

# DEC GKS

---

## C Binding Reference Manual, Part 2

Order Number: AA-PQP5A-TE

**June 1992**

This manual describes the C binding functions provided for DEC GKS™.

**Revision/Update Information:** This is a new manual.

**Digital Equipment Corporation  
Maynard, Massachusetts**

---

**First Printing, March 1984**

**Revised, November 1984, May 1986, March 1987, April 1989, February 1990, June 1992**

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1984, 1986, 1987, 1989, 1990, 1992.

All Rights Reserved.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: DDIF, DEC, DEC GKS, DEC GKS-3D, DECnet, DECstation, DECwindows, LA75, LVP16, MicroVAX, ReGIS, VAX, VAX Ada, VAX BASIC, VAX C, VAX COBOL, VAX FORTRAN, VAX Pascal, VAXstation, VAXstation II, VAXstationII/GPX, VMS, VT125, VT240, VT241, VT330, VT340, ULTRIX, ULTRIX Worksystem Software, and the DIGITAL logo.

BASIC is a registered trademark of Dartmouth College. HP-GL, HP7475, HP7550, HP7580, HP7585, and Hewlett-Packard are trademarks of Hewlett-Packard Company. Motif and OSF/Motif are registered trademarks of Open Software Foundation, Inc. MPS-2000 is a trademark of Laser Graphics, Inc. PostScript is a registered trademark of Adobe Systems, Incorporated. Tektronix is a registered trademark of Tektronix, Inc.

ZK5681

This manual is available on CDROM.

This document was prepared using VAX DOCUMENT, Version 2.1.

---

# Contents

<b>Preface</b> .....	ix
<b>11 Inquiry Functions</b>	
11.1 Using the Inquiry Functions .....	11-2
11.1.1 The Error Status Argument .....	11-3
11.1.2 The Value Type Argument .....	11-4
11.1.3 Returned Lists .....	11-4
11.2 Function Descriptions .....	11-5
INQUIRE ASPECT SOURCE FLAGS .....	11-6
INQUIRE ASPECT SOURCE FLAGS 3 .....	11-7
INQUIRE CHARACTER BASE VECTOR .....	11-8
INQUIRE CHARACTER EXPANSION FACTOR .....	11-9
INQUIRE CHARACTER HEIGHT .....	11-10
INQUIRE CHARACTER SPACING .....	11-11
INQUIRE CHARACTER UP VECTOR .....	11-12
INQUIRE CHARACTER WIDTH .....	11-13
INQUIRE CHOICE DEVICE STATE .....	11-14
INQUIRE CHOICE DEVICE STATE 3 .....	11-17
INQUIRE CLIPPING .....	11-20
INQUIRE CLIPPING 3 .....	11-21
INQUIRE COLOUR FACILITIES .....	11-22
INQUIRE COLOUR MODEL .....	11-23
INQUIRE COLOUR MODEL FACILITIES .....	11-24
INQUIRE COLOUR REPRESENTATION .....	11-25
INQUIRE CURRENT HLHSR IDENTIFIER VALUE .....	11-26
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES .....	11-27
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3 .....	11-29
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER .....	11-32
INQUIRE CURRENT PICK IDENTIFIER VALUE .....	11-33
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES .....	11-34
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3 .....	11-36
INQUIRE DEFAULT CHOICE DEVICE DATA .....	11-38
INQUIRE DEFAULT CHOICE DEVICE DATA 3 .....	11-40
INQUIRE DEFAULT DEFERRAL STATE VALUES .....	11-42
INQUIRE DEFAULT LOCATOR DEVICE DATA .....	11-43
INQUIRE DEFAULT LOCATOR DEVICE DATA 3 .....	11-45
INQUIRE DEFAULT PICK DEVICE DATA .....	11-47

INQUIRE DEFAULT PICK DEVICE DATA 3 .....	11-49
INQUIRE DEFAULT STRING DEVICE DATA .....	11-51
INQUIRE DEFAULT STRING DEVICE DATA 3 .....	11-53
INQUIRE DEFAULT STROKE DEVICE DATA .....	11-55
INQUIRE DEFAULT STROKE DEVICE DATA 3 .....	11-57
INQUIRE DEFAULT VALUATOR DEVICE DATA .....	11-59
INQUIRE DEFAULT VALUATOR DEVICE DATA 3 .....	11-61
INQUIRE DISPLAY SPACE SIZE .....	11-63
INQUIRE DISPLAY SPACE SIZE 3 .....	11-64
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES .....	11-66
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES .....	11-67
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3 .....	11-69
INQUIRE EDGE COLOUR INDEX .....	11-71
INQUIRE EDGE FACILITIES .....	11-72
INQUIRE EDGE FLAG .....	11-73
INQUIRE EDGE REPRESENTATION .....	11-74
INQUIRE EDGETYPE .....	11-75
INQUIRE EDGEWIDTH SCALE FACTOR .....	11-76
INQUIRE FILL AREA COLOUR INDEX .....	11-77
INQUIRE FILL AREA FACILITIES .....	11-78
INQUIRE FILL AREA INDEX .....	11-79
INQUIRE FILL AREA INTERIOR STYLE .....	11-80
INQUIRE FILL AREA REPRESENTATION .....	11-81
INQUIRE FILL AREA STYLE INDEX .....	11-82
INQUIRE GENERALIZED DRAWING PRIMITIVE .....	11-83
INQUIRE GENERALIZED DRAWING PRIMITIVE 3 .....	11-84
INQUIRE HLHSR FACILITIES .....	11-85
INQUIRE HLHSR MODE .....	11-86
INQUIRE INPUT QUEUE OVERFLOW .....	11-87
INQUIRE LEVEL OF GKS .....	11-88
INQUIRE LINETYPE .....	11-89
INQUIRE LINEWIDTH SCALE FACTOR .....	11-90
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES .....	11-91
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 .....	11-92
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES .....	11-93
INQUIRE LIST OF COLOUR INDICES .....	11-94
INQUIRE LIST OF EDGE INDICES .....	11-95
INQUIRE LIST OF FILL AREA INDICES .....	11-96
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS .....	11-97
INQUIRE LIST OF PATTERN INDICES .....	11-98
INQUIRE LIST OF POLYLINE INDICES .....	11-99
INQUIRE LIST OF POLYMARKER INDICES .....	11-100

INQUIRE LIST OF TEXT INDICES .....	11-101
INQUIRE LIST OF VIEW INDICES .....	11-102
INQUIRE LOCATOR DEVICE STATE .....	11-103
INQUIRE LOCATOR DEVICE STATE 3 .....	11-108
INQUIRE MARKER SIZE SCALE FACTOR .....	11-113
INQUIRE MARKER TYPE .....	11-114
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES .....	11-115
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3 .....	11-116
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER .....	11-117
INQUIRE MORE SIMULTANEOUS EVENTS .....	11-118
INQUIRE NAME OF OPEN SEGMENT .....	11-119
INQUIRE NORMALIZATION TRANSFORMATION .....	11-120
INQUIRE NORMALIZATION TRANSFORMATION 3 .....	11-121
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES .....	11-122
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED ....	11-123
INQUIRE OPERATING STATE VALUE .....	11-124
INQUIRE PATTERN FACILITIES .....	11-125
INQUIRE PATTERN HEIGHT VECTOR .....	11-126
INQUIRE PATTERN REFERENCE POINT .....	11-127
INQUIRE PATTERN REFERENCE POINT AND VECTORS .....	11-128
INQUIRE PATTERN REPRESENTATION .....	11-129
INQUIRE PATTERN WIDTH VECTOR .....	11-130
INQUIRE PICK DEVICE STATE .....	11-131
INQUIRE PICK DEVICE STATE 3 .....	11-133
INQUIRE PIXEL .....	11-135
INQUIRE PIXEL ARRAY .....	11-136
INQUIRE PIXEL ARRAY DIMENSIONS .....	11-138
INQUIRE POLYLINE COLOUR INDEX .....	11-139
INQUIRE POLYLINE FACILITIES .....	11-140
INQUIRE POLYLINE INDEX .....	11-141
INQUIRE POLYLINE REPRESENTATION .....	11-142
INQUIRE POLYMARKER COLOUR INDEX .....	11-143
INQUIRE POLYMARKER FACILITIES .....	11-144
INQUIRE POLYMARKER INDEX .....	11-145
INQUIRE POLYMARKER REPRESENTATION .....	11-146
INQUIRE PREDEFINED COLOUR REPRESENTATION .....	11-147
INQUIRE PREDEFINED EDGE REPRESENTATION .....	11-148
INQUIRE PREDEFINED FILL AREA REPRESENTATION .....	11-149
INQUIRE PREDEFINED PATTERN REPRESENTATION .....	11-150
INQUIRE PREDEFINED POLYLINE REPRESENTATION .....	11-151
INQUIRE PREDEFINED POLYMARKER REPRESENTATION .....	11-152
INQUIRE PREDEFINED TEXT REPRESENTATION .....	11-153
INQUIRE PREDEFINED VIEW REPRESENTATION .....	11-154

INQUIRE SEGMENT ATTRIBUTES .....	11-155
INQUIRE SEGMENT ATTRIBUTES 3 .....	11-157
INQUIRE SET OF ACTIVE WORKSTATIONS .....	11-159
INQUIRE SET OF ASSOCIATED WORKSTATIONS .....	11-160
INQUIRE SET OF OPEN WORKSTATIONS .....	11-161
INQUIRE SET OF SEGMENT NAMES IN USE .....	11-162
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION .....	11-163
INQUIRE STRING DEVICE STATE .....	11-164
INQUIRE STRING DEVICE STATE 3 .....	11-166
INQUIRE STROKE DEVICE STATE .....	11-168
INQUIRE STROKE DEVICE STATE 3 .....	11-171
INQUIRE TEXT ALIGNMENT .....	11-175
INQUIRE TEXT COLOUR INDEX .....	11-176
INQUIRE TEXT EXTENT .....	11-177
INQUIRE TEXT EXTENT 3 .....	11-178
INQUIRE TEXT FACILITIES .....	11-179
INQUIRE TEXT FONT AND PRECISION .....	11-181
INQUIRE TEXT INDEX .....	11-182
INQUIRE TEXT PATH .....	11-183
INQUIRE TEXT REPRESENTATION .....	11-184
INQUIRE VALUATOR DEVICE STATE .....	11-186
INQUIRE VALUATOR DEVICE STATE 3 .....	11-188
INQUIRE VIEW FACILITIES .....	11-190
INQUIRE VIEW REPRESENTATION 3 .....	11-191
INQUIRE WORKSTATION CATEGORY .....	11-193
INQUIRE WORKSTATION CLASSIFICATION .....	11-194
INQUIRE WORKSTATION CONNECTION AND TYPE .....	11-195
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES .....	11-196
INQUIRE WORKSTATION MAXIMUM NUMBERS .....	11-198
INQUIRE WORKSTATION STATE .....	11-199
INQUIRE WORKSTATION TRANSFORMATION .....	11-200
INQUIRE WORKSTATION TRANSFORMATION 3 .....	11-202

