.....	10
ZKINT.....	19
ZLINT.....	21
ZVINT.....	23

Part III

Integer Interface

Introduction.....	1
ZIMOV.....	2
ZIDRW.....	3
ZIPLY.....	4
ZIPGD.....	6
ZIPGI.....	8

DGL SUBROUTINES AND PARAMETERS

Z AEND (no parameters) Z AINT (IODLEN, IODES, <u>IERR</u>) Z ALPH (NCHARS, STRING) Z ASPK (XSIZE, YSIZE) Z BEGN (no parameters) Z BEND (no parameters) Z BINT (IODLEN, IODES, <u>IERR</u>) Z BMOD (OPCODE) Z BUTN (ECHO, <u>BUTTON</u>) Z COLM (MODEL) Z COLR (COLOR) Z CSIZ (WIDTH, HEIGHT) Z DCOL (COLOR, COLP 1, COLP 2, COLP 3) Z DEND (no parameters) Z DINT (IODLEN, IODES, CONTRL, <u>IERR</u>) Z DLIM (XMIN, XMAX, YMIN, YMAX, <u>IERR</u>) Z DPMM (WX, WY, <u>MMX</u>, <u>MMY</u>) Z DPST (PINDEX, DENSITY, ORIENT, EDGE) Z DRAW (WX, WY) Z END (no parameters) Z HIGH (HLIGHT) Z IACS (DWIDE, DHIGH, <u>AWIDE</u>, <u>AHIGH</u>) Z ICOL (COLOR, <u>COLP 1</u>, <u>COLP 2</u>, <u>COLP 3</u>) Z IDRW (IX, IY) Z IESC (OPCODE, ISIZE, RSIZE, <u>ILIST</u>, <u>RLIST</u>, <u>IERR</u>) Z IMOV (IX, IY) Z IPGD (NPTS, IXVEC, IYVEC, OPCODE) Z IPGI (NPTS, IXVEC, IYVEC, OPCODE) Z IPLY (NPTS, IXVEC, IYVEC) Z IPST (PINDEX, <u>DENSITY</u>, <u>ORIENT</u>, <u>EDGE</u>) Z IWS (OPCODE, SSIZE, ISIZE, RSIZE, <u>SLIST</u>, <u>ILIST</u>, <u>RLIST</u>, <u>IERR</u>)	Z KEND (no parameters) Z KINT (IODLEN, IODES, <u>IERR</u>) Z KYBD (ECHO, MAX, <u>ACTUAL</u>, <u>STRING</u>) Z LEND (no parameters) Z LINT (IODLEN, IODES, <u>IERR</u>) Z LLIM (XMIN, XMAX, YMIN, YMAX, <u>IERR</u>) Z LOCP (WX, WY) Z LPMM (WX, WY, <u>MMX</u>, <u>MMY</u>) Z LSTL (LSTYLE) Z LWID (LWIDTH) Z MARK (MARKNO) Z MCUR (no parameters) Z MOVE (WX, WY) Z NEWF (no parameters) Z OESC (OPCODE, ISIZE, RSIZE, <u>ILIST</u>, <u>RLIST</u>, <u>IERR</u>) Z PGDD (NPOINT, XVEC, YVEC, OPCODE) Z PGDI (NPOINT, XVEC, YVEC, OPCODE) Z PICL (COLOR) Z PILS (LSTYLE) Z POLY (NPTS, XVEC, YVEC) Z PSTL (PINDEX) Z SLOC (ECHO, <u>WX</u>, <u>WY</u>) Z SVAL (ECHO, SUBVAL, <u>VALUE</u>) Z TEXT (NCHARS, STRING) Z VEND (no parameters) Z VIEW (VXMIN, VXMAX, VYMIN, VYMAX) Z VINT (IODLEN, IODES, <u>IERR</u>) Z WIND (WXMIN, WXMAX, WYMIN, WYMAX) Z WLOC (ECHO, <u>LBUTN</u>, <u>WX</u>, <u>WY</u>) Z WVAL (ECHO, SUBVAL, <u>VBUTN</u>, <u>VALUE</u>)
--	--