## 12 Error-Handling Functions

12.1 Function Descriptions .....	12-2
EMERGENCY CLOSE GKS .....	12-3
ERROR HANDLING .....	12-4
ERROR LOGGING .....	12-5
12.2 Program Example .....	12-6

## A DEC GKS Error Messages

A.1 Operating State Errors .....	A-1
A.2 Workstation Errors .....	A-3
A.3 Transformation Function Errors .....	A-5
A.4 Attribute Function Errors .....	A-6
A.5 Output Function Errors .....	A-10
A.6 Segment Function Errors .....	A-11

A.7	Input Function Errors .....	A-12
A.8	Metafile Function Errors .....	A-14
A.9	Escape Function Errors .....	A-15
A.10	Miscellaneous Errors .....	A-15
A.11	System Errors .....	A-16
A.12	C Binding Errors .....	A-19
A.13	Implementation-Specific Errors .....	A-19
A.14	Fatal Errors .....	A-36

## B Constants

B.1	C Constants .....	B-1
B.2	C Function Identifiers .....	B-16
B.3	Error Constants .....	B-25

## C Program Example

C.1	Code Examples .....	C-1
C.2	C Program .....	C-3

## D Function Identifiers

## E Initial Attribute Values

E.1	Initial Polyline Attributes .....	E-1
E.2	Initial Polymarker Attributes .....	E-2
E.3	Initial Text Attributes .....	E-2
E.4	Initial Fill Area Attributes .....	E-3
E.5	Initial Fill Area Set Attributes .....	E-3
E.6	Initial Segment Attributes .....	E-4
E.7	Initial Normalization Transformation Settings .....	E-4

## F Differences in GKS Implementations

F.1	Global Differences .....	F-1
-----	--------------------------	-----

## Index

### Examples

12-1	Creating an Error Handler .....	12-6
C-1	C Program Example .....	C-4

### Tables

A-1	Operating State Errors .....	A-2
A-2	Workstation Errors .....	A-3
A-3	Transformation Function Errors .....	A-5
A-4	Attribute Function Errors .....	A-6
A-5	Output Function Errors .....	A-11
A-6	Segment Function Errors .....	A-11

A-7	Input Function Errors .....	A-12
A-8	Metafile Function Errors .....	A-14
A-9	Escape Function Errors .....	A-15
A-10	Miscellaneous Errors .....	A-16
A-11	System Errors .....	A-16
A-12	C Binding Errors .....	A-19
A-13	Implementation-Specific Errors .....	A-20
A-14	Fatal Errors .....	A-36
B-1	C Constants .....	B-1
B-2	C Function Identifiers .....	B-16
B-3	C Error Constants .....	B-26
C-1	Binding-Specific Code Examples .....	C-1
D-1	C Binding Function Identifiers .....	D-1
F-1	Global Differences .....	F-1



---

# Preface

This manual contains complete descriptions for the C binding functions provided for DEC GKS. Use this reference material to program DEC GKS on any supported operating system, using any of the languages supported by DEC GKS.

## Intended Audience

This manual is for programmers who have experience developing graphics applications in one of the languages supported by DEC GKS. They also should be familiar with the principles of programming DEC GKS, as described in the *DEC GKS User's Guide*.

## Structure of This Document

This manual is divided into two parts. Each chapter deals with a specific subject or group of functions, describing the syntax and arguments for each function. The appendixes provide additional information you may find useful. Part 1 includes the following chapters:

- Chapter 1 provides an introduction to DEC GKS.
- Chapter 2 provides information about DEC GKS and the VMS™ operating system.
- Chapter 3 provides information about DEC GKS and the ULTRIX™ operating system.
- Chapter 4 describes the functions you use to control DEC GKS and workstation environments.
- Chapter 5 describes the functions you use to generate output primitives.
- Chapter 6 describes the functions you use to generate attributes.
- Chapter 7 describes the functions you use to set up and perform normalization and workstation transformations.
- Chapter 8 describes the functions you use to store output primitives in segments.
- Chapter 9 describes the functions you use to accept input from workstations.
- Chapter 10 describes the functions you use to store graphic images as metafiles.

Part 2 includes the following chapters and appendixes:

- Chapter 11 describes the functions you use to inquire for information about the capabilities and state of the DEC GKS system.
- Chapter 12 describes the functions you use to handle errors.

- Appendix A lists DEC GKS error codes, along with the corresponding severity code and message for each one.
- Appendix B lists constants defined for the C binding interface.
- Appendix C provides an example program written in C using the C binding.
- Appendix D lists the DEC GKS functions and the corresponding C binding names.
- Appendix E lists specific input values that apply to all DEC GKS workstations that perform both input and output.
- Appendix F provides implementation-specific information about DEC GKS.

## Associated Documents

You may find the following documents useful when using DEC GKS:

- *DEC GKS User's Guide*—for programmers who need information that supplements the DEC GKS binding manuals
- *DEC GKS GKS\$ Binding Reference Manual*—for programmers who need specific syntax and argument descriptions for the GKS\$ binding
- *DEC GKS GKS3D\$ Binding Reference Manual*—for programmers who need specific syntax and argument descriptions for the GKS3D\$ binding
- *DEC GKS FORTRAN Binding Reference Manual*—for programmers who need specific syntax and argument descriptions for the FORTRAN binding
- *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*—for programmers who need information about specific devices
- *Building a Device Handler System for DEC GKS and DEC PHIGS*—for programmers who need to build workstation graphics handlers

The C language binding of GKS follows the standards and conventions of VAX C™. For information on the C language, see the VAX C documentation set.

## Conventions

The following conventions are used in this manual:

Convention	Meaning
<code>Return</code>	The symbol <code>RETURN</code> represents a single stroke of the Return key on a terminal.
<b>Boldface text</b>	Boldface text represents the introduction of a new term. In interactive examples, user input appears in boldface type.
<i>Italic text</i>	Italic text indicates a parameter name or a book name. DEC GKS description table and state list entry names, and workstation description table and state list entry names are also italicized.
UPPERCASE TEXT	Uppercase text indicates a DEC GKS function or symbol name.
.	A vertical ellipsis indicates that not all of the text of a program or program output is illustrated. Only relevant material is shown in the example.
...	A horizontal ellipsis indicates that additional arguments, options, or values can be entered. A comma preceding the ellipsis indicates that successive items must be separated by commas. Horizontal ellipses in illustrations indicate that there is information not illustrated that either precedes or follows the information included in the illustration itself.
[]	Square brackets, in function synopses and a few other contexts, indicate that a syntactic element is optional.



**Inquiry Functions**

**Insert tabbed divider here. Then discard this sheet.**



---

## Inquiry Functions

The DEC GKS inquiry functions allow you to obtain current and default values for the operating state, output function attributes, deferral and regeneration modes, transformations, segments, and device capabilities. DEC GKS writes the values from the state lists and description tables to the inquiry function arguments.

The following list describes the tables and lists that are sources of information for many of the inquiry functions:

<b>Table/List</b>	<b>Description</b>
GKS description table	<p>This table contains information about the DEC GKS implementation you are using, such as the level of GKS (with DEC GKS, level 2c), the number of available workstation types, the list of workstation types, and the maximum allowable open workstations.</p> <p>If you are transporting your programs from one implementation of GKS to another, you may need to inquire about the implementation level of GKS on a given system, so your program does not call unsupported functions.</p>
Workstation description table	<p>This type of table contains information about one particular workstation, such as the workstation type, the workstation category, the device-specific maximum coordinate values, and the different bundled output attribute values. Each graphics handler contains a workstation description table describing that particular device.</p> <p>If your DEC GKS application uses more than one workstation at a time, or if you are unsure of the capabilities of your workstation, you may need to inquire about the values contained in the workstation description table.</p>
GKS state list	<p>This list contains entries that specify the current DEC GKS values such as the set of open workstations (if any), the current normalization transformation number, and the current character height.</p> <p>If you need to check the alterable DEC GKS values, you may need to inquire about the values contained in the GKS state list.</p>

## Inquiry Functions

Table/List	Description
Workstation state list	<p>For each workstation you open, DEC GKS allocates space for a workstation state list. This list contains entries that specify whether output is “on hold” (deferred), whether or not the surface has to be redrawn to fulfill an output request, whether the workstation surface is “empty” as defined by GKS, and whether the picture on the surface represents all the requests for output by the application program.</p> <p>If you need information concerning the current state of a particular workstation, you may need to inquire about the values contained in the workstation state list.</p>
Segment state list	<p>When you create a segment, DEC GKS creates a segment state list. The segment state list contains entries that specify the segment name, the set of associated workstations, and the detectability of the segment.</p> <p>If you need information concerning a particular segment, you may need to inquire about the values contained in the segment state list.</p>

---

### Note

---

You cannot inquire from a VWS (VAX Workstation Software) workstation description table unless you are logged onto a system running DEC GKS and VWS.

---

The only other type of information obtained by the inquiry functions is information concerning the color and dimensions of one or more pixels on the workstation surface. To obtain this information, you can use the pixel inquiry functions.

Inquiry function calls are similar; consequently, only the INQUIRE . . . DEVICE STATE functions are illustrated in program examples. For complete examples that use calls to these input inquiry functions, see Chapter 9.

To gain an understanding of when to call certain DEC GKS inquiry functions, see the *DEC GKS User's Guide*. For more information concerning state lists and description tables, see Chapter 4.

## 11.1 Using the Inquiry Functions

The DEC GKS inquiry functions return information about the DEC GKS tables, lists, and state of the pixels on a given device, by writing values to arguments passed to the function. For example, consider the call:

```
ginqllevelgks ( level, error )
```

The two arguments to the function INQUIRE LEVEL OF GKS are passed as write-only parameters. If this function completes its task successfully, DEC GKS returns the value 0 in the write-only argument *error*. If this function encounters an error condition (see Section 11.1.1 for detailed information), DEC GKS returns an error status code in the *error* argument. This function returns the level of the DEC GKS implementation with which you are working in the write-only argument *level*.



## Inquiry Functions

### 11.1 Using the Inquiry Functions

Some of the inquiry functions have read-only arguments as well. For example, review the following syntax:

```
ginqlocst ( ws, dev, type, bufsize, state_size, state, error )
```

The first three arguments (*ws*, *dev*, *type*) are all read only; DEC GKS needs to know the workstation identifier, the device type, and the type of values to be returned to this function to return the proper values to the other arguments (see Section 11.1.2 for detailed information concerning the argument value type).

The function INQUIRE LOCATOR DEVICE STATE illustrates the usefulness of the inquiry functions when requesting input. If you wish to change one of the default input values, you have to assign values to all the input variables, one by one. This can be tedious if you only want to change one or two of the default variable values.

A practical way to initialize all the necessary variables with default input values is to pass the variables to the function INQUIRE LOCATOR DEVICE STATE. To initialize the values, do the following:

1. Call the function INQUIRE LOCATOR DEVICE STATE to initialize all the input variables.
2. Change the values of the variables you wish to change.
3. Pass all the variables to INITIALIZE LOCATOR.

For more information concerning the workstation identifier, see Chapter 4. For more information concerning the input device type or general input concepts, see Chapter 9.