Part I

System-Dependent Topics

OPERATING ENVIRONMENT

DGL is supported on the HP-UX operating system on the following HP 9000 processors:

- Series 500 - Model 520
- Series 500 - Model 530
- Series 500 - Model 540
- Series 200 - Model 226 with 98143U Upgrade Kit
- Series 200 - Model 236 with memory management
- Series 200 - Model 220 with memory management

I/O UNIT DESCRIPTORS

The I/O unit descriptor specified in the initialization calls is a device file. To create this file, use the `mknod` command (described in the *HP-UX System Manager's Manual*).

PROGRAMMING LANGUAGES

FORTRAN

Unless otherwise specified, the FORTRAN compiler defaults all integers to 32 bits, but DGL requires 16 bit integers. To assign 16 bit integers, use one of the following methods:

- (1) Use the compiler option, `$OPTION SHORT INTEGERS`.
- (2) 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'
```

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.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 routine 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.

<u>DGL expects</u>	<u>Pascal data type</u>
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].

<u>DGL expects</u>	<u>C data type</u>
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 definitions 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,&ierr);

    /*****
    /*   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 -lB0004 -lL0006 -lD0031 -lDIDD
```

For Pascal:

```
pc program.p -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77
```

For C:

```
cc program.c -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77 -lm
```

See the appropriate compiler manual and/or ld 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 that 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 handler (Bxxxx), a dummy device handler is implicitly loaded from DIDD.

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/zlctb.f.
```

2. Modify the copy using the instructions documented in the source code.
3. Compile the source module:

```
fc -c zlctb.f
```

4. Link the new color table in with the DGL program. Examples are shown for the various types of main programs.

FORTTRAN program:

```
fc program.f zlctb.o -lB0004 -lL0006 -lD0031 -lDIDD
```

Pascal program:

```
pc program.p zlctb.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77
```

C program:

```
cc program.c zlctb.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* intercepts 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, single fill lines or portions of the polygon set will not be filled. The intercept buffer has a 100 point default capacity, which can be changed only by modifying the source code.

To change the size of the polygon intercept buffer:

1. Make a copy of the source file:

```
/usr/lib/graphics/fortran/tlint.f.
```

2. Modify the copy using the instructions documented in the source code.
3. Compile the source module:

```
fc -c tlint.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 tlint.o -lB0004 -lL0006 -lD0031 -lDIDD
```

Pascal program:

```
pc program.p tlint.o -lB0004 -lL0006 -lD0031 -lDIDD -lF77 -lI77
```

C program:

```
cc program.c tlint.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.

<u>Key</u>	<u>Function</u>
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

ZAINT

PURPOSE: To enable the alphanumeric device.

CALLING SEQUENCE: CALL ZAIN (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 ZAIN before attempting to output alphanumeric data using ZALPH. ZAIN 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 ZAIN 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

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

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:

- o The incorrect I/O unit descriptor was specified.
- o The specified device is down.
- o The physical device at the I/O 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.



ZDINT

- = 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 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-coordinate of window.
 (2) = Maximum X-coordinate of window.
 (3) = Minimum Y-coordinate of window.
 (4) = Maximum Y-coordinate 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 undeterminable (e.g. lineprinter).

ZIWS

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

ZKINT

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

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

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

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

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.

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.



READER COMMENT SHEET

Device-independent Graphics Library Supplement for HP-UX Systems

97084-90001 Nov 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
19420 Homestead Road
Cupertino, California 95014

FOLD

FOLD





MANUAL PART NO. 97084-90001
Printed in U.S.A.
E1183

HEWLETT-PACKARD COMPANY
Engineering Productivity Division
19420 Homestead Road
Cupertino, California 95014