#### 11.1.1 The Error Status Argument

DEC GKS inquiry functions never generate an error, but they can encounter error conditions. The value passed to the *error* argument determines whether the values passed to the remaining write-only arguments are valid.

Because the inquiry functions obtain values from the description tables and state lists, and because the description tables and state lists are not accessible unless you have called the proper DEC GKS control functions, the inquiry functions may or may not be able to access the values you need. There are other device-dependent situations that would cause a DEC GKS inquiry function to encounter an error condition.

If all values are available, the inquiry function returns the value 0 in the *error* argument.

If a value is not presently available, the inquiry function returns a number, corresponding to an appropriate DEC GKS error message, in the *error* argument. If the value passed to the *error* argument is anything other than the value 0, the values that the inquiry function passed to the remaining arguments are invalid.

For more information concerning the DEC GKS error messages and their numbers, see Appendix A. For more information concerning DEC GKS error handling, see Chapter 12.

## Inquiry Functions

### 11.1 Using the Inquiry Functions

#### 11.1.2 The Value Type Argument

Several of the inquiry functions that take their values from the workstation state list have a return type argument. This argument determines whether DEC GKS returns the exact values that you previously set in the application program, or returns the values the DEC GKS device handlers determine closely approximate the values you requested.

The possible values for this argument are:

Value	Description
GSET	If you specify this constant (or the value 0), the inquiry function returns the requested values exactly as specified in the application program. If you did not assign any values in the application program, the inquiry function returns the default values.
GREALIZED	If you specify this constant (or the value 1), and if you specified values in your application program that a particular workstation cannot fully support, the inquiry function returns the realized values that closely approximate the values you specified in the application program. If you did not assign any values in the application program, the inquiry function returns the default values.

For example, some devices support a limited number of pick aperture sizes (the size of the tracking prompt used for picking segments). A set aperture size is one set by the application program, and a realized size is used by the graphics handler. Using the function `INQUIRE PICK DEVICE STATE`, you can inquire about both types of values.

For more information concerning pick input, see Chapter 9. For a program example using inquiry functions, see the `INQUIRY_CBND.C` program in `GKS$EXAMPLES` or `/usr/lib/GKS/examples/inquiry_cbnd.c`.

#### 11.1.3 Returned Lists

Numerous inquiry functions return integer lists. You must allocate space for these lists and set up the pointers to these allocated spaces before calling the function. One of the functions that returns an integer list is `INQUIRE FILL AREA FACILITIES`. The following code segment demonstrates the allocation of space for a maximum of 10 hatch styles:

```
Gflfac      flfac;
Gintlist    intlist;
Gint        users_int_list[10];

flfac.types = &intlist;
intlist.integers = users_int_list;
```

You must also allocate space for the list of interior styles before calling the `INQUIRE FILL AREA FACILITIES` function. The C binding does not provide for the size of this list, so you must allocate an array of at least four `Gflinter` enumerateds. The following code segment demonstrates how to do this:

```
Gflinter     flinter_list[4];
flfac.interiors = flinter_list;
```

The following inquiry functions return integer lists:

- INQUIRE COLOUR MODEL FACILITIES
- INQUIRE DEFAULT CHOICE DEVICE DATA (3)
- INQUIRE DEFAULT LOCATOR DEVICE DATA (3)
- INQUIRE DEFAULT PICK DEVICE DATA (3)
- INQUIRE DEFAULT STRING DEVICE DATA (3)
- INQUIRE DEFAULT STROKE DEVICE DATA (3)
- INQUIRE DEFAULT VALUATOR DEVICE DATA (3)
- INQUIRE EDGE FACILITIES
- INQUIRE FILL AREA FACILITIES
- INQUIRE GENERALIZED DRAWING PRIMITIVE (3)
- INQUIRE HLHSR FACILITIES
- INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES (3)
- INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
- INQUIRE LIST OF COLOUR INDICES
- INQUIRE LIST OF EDGE INDICES
- INQUIRE LIST OF FILL AREA INDICES
- INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS
- INQUIRE LIST OF PATTERN INDICES
- INQUIRE LIST OF POLYLINE INDICES
- INQUIRE LIST OF POLYMARKER INDICES
- INQUIRE LIST OF TEXT INDICES
- INQUIRE LIST OF VIEW INDICES
- INQUIRE PIXEL ARRAY
- INQUIRE POLYLINE FACILITIES
- INQUIRE POLYMARKER FACILITIES
- INQUIRE SET OF ACTIVE WORKSTATIONS
- INQUIRE SET OF ASSOCIATED WORKSTATIONS
- INQUIRE SET OF OPEN WORKSTATIONS
- INQUIRE SET OF SEGMENT NAMES IN USE
- INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

## **11.2 Function Descriptions**

This section describes the DEC GKS inquiry functions in detail.

## INQUIRE ASPECT SOURCE FLAGS

---

## INQUIRE ASPECT SOURCE FLAGS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqasf (  
    Gasfs      *asflist,    /* (0) List of 13 aspect source flags */  
    Gint       *error      /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* ASPECT SOURCE FLAGS (constant) */  
    Gasf      ln_type;    /* line type */  
    Gasf      ln_width;   /* line width */  
    Gasf      ln_colour;  /* line color */  
    Gasf      mk_type;    /* marker type */  
    Gasf      mk_size;    /* marker size */  
    Gasf      mk_colour;  /* marker color */  
    Gasf      tx_fp;      /* text font and precision */  
    Gasf      tx_exp;     /* text expansion */  
    Gasf      tx_space;   /* text character spacing */  
    Gasf      tx_colour;  /* text color */  
    Gasf      fl_inter;   /* fill area interior style */  
    Gasf      fl_style;   /* fill area style index */  
    Gasf      fl_colour;  /* fill area color */  
} Gasfs;
```

### Constants

Data Type	Constant	Description
Gasf	GBUNDLED	Bundled attributes
	GINDIVIDUAL	Individual attributes

### Description

The INQUIRE ASPECT SOURCE FLAGS function returns the value for the *current list of aspect source flags* entry in the GKS state list.

The ASF indicates whether a particular primitive attribute is selected from an attribute bundle, or as an individual attribute selection.

### See Also

SET ASPECT SOURCE FLAGS

---

## INQUIRE ASPECT SOURCE FLAGS 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqasf3 (
    Gasfs3    *asfs,    /* (0) List of 17 aspect source flags */
    Gint      *error    /* (0) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* ASPECT FLAGS 3 (constant) */
    Gasf  ln_type;       /* line type */
    Gasf  ln_width;     /* line width */
    Gasf  ln_colour;    /* line color */
    Gasf  mk_type;      /* marker type */
    Gasf  mk_size;      /* marker size */
    Gasf  mk_colour;    /* marker color */
    Gasf  tx_fp;        /* text font and precision */
    Gasf  tx_exp;       /* text expansion */
    Gasf  tx_space;     /* text character spacing */
    Gasf  tx_colour;    /* text color */
    Gasf  fl_inter;     /* fill area interior style */
    Gasf  fl_style;     /* fill area style index */
    Gasf  fl_colour;    /* fill area color */
    Gasf  edge_flag;    /* edge flag */
    Gasf  edge_type;    /* edge type */
    Gasf  edge_width;   /* edge width */
    Gasf  edge_colour;  /* edge color */
} Gasfs3;
```

### Constants

Data Type	Constant	Description
Gasf	GBUNDLED	Bundled attributes
	GINDIVIDUAL	Individual attributes

### Description

The INQUIRE ASPECT SOURCE FLAGS 3 function returns the value for the *current list of aspect source flags 3* entry in the GKS state list.

The ASF indicates whether a particular primitive attribute is selected from an attribute bundle, or as an individual attribute selection.

### See Also

SET ASPECT SOURCE FLAGS 3

## INQUIRE CHARACTER BASE VECTOR

---

### INQUIRE CHARACTER BASE VECTOR

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqcharbase (  
    Gpoint    *base,        /* (0) Character base vector */  
    Gint      *error        /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct {    /* COORDINATE POINT */  
    Gfloat    x;    /* X coordinate */  
    Gfloat    y;    /* Y coordinate */  
} Gpoint;
```

#### Description

The INQUIRE CHARACTER BASE VECTOR function returns the principle base direction value for the text string.

The direction is a two-dimensional vector in the text plane specified in the text output.

#### See Also

SET CHARACTER UP VECTOR

---

## INQUIRE CHARACTER EXPANSION FACTOR

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqcharexpan (  
    Gfloat    *chexp,    /* (0) Character expansion factor */  
    Gint      *error     /* (0) Error indicator */  
)
```

### Description

The INQUIRE CHARACTER EXPANSION FACTOR function returns the value for the *current character expansion factor* entry in the GKS state list.

The character expansion factor specifies the deviation of character width from the defined nominal value.

### See Also

SET CHARACTER EXPANSION FACTOR

## INQUIRE CHARACTER HEIGHT

---

### INQUIRE CHARACTER HEIGHT

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqcharheight (  
    Gfloat    *height,    /* (0) Character height in WC points */  
    Gint      *error      /* (0) Error indicator */  
)
```

#### Description

The INQUIRE CHARACTER HEIGHT function returns the value for the text attribute *current character height* entry in the GKS state list.

The character height value is workstation independent, expressed in WC values, and used when creating subsequent TEXT output primitives.

#### See Also

SET CHARACTER HEIGHT



---

## INQUIRE CHARACTER SPACING

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqcharspace (  
    Gfloat    *chspc,    /* (0) Character spacing */  
    Gint      *error     /* (0) Error indicator */  
)
```

### Description

The INQUIRE CHARACTER SPACING function returns the value for the *current text spacing* entry in the GKS state list.

DEC GKS measures the spacing between characters as a fraction of the character height; adjusting character height proportionately adjusts spacing. The character spacing value 0.0 places the character bodies next to each other without any separating space contained in the font specification for the letter bodies. Whether the characters actually touch depends on the type of font you use. Positive spacing values increase the space between characters; negative values decrease the space. Using negative spacing values, you can overlap characters or reverse the text so that characters are written in the opposite direction.

### See Also

SET CHARACTER SPACING

## INQUIRE CHARACTER UP VECTOR

---

### INQUIRE CHARACTER UP VECTOR

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqcharup (  
    Gpoint    *up,          /* (0) Character up vector */  
    Gint      *error        /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat    x; /* X coordinate */  
    Gfloat    y; /* Y coordinate */  
} Gpoint;
```

#### Description

The INQUIRE CHARACTER UP VECTOR function returns the value for the geometric attribute *current character up vector* entry in the GKS state list. The character up vector gives the "up" direction of a character.

DEC GKS uses the value specified in the call to SET CHARACTER UP VECTOR for all subsequent calls to TEXT until another value is specified.

#### See Also

SET CHARACTER UP VECTOR  
TEXT

---

## INQUIRE CHARACTER WIDTH

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqcharwidth (  
    Gfloat      *width,      /* (0) Character width */  
    Gint        *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE CHARACTER WIDTH function returns the value for the geometric attribute *current character width* entry in the GKS state list. The character width is calculated using the value set by the SET CHARACTER EXPANSION FACTOR or SET CHARACTER HEIGHT function.

DEC GKS uses the value specified in the calls to SET CHARACTER HEIGHT or SET CHARACTER EXPANSION FACTOR for all subsequent calls to TEXT until another value is specified.

### See Also

SET CHARACTER EXPANSION FACTOR  
SET CHARACTER HEIGHT  
SET CHARACTER SPACING  
TEXT

## INQUIRE CHOICE DEVICE STATE

---

## INQUIRE CHOICE DEVICE STATE

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqchoicest (  
  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      dev,        /* (I) Choice device number */  
    Gint      bufsize,    /* (I) Allocated record buffer size */  
    Gint      *state_size, /* (O) Required record buffer size */  
    Gchoicest *state,     /* (O) Returned data structure */  
    Gint      *error      /* (O) Error indicator */  
  
)
```

### Data Structures

```
typedef struct {          /* CHOICE STATE */  
    Gimode      mode;     /* mode (constant) */  
    Gesw        esw;     /* echo switch (constant) */  
    Gchoice     choice;   /* choice data */  
    Gint        pet;     /* prompt and echo type */  
    Glimit      e_area;   /* echo area */  
    Gchoicerec  record;   /* choice data record */  
} Gchoicest;  
  
typedef struct {          /* CHOICE DATA */  
    Gcstat      status;   /* choice status (constant) */  
    Gint        choice;   /* choice number */  
} Gchoice;  
  
typedef struct {          /* COORDINATE LIMITS */  
    Gfloat      xmin;     /* X minimum limit */  
    Gfloat      xmax;     /* X maximum limit */  
    Gfloat      ymin;     /* Y minimum limit */  
    Gfloat      ymax;     /* Y maximum limit */  
} Glimit;  
  
typedef union {           /* CHOICE DATA RECORD */  
    Gchoicepet_0001      choicepet_1_datarec;  
    Gchoicepet0001      choicepet1_datarec;  
    Gchoicepet0002      choicepet2_datarec;  
    Gchoicepet0003      choicepet3_datarec;  
    Gchoicepet0004      choicepet4_datarec;  
    Gchoicepet0005      choicepet5_datarec;  
} Gchoicerec;  
  
typedef Gchoicepetneg0001 Gchoicepet_0001;  
  
typedef Gchoicepet0001 Gchoicepetneg0001;  
  
typedef struct {  
    Gint      number;     /* number of choice strings */  
    Gint      *lengths;   /* lengths of choice strings */  
    Gchar     **strings;   /* array of strings */  
    Gchar     *title_string; /* the title string */  
} Gchoicepet0001;
```

## INQUIRE CHOICE DEVICE STATE

```
typedef struct {
    Gint    number;          /* number of alternatives */
    Gprflag *enable;        /* array of prompts */
    Gchar   *title_string;  /* title string */
} Gchoicepet0002;

typedef struct {
    Gint    number;          /* number of choice strings */
    Gchar   **strings;      /* array of strings */
    Gchar   *title_string;  /* the title string */
} Gchoicepet0003;

typedef Gchoicepet0003 Gchoicepet0004;

typedef struct {
    Gint    seg;            /* segment name */
    Gint    number;        /* number of alternatives */
    Gint    *pickids;      /* array of pick identifiers */
    Gchar   *title_string; /* the title string */
} Gchoicepet0005;
```

### Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Triggered without choosing
Gprflag	GPROFF	Choice prompt input flag off
	GPRON	Choice prompt input flag on

### Description

The INQUIRE CHOICE DEVICE STATE function returns the current state of the given choice-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gchoicerec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the choice data record.

If you need all the state information, and *bufsize* is equal to the `sizeof(Gchoicerec)`, you must initialize the `Gchoicerec` structure before calling this function. Because the `Gchoicerec` structure depends on the PET, you first have to ask GKS for the current PET by calling this function with *bufsize* equal to 0. Assuming the function returns PET 1, you would then have to initialize the `Gchoicepet001` record as follows, and then call the function again:

- On input, the *number* field of structure `Gchoicepet0001` (*state*→*record.choicepet1\_datarec.number*) contains the maximum number of choices that can be returned. The number of choices should be large enough to hold all the entries. (To determine the maximum number of choices, call INQUIRE DEFAULT CHOICE DEVICE DATA with *data\_size* equal to 0.) On output, this field contains the number of choices returned.

## INQUIRE CHOICE DEVICE STATE

- The *\*lengths* field of structure `Gchoicepet0001` (*state*→*record.choicepet1\_datarec.lengths*) must point to an array of type `Gint` with *number* elements.
- The *\*\*strings* field of structure `Gchoicepet0001` (*state*→*record.choicepet1\_datarec.strings*) must point to an array of string buffer pointers of type `*Gchar` with *number* elements. Each string buffer pointer should be large enough to hold the choice string (256 characters).

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE CHOICE

SET CHOICE MODE

Example 9–3 for a program example using an `INQUIRE . . . DEVICE STATE` function

---

**INQUIRE CHOICE DEVICE STATE 3**
**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginqchoicest3 (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      dev,        /* (I) Choice device number */
    Gint      bufsize,    /* (I) Allocated record buffer size */
    Gint      *state_size, /* (O) Required record buffer size */
    Gchoicest3 *state,    /* (O) Returned data structure */
    Gint      *error      /* (O) Error indicator */
)

```

**Data Structures**

```
typedef struct {          /* CHOICE 3 STATE */
    Gimode      mode;    /* choice mode (constant) */
    Gesw        esw;    /* echo switch (constant) */
    Gchoice     choice;  /* choice data */
    Gint        pet;    /* prompt and echo type */
    Glimit3     e_vol;  /* echo volume */
    Gchoicerec  record; /* choice data record */
} Gchoicest3;

typedef struct {          /* CHOICE DATA */
    Gcstat      status; /* choice status (constant) */
    Gint        choice; /* choice number */
} Gchoice;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat      xmin;   /* X minimum limit */
    Gfloat      xmax;   /* X maximum limit */
    Gfloat      ymin;   /* Y minimum limit */
    Gfloat      ymax;   /* Y maximum limit */
    Gfloat      zmin;   /* Z minimum limit */
    Gfloat      zmax;   /* Z maximum limit */
} Glimit3;

typedef union {           /* CHOICE DATA RECORD */
    Gchoicepet_0001      choicepet_1_datarec;
    Gchoicepet0001      choicepet1_datarec;
    Gchoicepet0002      choicepet2_datarec;
    Gchoicepet0003      choicepet3_datarec;
    Gchoicepet0004      choicepet4_datarec;
    Gchoicepet0005      choicepet5_datarec;
} Gchoicerec;

typedef Gchoicepetneg0001 Gchoicepet_0001;
typedef Gchoicepet0001 Gchoicepetneg0001;

typedef struct {
    Gint      number;    /* number of choice strings */
    Gint      *lengths; /* lengths of choice strings */
    Gchar     **strings; /* array of strings */
    Gchar     *title_string; /* the title string */
} Gchoicepet0001;

```

## INQUIRE CHOICE DEVICE STATE 3

```
typedef struct {
    Gint    number;          /* number of alternatives */
    Gprflag *enable;        /* array of prompts */
    Gchar   *title_string;  /* title string */
} Gchoicepet0002;

typedef struct {
    Gint    number;          /* number of choice strings */
    Gchar   **strings;      /* array of strings */
    Gchar   *title_string;  /* the title string */
} Gchoicepet0003;

typedef Gchoicepet0003 Gchoicepet0004;

typedef struct {
    Gint    seg;            /* segment name */
    Gint    number;        /* number of alternatives */
    Gint    *pickids;      /* array of pick identifiers */
    Gchar   *title_string; /* the title string */
} Gchoicepet0005;
```

### Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Triggered without choosing
Gprflag	GPROFF	Choice prompt input flag off
	GPRON	Choice prompt input flag on

### Description

The INQUIRE CHOICE DEVICE STATE 3 function returns the current state of the given choice-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gchoicerec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the choice data record.

If you need all the state information, and *bufsize* is equal to the `sizeof(Gchoicerec)`, you must initialize the `Gchoicerec` structure before calling this function. Because the `Gchoicerec` structure depends on the PET, you first have to ask GKS for the current PET by calling this function with *bufsize* equal to 0. Assuming the function returns PET 1, you would then have to initialize the `Gchoicepet001` record as follows, and then call the function again:

- On input, the *number* field of structure `Gchoicepet0001` (*state*→*record.choicepet1\_datarec.number*) contains the maximum number of choices that can be returned. The number of choices should be large enough to hold all the entries. (To determine the maximum number of choices, call INQUIRE DEFAULT CHOICE DEVICE DATA 3 with *data\_size* equal to 0.) On output, this field contains the number of choices returned.



## INQUIRE CHOICE DEVICE STATE 3

- The *\*lengths* field of structure `Gchoicepet0001` (`state→record.choicepet1_datarec.lengths`) must point to an array of type `Gint` with *number* elements.
- The *\*\*strings* field of structure `Gchoicepet0001` (`state→record.choicepet1_datarec.strings`) must point to an array of string buffer pointers of type `*Gchar` with *number* elements. Each string buffer pointer should be large enough to hold the choice string (256 characters).

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE CHOICE 3

SET CHOICE MODE

Example 9–3 for a program example using an INQUIRE . . . DEVICE STATE function

## INQUIRE CLIPPING

---

### INQUIRE CLIPPING

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqclip (  
    Gcliprect    *clipping, /* (0) Clipping indicator and rectangle */  
    Gint         *error     /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct { /* CLIPPING RECTANGLE */  
    Gclip    ind; /* clipping indicator (constant) */  
    Glimit   rec; /* clipping rectangle */  
} Gcliprect;  
  
typedef struct { /* COORDINATE LIMITS */  
    Gfloat    xmin; /* X minimum limit */  
    Gfloat    xmax; /* X maximum limit */  
    Gfloat    ymin; /* Y minimum limit */  
    Gfloat    ymax; /* Y maximum limit */  
} Glimit;
```

#### Constants

Data Type	Constant	Description
Gclip	GNOCLIP	Clipping disabled
	GCLIP	Clipping enabled

#### Description

The INQUIRE CLIPPING function returns the value in NDC points of the current clipping rectangle.

#### See Also

SET CLIPPING INDICATOR

---

## INQUIRE CLIPPING 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqclip3 (
    Gcliprect3    *clipping, /* (0) Clipping indicator and volume */
    Gint          *error     /* (0) Error indicator pointer */
)
```

### Data Structures

```
typedef struct { /* NDC CLIPPING PARAMETERS */
    Gclip    ind; /* clipping indicator (constant) */
    Glimit3  volume; /* clipping volume */
} Gcliprect3;

typedef struct { /* COORDINATE LIMITS */
    Gfloat    xmin; /* X minimum limit */
    Gfloat    xmax; /* X maximum limit */
    Gfloat    ymin; /* Y minimum limit */
    Gfloat    ymax; /* Y maximum limit */
    Gfloat    zmin; /* Z minimum limit */
    Gfloat    zmax; /* Z maximum limit */
} Glimit3;
```

### Constants

Data Type	Constant	Description
Gclip	GNOCLIP	Clipping disabled
	GCLIP	Clipping enabled

### Description

The INQUIRE CLIPPING 3 function returns the value in NDC points of the current clipping volume.

### See Also

SET CLIPPING INDICATOR

## INQUIRE COLOUR FACILITIES

---

### INQUIRE COLOUR FACILITIES

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqcolourfacil (  
    Gwstype  *type,          /* (I) Workstation type. */  
    Gint     bufsize,        /* (I) Not used */  
    Gint     *fac_size,      /* (O) Not used */  
    Gcofac   *fac,           /* (O) Returned color facilities. If the  
                             function returns the value 0 for  
                             the number of colors, a continuous  
                             range of colors is available. */  
    Gint     *error          /* (O) Error indicator. */  
)
```

#### Data Structures

```
typedef struct {          /* COLOR FACILITIES */  
    Gint     colours;      /* number of colors */  
    Gcoavail coavail;      /* color availability (constant) */  
    Gint     predefined;   /* number of predefined bundles */  
} Gcofac;
```

#### Constants

Data Type	Constant	Description
Gcoavail	GCOLOUR	Color device
	GMONOCHROME	Monochrome device

#### Description

The INQUIRE COLOUR FACILITIES function returns the number of available colors, the color availability, and the number of predefined color bundles of a specified workstation.

---

## INQUIRE COLOUR MODEL

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqcolourmodel (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gint    *model,     /* (O) Color model */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Description

The INQUIRE COLOUR MODEL function returns the number of the current color model.

## INQUIRE COLOUR MODEL FACILITIES

---

## INQUIRE COLOUR MODEL FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqcolourmodelfacil (  
    Gwstype    *type,           /* (I) Workstation type */  
    Gint       max_models,      /* (I) Size of integer list */  
    Gint       start,          /* (I) Starting point of inquiry */  
    Gint       *default_model,  /* (O) Default color model */  
    Gintlist   *models,        /* (O) List of available models */  
    Gint       *actual_models,  /* (O) Total number of models available */  
    Gint       *error           /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {           /* INTEGER LIST */  
    Gint    number;        /* number of integers in list */  
    Gint    *integers;     /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE COLOUR MODEL FACILITIES function returns the number of available color models, a list of color models, and the default color model for the specified workstation type.

## INQUIRE COLOUR REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqcolourep (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      index,      /* (I) Color index */
    Ginttype  type,       /* (I) Return type (constant) */
    Gcobundl  *rep,       /* (O) Returned color representation */
    Gint      *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* COLOR BUNDLE */
    Gfloat    red;        /* red intensity */
    Gfloat    green;     /* green intensity */
    Gfloat    blue;      /* blue intensity */
} Gcobundl;
```

### Constants

Data Type	Constant	Description
Ginttype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.

### Description

The INQUIRE COLOUR REPRESENTATION function returns the color triplet values, which are the color coordinates of the current color model on the workstation. (See the SET COLOUR REPRESENTATION function in Chapter 6.)

If the specified color index is not in the color table on the specified workstation, and the specified type of returned values is REALIZED, the representation for color 1 is returned.

### See Also

SET COLOUR REPRESENTATION

## INQUIRE CURRENT HLHSR IDENTIFIER VALUE

---

### INQUIRE CURRENT HLHSR IDENTIFIER VALUE

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqhlhsrid (  
    Gint      *hlhsrid,      /* (0) Current HLHSR identifier value */  
    Gint      *error         /* (0) Error indicator */  
)
```

#### Description

The INQUIRE CURRENT HLHSR IDENTIFIER VALUE function returns the current hidden line and hidden surface removal (HLHSR) identifier. Implementation of HLHSR is described in the *DEC GKS User's Guide*.



# INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

---

## INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqindivattr (  
    Gindivattr    *indivattr,    /* (0) Returned attributes */  
    Gint          *error         /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct {                /* INDIVIDUAL ATTRIBUTES */  
    Gint    ln_type;           /* line type (constant) */  
    Gfloat  ln_width;         /* line width */  
    Gint    ln_colour;        /* line color */  
    Gint    mk_type;          /* marker type (constant) */  
    Gfloat  mk_size;          /* marker size scale factor */  
    Gint    mk_colour;        /* marker color */  
    Gtxfp   tx_fontprec;      /* text font and precision */  
    Gfloat  ch_exp;           /* character expansion factor */  
    Gfloat  ch_space;         /* character spacing */  
    Gint    tx_colour;        /* text color */  
    Gflinter fl_interior;     /* fill area interior style (constant) */  
    Gint    fl_style;         /* fill area style */  
    Gint    fl_colour;        /* fill area color */  
    Gasfs  asfs;             /* aspect source flags */  
} Gindivattr;  
  
typedef struct {                /* TEXT FONT AND PRECISION */  
    Gint    font;            /* text font */  
    Gtxprec prec;           /* text precision (constant) */  
} Gtxfp;  
  
typedef struct {                /* ASPECT SOURCE FLAGS (constant) */  
    Gasf    ln_type;         /* line type */  
    Gasf    ln_width;        /* line width */  
    Gasf    ln_colour;       /* line color */  
    Gasf    mk_type;         /* marker type */  
    Gasf    mk_size;         /* marker size */  
    Gasf    mk_colour;       /* marker color */  
    Gasf    tx_fp;           /* text font and precision */  
    Gasf    tx_exp;          /* text expansion */  
    Gasf    tx_space;        /* text character spacing */  
    Gasf    tx_colour;       /* text color */  
    Gasf    fl_inter;        /* fill area interior style */  
    Gasf    fl_style;        /* fill area style index */  
    Gasf    fl_colour;       /* fill area color */  
} Gasfs;
```

# INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

## Constants

Data Type	Constant	Description
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.
Marker types	GMK_POINT	Dot.
	GMK_PLUS	Plus sign.
	GMK_STAR	Asterisk.
	GMK_O	Circle.
	GMK_X	Diagonal cross.
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
Gflinter	GHOLLOW	Hollow interior.
	GSOLID	Solid interior.
	GPATTERN	Patterned interior.
	GHATCH	Hatched interior.
Gasf	GBUNDLED	Bundled attributes.
	GINDIVIDUAL	Individual attributes.

## Description

The INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function returns the current DEC GKS individual primitive attribute values.

## See Also

SET ASPECT SOURCE FLAGS  
SET COLOUR REPRESENTATION  
SET FILL AREA STYLE INDEX

## INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqindivattr3 (
    Gindivattr3    *indivattr3,    /* (0) Returned attributes */
    Gint           *error          /* (0) Error indicator */
)
```

### Data Structures

```
typedef struct {
    Gint           ln_type;        /* INDIVIDUAL ATTRIBUTES 3 */
    Gfloat         ln_width;      /* current line type (constant) */
    Gint           ln_colour;     /* current line width scale factor */
    Gfloat         mk_type;       /* current line color */
    Gint           mk_size;       /* current marker type (constant) */
    Gfloat         mk_colour;     /* current marker size scale factor */
    Gint           tx_fontprec;   /* current marker color */
    Gtxfp          ch_exp;        /* current text font and precision */
    Gfloat         ch_space;      /* current character expansion factor */
    Gint           tx_colour;     /* current character spacing */
    Gflinter       fl_interior;   /* current text color */
    Gint           fl_style;      /* current fill interior style (constant)*/
    Gint           fl_colour;     /* current fill style index */
    Gedge_f        edge_flag;     /* current fill color */
    Gint           edge_type;     /* current edge flag (constant) */
    Gfloat         edge_width;    /* current edge type (constant) */
    Gint           edge_colour;   /* current edge width scale factor */
    Gasfs3         asf3;         /* current edge color */
} Gindivattr3;

typedef struct {
    Gint           font;         /* TEXT FONT AND PRECISION */
    Gtxprec        prec;        /* text font */
} Gtxfp;
/* text precision (constant) */
```

## INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3

```

typedef struct {          /* ASPECT SOURCE FLAGS 3 (constant) */
    Gasf    ln_type;      /* line type */
    Gasf    ln_width;     /* line width */
    Gasf    ln_colour;    /* line color */
    Gasf    mk_type;      /* marker type */
    Gasf    mk_size;      /* marker size */
    Gasf    mk_colour;    /* marker color */
    Gasf    tx_fp;        /* text font and precision */
    Gasf    tx_exp;       /* text expansion */
    Gasf    tx_space;     /* text character spacing */
    Gasf    tx_colour;    /* text color */
    Gasf    fl_inter;     /* fill area interior style */
    Gasf    fl_style;     /* fill area style index */
    Gasf    fl_colour;    /* fill area color */
    Gasf    edge_flag;    /* edge flag */
    Gasf    edge_type;    /* edge type */
    Gasf    edge_width;   /* edge width */
    Gasf    edge_colour;  /* edge color */
} Gasfs3;

```

### Constants

Data Type	Constant	Description
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.
Marker types	GMK_POINT	Dot.
	GMK_PLUS	Plus sign.
	GMK_STAR	Asterisk.
	GMK_O	Circle.
	GMK_X	Diagonal cross.
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
Gflinter	GHOLLOW	Hollow interior.
	GSOLID	Solid interior.
	GPATTERN	Patterned interior.
	GHATCH	Hatched interior.
Gasf	GBUNDLED	Bundled attributes.
	GINDIVIDUAL	Individual attributes.
Gedge_f	GEDGE_OFF	Edge off.
	GEDGE_ON	Edge on.
Edge types	GED_SOLID	Solid edge.
	GED_DASHED	Dashed edge.
	GED_DOTTED	Dotted edge.
	GED_DASHDOT	Dashed-dotted edge.

## INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3

### Description

The INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3 function returns the current DEC GKS individual primitive attribute values.

### See Also

SET ASPECT SOURCE FLAGS 3  
SET COLOUR REPRESENTATION  
SET FILL AREA STYLE INDEX

## INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

---

### INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqcurtrannum (  
    Gint      *tran,      /* (0) Current transformation number */  
    Gint      *error      /* (0) Error indicator */  
)
```

#### Description

The INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER function returns the normalization transformation number currently in effect.

#### See Also

SELECT NORMALIZATION TRANSFORMATION

---

## INQUIRE CURRENT PICK IDENTIFIER VALUE

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqcurpickid (  
    Gint      *pickid,      /* (0) Current pick identifier */  
    Gint      *error        /* (0) Error indicator */  
)
```

### Description

The INQUIRE CURRENT PICK IDENTIFIER VALUE function queries the GKS state list and returns the current GKS pick identifier.

### See Also

SET PICK IDENTIFIER

## INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

---

## INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqprimattr (  
    Gprimattr    *primattr,    /* (0) Returned attributes */  
    Gint         *error        /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* CURRENT PRIMITIVE ATTRIBUTES */  
    Gint    ln_index;     /* current line index */  
    Gint    mk_index;     /* current marker index */  
    Gint    tx_index;     /* current text index */  
    Gfloat  ch_height;    /* current character height */  
    Gpoint  ch_up;        /* current character up vector */  
    Gfloat  ch_width;     /* current character width */  
    Gpoint  ch_base;      /* current character base */  
    Gtxpath tx_path;      /* current text path (constant) */  
    Gtxalign tx_align;    /* current text alignment */  
    Gint    fl_index;     /* current fill area index */  
    Gpoint  pa_width;     /* current pattern width */  
    Gpoint  pa_height;    /* current pattern height */  
    Gpoint  pa_refpt;     /* current pattern reference point */  
} Gprimattr;  
  
typedef struct {          /* COORDINATE POINT */  
    Gfloat  x;           /* X coordinate */  
    Gfloat  y;           /* Y coordinate */  
} Gpoint;  
  
typedef struct {          /* TEXT ALIGNMENT */  
    Gtxhor  hor;         /* horizontal component (constant) */  
    Gtxver  ver;         /* vertical component (constant) */  
} Gtxalign;
```

### Constants

Data Type	Constant	Description
Gtxpath	GTP_RIGHT	Text string reads from left to right.
	GTP_LEFT	Text string reads from right to left.
	GTP_UP	Text string reads from bottom to top.
	GTP_DOWN	Text string reads from top to bottom.
Gtxhor	GAH_NORMAL	Normal.
	GAH_LEFT	Left.
	GAH_CENTRE	Center.
	GAH_RIGHT	Right.



## INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

Gtxver	GAV_NORMAL	Normal.
	GAV_TOP	Top.
	GAV_CAP	Cap.
	GAV_HALF	Half.
	GAV_BASE	Base.
	GAV_BOTTOM	Bottom.

### Description

The INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function returns the current bundle index for each output function and the current value for each of the geometric output attributes.

## INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3

---

### INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqprimattr3 (  
    Gprimattr3    *primattr3,    /* (0) Returned attributes */  
    Gint          *error         /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct { /* CURRENT PRIMITIVE ATTRIBUTES 3 */  
    Gint    ln_index; /* current line index */  
    Gint    mk_index; /* current marker index */  
    Gint    tx_index; /* current text index */  
    Gfloat  ch_height; /* current character height */  
    Gpoint  ch_up;     /* current character up vector */  
    Gfloat  ch_width;  /* current character width */  
    Gpoint  ch_base;   /* current character base vector */  
    Gtxpath tx_path;   /* current text path (constant) */  
    Gtxalign tx_align; /* current text alignment */  
    Gint    fl_index; /* current fill area index */  
    Gpoint  pa_width;  /* current pattern width vector */  
    Gpoint  pa_height; /* current pattern height vector */  
    Gpoint3 pa_refpt;  /* current pattern reference point */  
    Gpoint3 pa_vector1; /* current pattern reference vector 1 */  
    Gpoint3 pa_vector2; /* current pattern reference vector 2 */  
    Gint    ed_index; /* current edge index */  
    Gint    vw_index; /* current view index */  
} Gprimattr3;  
  
    typedef struct { /* COORDINATE POINT */  
        Gfloat  x; /* X coordinate */  
        Gfloat  y; /* Y coordinate */  
    } Gpoint;  
  
    typedef struct { /* TEXT ALIGNMENT */  
        Gtxhor  hor; /* horizontal component (constant) */  
        Gtxver  ver; /* vertical component (constant) */  
    } Gtxalign;  
  
    typedef struct { /* COORDINATE POINT */  
        Gfloat  x; /* X coordinate */  
        Gfloat  y; /* Y coordinate */  
        Gfloat  z; /* Z coordinate */  
    } Gpoint3;
```

## INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3

### Constants

Data Type	Constant	Description
Gtxpath	GTP_RIGHT	Text string reads from left to right.
	GTP_LEFT	Text string reads from right to left.
	GTP_UP	Text string reads from bottom to top.
	GTP_DOWN	Text string reads from top to bottom.
Gtxhor	GAH_NORMAL	Normal.
	GAH_LEFT	Left.
	GAH_CENTRE	Center.
	GAH_RIGHT	Right.
Gtxver	GAV_NORMAL	Normal.
	GAV_TOP	Top.
	GAV_CAP	Cap.
	GAV_HALF	Half.
	GAV_BASE	Base.
	GAV_BOTTOM	Bottom.

### Description

The INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3 function returns the current bundle index for each output function and the current value for each of the geometric output attributes.

## INQUIRE DEFAULT CHOICE DEVICE DATA

---

## INQUIRE DEFAULT CHOICE DEVICE DATA

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdefchoice (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gint         dev,        /* (I) Choice device number */  
    Gint         bufsize,    /* (I) Maximum number of PETs that will  
                             fit into buffer pets of  
                             the Gdefchoice data structure */  
    Gint         *data_size, /* (M) Size of the choice data record */  
    Gdefchoice   *data,      /* (O) Choice data structure pointer */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* DEFAULT CHOICE DATA */  
    Gint         choices; /* maximum number of choices */  
    Gintlist     *pets;   /* list of prompt and echo types */  
    Glimit       e_area;  /* default echo area */  
    Gchoicerec   record;  /* default choice data record */  
} Gdefchoice;  
  
typedef struct {         /* INTEGER LIST */  
    Gint         number; /* number of integers in list */  
    Gint         *integers; /* list of integers */  
} Gintlist;  
  
typedef struct {        /* COORDINATE LIMITS */  
    Gfloat       xmin;   /* X minimum limit */  
    Gfloat       xmax;   /* X maximum limit */  
    Gfloat       ymin;   /* Y minimum limit */  
    Gfloat       ymax;   /* Y maximum limit */  
} Glimit;  
  
typedef union {         /* CHOICE DATA RECORD */  
    Gchoicepet_0001    choicepet_1_datarec;  
    Gchoicepet0001    choicepet1_datarec;  
    Gchoicepet0002    choicepet2_datarec;  
    Gchoicepet0003    choicepet3_datarec;  
    Gchoicepet0004    choicepet4_datarec;  
    Gchoicepet0005    choicepet5_datarec;  
} Gchoicerec;  
  
typedef Gchoicepetneg0001 Gchoicepet_0001;  
typedef Gchoicepet0001 Gchoicepetneg0001;
```

## INQUIRE DEFAULT CHOICE DEVICE DATA

```
typedef struct {
    Gint    number;           /* number of choice strings */
    Gint    *lengths;        /* lengths of choice strings */
    Gchar   **strings;       /* array of strings */
    Gchar   *title_string;   /* the title string */
} Gchoicepet0001;
```

---

### Note

---

Choice strings returned for prompt and echo type 1 are not null terminated. Use the *lengths* array to determine the lengths of the strings.

---

## Description

The INQUIRE DEFAULT CHOICE DEVICE DATA function returns the default values for the specified choice-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. No title string is returned by this function.

Before calling this function, you must initialize the Gdefchoice structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the `sizeof(Gchoicerec)`, DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the choice data record. In the case *data\_size* equal to the `sizeof(Gchoicerec)`, the Gchoicerec data structure has to be initialized as follows:

- On input, the *number* field of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.number*) must contain the maximum number of choices that can be returned. The number of choices should be large enough to hold all the entries. This function returns the maximum number of choices (if *data\_size* = 0). On output, this element contains the number of choices returned.
- The field *\*lengths* of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.lengths*) must point to an array of type Gint with *number* elements.
- The field *\*\*strings* of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.strings*) must point to an array of string buffer pointers of type \*Gchar with *number* elements. Each string buffer pointer should be large enough to hold the choice string (256 bytes).

## INQUIRE DEFAULT CHOICE DEVICE DATA 3

---

## INQUIRE DEFAULT CHOICE DEVICE DATA 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdefchoice3 (  
    Gwstype      *type,          /* (I) Workstation type */  
    Gint         dev,           /* (I) Choice 3 device number */  
    Gint         bufsize,      /* (I) Maximum number of PETs that will  
                                fit into buffer pets of  
                                the Gdefchoice3 data structure */  
    Gint         *data_size,    /* (M) Size of the choice data record */  
    Gdefchoice3 *data,         /* (O) Choice data structure pointer */  
    Gint         *error        /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* DEFAULT CHOICE 3 DATA */  
    Gint         choices;    /* maximum number of choices */  
    Gintlist     *pets;     /* list of prompt and echo types */  
    Glimit3      e_vol;     /* default echo area */  
    Gchoicerec  record;    /* default choice data record */  
} Gdefchoice3;  
  
typedef struct {        /* INTEGER LIST */  
    Gint         number;    /* number of integers in list */  
    Gint         *integers; /* list of integers */  
} Gintlist;  
  
typedef struct {        /* COORDINATE LIMITS */  
    Gfloat       xmin;     /* X minimum limit */  
    Gfloat       xmax;     /* X maximum limit */  
    Gfloat       ymin;     /* Y minimum limit */  
    Gfloat       ymax;     /* Y maximum limit */  
    Gfloat       zmin;     /* Z minimum limit */  
    Gfloat       zmax;     /* Z maximum limit */  
} Glimit3;  
  
typedef union {         /* CHOICE DATA RECORD */  
    Gchoicepet_0001    choicepet_1_datarec;  
    Gchoicepet0001    choicepet1_datarec;  
    Gchoicepet0002    choicepet2_datarec;  
    Gchoicepet0003    choicepet3_datarec;  
    Gchoicepet0004    choicepet4_datarec;  
    Gchoicepet0005    choicepet5_datarec;  
} Gchoicerec;  
  
typedef Gchoicepetneg0001 Gchoicepet_0001;  
typedef Gchoicepet0001 Gchoicepetneg0001;  
  
typedef struct {  
    Gint         number;    /* number of choice strings */  
    Gint         *lengths;  /* lengths of choice strings */  
    Gchar        **strings; /* array of strings */  
    Gchar        *title_string; /* the title string */  
} Gchoicepet0001;
```

## Description

The INQUIRE DEFAULT CHOICE DEVICE DATA 3 function returns the default values for the specified choice-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. No title string is returned by this function.

Before calling this function, you must initialize the Gdefchoice3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof(Gchoicerec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the choice data record. In the case *data\_size* equal to the sizeof(Gchoicerec), the Gchoicerec data structure has to be initialized as follows:

- On input, the *number* field of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.number*) must contain the maximum number of choices that can be returned. The number of choices should be large enough to hold all the entries. This function returns the maximum number of choices (if *data\_size* = 0). On output, this element contains the number of choices returned.
- The field *\*lengths* of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.lengths*) must point to an array of type Gint with *number* elements.
- The field *\*\*strings* of structure Gchoicepet0001 (*data*→*record.choicepet1\_datarec.strings*) must point to an array of string buffer pointers of type \*Gchar with *number* elements. Each string buffer pointer should be large enough to hold the choice string (256 bytes).

# INQUIRE DEFAULT DEFERRAL STATE VALUES

---

## INQUIRE DEFAULT DEFERRAL STATE VALUES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdefdeferst (  
    Gwstype  *type,      /* (I) Workstation type */  
    Gdefer   *def,       /* (O) Deferral and implicit regeneration modes */  
    Gint     *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* DEFERRAL STATE */  
    Gdefmode  defmode;    /* deferral mode (constant) */  
    Girgmode  irgmode;    /* implicit regeneration mode (constant) */  
} Gdefex;
```

### Constants

Data Type	Constant	Description
Gdefmode	GASAP	Generate images as soon as possible.
	GBNIG	Generate images before the next interaction globally or before a sample or event input occurs.
	GBNIL	Generate images before the next interaction locally or before a sample or event input occurs.
	GASTI	Generate images at some time. The exact time is determined by the workstation.
Girgmode	GSUPPRESSED	Implicit regeneration is suppressed.
	GALLOWED	Implicit regeneration is allowed.

### Description

The INQUIRE DEFAULT DEFERRAL STATE VALUES function returns the default deferral and implicit regeneration modes.

### See Also

SET DEFERRAL STATE



---

**INQUIRE DEFAULT LOCATOR DEVICE DATA**

**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefloc (
    Gwstype      *type,      /* (I) Workstation type */
    Gint         dev,        /* (I) Locator device number */
    Gint         bufsize,    /* (I) Maximum number of PETs that will
                             fit in the buffer pets of
                             the Gdefloc data structure */
    Gint         *data_size, /* (M) Size of the locator data record */
    Gdefloc      *data,      /* (O) Locator data structure pointer */
    Gint         *error      /* (O) Error indicator */
)
    
```

**Data Structures**

```

typedef struct {          /* DEFAULT LOCATOR DATA */
    Gpoint    position;   /* initial locator position */
    Gintlist  *pets;      /* list of prompt and echo types */
    Glimit    e_area;     /* default echo area */
    Glocrec   record;     /* default locator data record */
} Gdefloc;

    typedef struct {      /* COORDINATE POINT */
        Gfloat    x;      /* X coordinate */
        Gfloat    y;      /* Y coordinate */
    } Gpoint;

    typedef struct {      /* INTEGER LIST */
        Gint      number; /* number of integers in list */
        Gint      *integers; /* list of integers */
    } Gintlist;

    typedef struct {      /* COORDINATE LIMITS */
        Gfloat    xmin;   /* X minimum limit */
        Gfloat    xmax;   /* X maximum limit */
        Gfloat    ymin;   /* Y minimum limit */
        Gfloat    ymax;   /* Y maximum limit */
    } Glimit;
    
```

## INQUIRE DEFAULT LOCATOR DEVICE DATA

```
typedef union {          /* LOCATOR DATA RECORD */
  Glocpet_0001          locpet_1_datarec;
  Glocpet_0002          locpet_2_datarec;
  Glocpet_0003          locpet_3_datarec;
  Glocpet_0004          locpet_4_datarec;
  Glocpet_0005          locpet_5_datarec;
  Glocpet_0006          locpet_6_datarec;
  Glocpet_0007          locpet_7_datarec;
  Glocpet_0008          locpet_8_datarec;
  Glocpet_0009          locpet_9_datarec;
  Glocpet_0010          locpet_10_datarec;
  Glocpet_0011          locpet_11_datarec;
  Glocpet_0012          locpet_12_datarec;
  Glocpet0001           locpet1_datarec;
  Glocpet0002           locpet2_datarec;
  Glocpet0003           locpet3_datarec;
  Glocpet0004           locpet4_datarec;
  Glocpet0005           locpet5_datarec;
  Glocpet0006           locpet6_datarec;
} Glocrec;

typedef struct {         /* LOCATOR prompt and echo types */
  Gchar *data;         /* device/implementation dependent data */
} Glocpet0001;
```

### Description

The INQUIRE DEFAULT LOCATOR DEVICE DATA function returns the default values for the specified locator-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefloc structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Glocrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the locator data record.

---

**INQUIRE DEFAULT LOCATOR DEVICE DATA 3**
**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefloc3 (
    Gwstype      *type,      /* (I) Workstation type */
    Gint         dev,        /* (I) Locator 3 device number */
    Gint         bufsize,    /* (I) Maximum number of PETs that will
                             fit in the buffer pets of
                             the Gdefloc3 data structure */
    Gint         *data_size, /* (M) Size of the locator data record */
    Gdefloc3     *data,      /* (O) Locator data structure pointer */
    Gint         *error      /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* DEFAULT LOCATOR DATA */
    Gpoint3    position;  /* initial locator position */
    Gintlist   *pets;     /* list of prompt and echo types */
    Glimit3    e_vol;     /* default echo volume */
    Glocrec    record;    /* default locator data record */
} Gdefloc3;

typedef struct { /* COORDINATE POINT */
    Gfloat    x; /* X coordinate */
    Gfloat    y; /* Y coordinate */
    Gfloat    z; /* Z coordinate */
} Gpoint3;

typedef struct { /* INTEGER LIST */
    Gint    number; /* number of integers in list */
    Gint    *integers; /* list of integers */
} Gintlist;

typedef struct { /* COORDINATE LIMITS */
    Gfloat    xmin; /* X minimum limit */
    Gfloat    xmax; /* X maximum limit */
    Gfloat    ymin; /* Y minimum limit */
    Gfloat    ymax; /* Y maximum limit */
    Gfloat    zmin; /* Z minimum limit */
    Gfloat    zmax; /* Z maximum limit */
} Glimit3;

```

## INQUIRE DEFAULT LOCATOR DEVICE DATA 3

```
typedef union {          /* LOCATOR DATA RECORD */
  Glocpet_0001          locpet_1_datarec;
  Glocpet_0002          locpet_2_datarec;
  Glocpet_0003          locpet_3_datarec;
  Glocpet_0004          locpet_4_datarec;
  Glocpet_0005          locpet_5_datarec;
  Glocpet_0006          locpet_6_datarec;
  Glocpet_0007          locpet_7_datarec;
  Glocpet_0008          locpet_8_datarec;
  Glocpet_0009          locpet_9_datarec;
  Glocpet_0010          locpet_10_datarec;
  Glocpet_0011          locpet_11_datarec;
  Glocpet_0012          locpet_12_datarec;
  Glocpet0001          locpet1_datarec;
  Glocpet0002          locpet2_datarec;
  Glocpet0003          locpet3_datarec;
  Glocpet0004          locpet4_datarec;
  Glocpet0005          locpet5_datarec;
  Glocpet0006          locpet6_datarec;
} Glocrec;

typedef struct {         /* LOCATOR prompt and echo types */
  Gchar  *data;         /* device/implementation dependent data */
} Glocpet0001;
```

### Description

The INQUIRE DEFAULT LOCATOR DEVICE DATA 3 function returns the default values for the specified locator-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefloc3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Glocrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the locator data record.

---

**INQUIRE DEFAULT PICK DEVICE DATA**
**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefpick (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         dev,           /* (I) Pick device number */
    Gint         bufsize,      /* (I) Maximum number of PETs that will
                               fit in the buffer pets of
                               the Gdefpick data structure */
    Gint         *data_size,    /* (M) Size of the locator data record */
    Gdefpick     *data,        /* (O) Pick data structure pointer */
    Gint         *error        /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* DEFAULT PICK DATA */
    Gintlist *pets;      /* list of prompt and echo types */
    Glimit   e_area;     /* default echo area */
    Gpickrec record;    /* default pick data record */
} Gdefpick;

typedef struct {         /* INTEGER LIST */
    Gint     number;     /* number of integers in list */
    Gint     *integers; /* list of integers */
} Gintlist;

typedef struct {        /* COORDINATE LIMITS */
    Gfloat   xmin;      /* X minimum limit */
    Gfloat   xmax;      /* X maximum limit */
    Gfloat   ymin;      /* Y minimum limit */
    Gfloat   ymax;      /* Y maximum limit */
} Glimit;

typedef union {         /* PICK DATA RECORD */
    Gpickpet0001 pickpet1_datarec;
    Gpickpet0002 pickpet2_datarec;
    Gpickpet0003 pickpet3_datarec;
} Gpickrec;

typedef struct {
    Gfloat   aperture; /* pick aperture in device coordinates */
    Gchar    *data;    /* device/implementation dependent data */
} Gpickpet0001;

```

**Description**

The INQUIRE DEFAULT PICK DEVICE DATA function returns the default values for the specified pick-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

## INQUIRE DEFAULT PICK DEVICE DATA

Before calling this function, you must initialize the `Gdefpick` structure. The field *\*integers* of structure `Gintlist` (*data*→*pets*→*integers*) must point to an array of type `Gint`. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the `sizeof(Gpickrec)`, DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the pick data record.

---

**INQUIRE DEFAULT PICK DEVICE DATA 3**
**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefpick3 (
    Gwstype      *type,      /* (I) Workstation type */
    Gint         dev,        /* (I) Pick 3 device number */
    Gint         bufsize,    /* (I) Maximum number of PETs that will
                             fit in the buffer pets of
                             the Gdefpick3 data structure */
    Gint         *data_size, /* (M) Size of the locator data record */
    Gdefpick3    *data,      /* (O) Pick data structure pointer */
    Gint         *error      /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* DEFAULT PICK 3 DATA */
    Gintlist  *pets;      /* list of prompt and echo types */
    Glimit3   e_vol;      /* default echo volume */
    Gpickrec  record;     /* default pick data record */
} Gdefpick3;

typedef struct {          /* INTEGER LIST */
    Gint      number;     /* number of integers in list */
    Gint      *integers;  /* list of integers */
} Gintlist;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat    xmin;       /* X minimum limit */
    Gfloat    xmax;       /* X maximum limit */
    Gfloat    ymin;       /* Y minimum limit */
    Gfloat    ymax;       /* Y maximum limit */
    Gfloat    zmin;       /* Z minimum limit */
    Gfloat    zmax;       /* Z maximum limit */
} Glimit3;

typedef union {          /* PICK DATA RECORD */
    Gpickpet0001 pickpet1_datarec;
    Gpickpet0002 pickpet2_datarec;
    Gpickpet0003 pickpet3_datarec;
} Gpickrec;

typedef struct {
    Gfloat    aperture;   /* pick aperture in device coordinates */
    Gchar     *data;      /* device/implementation dependent data */
} Gpickpet0001;

```

## INQUIRE DEFAULT PICK DEVICE DATA 3

### Description

The INQUIRE DEFAULT PICK DEVICE DATA 3 function returns the default values for the specified pick-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefpick3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gpickrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the pick data record.



---

## INQUIRE DEFAULT STRING DEVICE DATA

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdefstring (
    Gwstype      *type,      /* (I) Workstation type */
    Gint         dev,        /* (I) String device number */
    Gint         bufsize,    /* (I) Maximum number of PETs that will
                             fit in the buffer pets of
                             the Gdefstring data structure */
    Gint         *data_size, /* (M) Size of the locator data record */
    Gdefstring   *data,      /* (O) String data structure pointer */
    Gint         *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* DEFAULT STRING DATA */
    Gint      bufsize;    /* initial buffer size */
    Gintlist  *pets;      /* list of prompt and echo types */
    Glimit    e_area;     /* default echo area */
    Gstringrec record;    /* default string data record */
} Gdefstring;

typedef struct {          /* INTEGER LIST */
    Gint      number;     /* number of integers in list */
    Gint      *integers;  /* list of integers */
} Gintlist;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat    xmin;       /* X minimum limit */
    Gfloat    xmax;       /* X maximum limit */
    Gfloat    ymin;       /* Y minimum limit */
    Gfloat    ymax;       /* Y maximum limit */
} Glimit;

typedef union {           /* STRING DATA RECORD */
    Gstringpet0001
} Gstringrec;

typedef struct {
    Gint      bufsize;    /* buffer size */
    Gint      position;   /* initial cursor position */
    Gchar     *title_string; /* title string */
} Gstringpet0001;
```

### Description

The INQUIRE DEFAULT STRING DEVICE DATA function returns the default values for the specified string-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

## INQUIRE DEFAULT STRING DEVICE DATA

Before calling this function, you must initialize the `Gdefstring` structure. The field `*integers` of structure `Gintlist` (`data→pets→integers`) must point to an array of type `Gint`. The size `bufsize` of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the `data_size` argument. If `data_size` is equal to the `sizeof(Gstringrec)`, DEC GKS returns all the state information. If `data_size` is 0, DEC GKS returns all the information except the string data record.

---

**INQUIRE DEFAULT STRING DEVICE DATA 3**
**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefstring3 (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         dev,           /* (I) String 3 device number */
    Gint         bufsize,      /* (I) Maximum number of PETs that will
                               fit in the buffer pets of
                               the Gdefstring3 data structure */
    Gint         *data_size,    /* (M) Size of the locator data record */
    Gdefstring3  *data,        /* (O) String data structure pointer */
    Gint         *error        /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {                /* DEFAULT STRING 3 DATA */
    Gint         bufsize;      /* initial buffer size */
    Gintlist     *pets;        /* list of prompt and echo types */
    Glimit3      e_vol;        /* default echo volume */
    Gstringrec   record;      /* default string data record */
} Gdefstring3;

typedef struct {                /* INTEGER LIST */
    Gint         number;      /* number of integers in list */
    Gint         *integers;   /* list of integers */
} Gintlist;

typedef struct {                /* COORDINATE LIMITS */
    Gfloat       xmin;        /* X minimum limit */
    Gfloat       xmax;        /* X maximum limit */
    Gfloat       ymin;        /* Y minimum limit */
    Gfloat       ymax;        /* Y maximum limit */
    Gfloat       zmin;        /* Z minimum limit */
    Gfloat       zmax;        /* Z maximum limit */
} Glimit3;

typedef union {                /* STRING DATA RECORD */
    Gstringpet0001 stringpet1_datarec;
} Gstringrec;

typedef struct {
    Gint         bufsize;     /* buffer size */
    Gint         position;    /* initial cursor position */
    Gchar        *title_string; /* title string */
} Gstringpet0001;

```

## INQUIRE DEFAULT STRING DEVICE DATA 3

### Description

The INQUIRE DEFAULT STRING DEVICE DATA 3 function returns the default values for the specified string-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefstring3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gstringrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the string data record.

---

INQUIRE DEFAULT STROKE DEVICE DATA

**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```
ginqdefstroke (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         dev,           /* (I) Stroke device number */
    Gint         bufsize,      /* (I) Maximum number of PETs that will
                                fit in the buffer pets of
                                the Gdefstroke data structure */
    Gint         *data_size,    /* (M) Size of the locator data record */
    Gdefstroke   *data,        /* (O) Stroke data structure pointer */
    Gint         *error        /* (O) Error indicator */
)
```

**Data Structures**

```
typedef struct {          /* DEFAULT STROKE DATA */
    Gint         bufsiz;   /* buffer size */
    Gintlist     *pets;   /* list of prompt and echo types */
    Glimit       e_area;  /* default echo area */
    Gstrokevec   record;  /* default stroke data record */
} Gdefstroke;

typedef struct {         /* INTEGER LIST */
    Gint         number;  /* number of integers in list */
    Gint         *integers; /* list of integers */
} Gintlist;

typedef struct {        /* COORDINATE LIMITS */
    Gfloat       xmin;   /* X minimum limit */
    Gfloat       xmax;   /* X maximum limit */
    Gfloat       ymin;   /* Y minimum limit */
    Gfloat       ymax;   /* Y maximum limit */
} Glimit;

typedef union {        /* STROKE DATA RECORD */
    Gstrokepet0001 strokepet1_datarec;
    Gstrokepet0002 strokepet2_datarec;
    Gstrokepet0003 strokepet3_datarec;
    Gstrokepet0004 strokepet4_datarec;
} Gstrokevec;

typedef struct {
    Gint         bufsiz;  /* input buffer size */
    Gint         editpos; /* editing position */
    Gpoint       interval; /* X, Y interval */
    Gfloat       time;    /* time interval */
    Gchar        *data;   /* device/implementation dependent data */
} Gstrokepet0001;

typedef struct {       /* COORDINATE POINT */
    Gfloat       x;      /* X coordinate */
    Gfloat       y;      /* Y coordinate */
} Gpoint;
```

## INQUIRE DEFAULT STROKE DEVICE DATA

### Description

The INQUIRE DEFAULT STROKE DEVICE DATA function returns the default values for the specified stroke-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefstroke structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gstrokerec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the stroke data record.

## INQUIRE DEFAULT STROKE DEVICE DATA 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdefstroke3 (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         dev,           /* (I) Stroke 3 device number */
    Gint         bufsize,       /* (I) Maximum number of PETs that will
                                fit in the buffer pets of
                                the Gdefstroke3 data structure */
    Gint         *data_size,     /* (M) Size of the locator data record */
    Gdefstroke3  *data,         /* (O) Stroke data structure pointer */
    Gint         *error         /* (O) Error indicator */
)

```

### Data Structures

```
typedef struct {          /* DEFAULT STROKE 3 DATA */
    Gint     bufsize;    /* buffer size */
    Gintlist *pets;     /* list of prompt and echo types */
    Glimit3  e_vol;     /* default echo volume */
    Gstroke3 record;    /* default stroke data record */
} Gdefstroke3;

typedef struct {          /* INTEGER LIST */
    Gint     number;    /* number of integers in list */
    Gint     *integers; /* list of integers */
} Gintlist;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat   xmin;     /* X minimum limit */
    Gfloat   xmax;     /* X maximum limit */
    Gfloat   ymin;     /* Y minimum limit */
    Gfloat   ymax;     /* Y maximum limit */
    Gfloat   zmin;     /* Z minimum limit */
    Gfloat   zmax;     /* Z maximum limit */
} Glimit3;

typedef union {          /* STROKE DATA RECORD */
    Gstrokepet0001 strokepet1_datarec;
    Gstrokepet0002 strokepet2_datarec;
    Gstrokepet0003 strokepet3_datarec;
    Gstrokepet0004 strokepet4_datarec;
} Gstroke3rec;

typedef struct {
    Gint     bufsize;    /* input buffer size */
    Gint     editpos;    /* editing position */
    Gpoint   interval;  /* X, Y interval */
    Gfloat   time;       /* time interval */
    Gchar    *data;     /* device/implementation dependent data */
} Gstrokepet0001;

typedef struct {          /* COORDINATE POINT */
    Gfloat   x;         /* X coordinate */
    Gfloat   y;         /* Y coordinate */
} Gpoint;

```

## INQUIRE DEFAULT STROKE DEVICE DATA 3

### Description

The INQUIRE DEFAULT STROKE DEVICE DATA 3 function returns the default values for the specified stroke-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefstroke3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gstrokerec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the stroke data record.



## INQUIRE DEFAULT VALUATOR DEVICE DATA

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```

ginqdefval (
    Gwstype    *type,          /* (I) Workstation type */
    Gint       dev,            /* (I) Valuator device number */
    Gint       bufsize,        /* (I) Maximum number of PETs that will
                               fit in the buffer pets of
                               the Gdefval data structure */
    Gint       *data_size,     /* (M) Size of the locator data record */
    Gdefval    *data,          /* (O) Valuator data structure pointer */
    Gint       *error          /* (O) Error indicator */
)
    
```

### Data Structures

```

typedef struct {          /* DEFAULT VALUATOR DATA */
    Gfloat      value;     /* initial value */
    Gintlist    *pets;     /* list of prompt and echo types */
    Glimit      e_area;    /* default echo area */
    Gvalrec     record;    /* default valuator data record */
} Gdefval;

typedef struct {          /* INTEGER LIST */
    Gint        number;    /* number of integers in list */
    Gint        *integers; /* list of integers */
} Gintlist;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat      xmin;     /* X minimum limit */
    Gfloat      xmax;     /* X maximum limit */
    Gfloat      ymin;     /* Y minimum limit */
    Gfloat      ymax;     /* Y maximum limit */
} Glimit;

typedef union {          /* VALUATOR DATA RECORD */
    Gvalpet_0001    valpet_1_datarec;
    Gvalpet_0002    valpet_2_datarec;
    Gvalpet_0003    valpet_3_datarec;
    Gvalpet0001     valpet1_datarec;
    Gvalpet0002     valpet2_datarec;
    Gvalpet0003     valpet3_datarec;
} Gvalrec;

typedef struct {
    Gfloat      low;       /* low range limit */
    Gfloat      high;      /* high range limit */
    Gchar       *title_string; /* the title string */
} Gvalpet0001;
    
```

## INQUIRE DEFAULT VALUATOR DEVICE DATA

### Description

The INQUIRE DEFAULT VALUATOR DEVICE DATA function returns the default values for the specified valuator-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefval structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gvalrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the valuator data record.

---

**INQUIRE DEFAULT VALUATOR DEVICE DATA 3**
**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```

ginqdefval3 (
    Gwstype  *type,          /* (I) Workstation type */
    Gint     dev,           /* (I) Valuator 3 device number */
    Gint     bufsize,      /* (I) Maximum number of PETs that will
                           fit in the buffer pets of
                           the Gdefval3 data structure */
    Gint     *data_size,   /* (M) Size of the locator data record */
    Gdefval3 *data,        /* (O) Valuator data structure pointer */
    Gint     *error        /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* DEFAULT VALUATOR 3 DATA */
    Gfloat  value;       /* initial value */
    Gintlist *pets;     /* list of prompt and echo types */
    Glimit3 e_vol;      /* default echo volume */
    Gvalrec record;     /* default valuator data record */
} Gdefval3;

typedef struct {         /* INTEGER LIST */
    Gint  number;       /* number of integers in list */
    Gint  *integers;   /* list of integers */
} Gintlist;

typedef struct {        /* COORDINATE LIMITS */
    Gfloat  xmin;      /* X minimum limit */
    Gfloat  xmax;      /* X maximum limit */
    Gfloat  ymin;      /* Y minimum limit */
    Gfloat  ymax;      /* Y maximum limit */
    Gfloat  zmin;      /* Z minimum limit */
    Gfloat  zmax;      /* Z maximum limit */
} Glimit3;

typedef union {         /* VALUATOR DATA RECORD */
    Gvalpet_0001 valpet_1_datarec;
    Gvalpet_0002 valpet_2_datarec;
    Gvalpet_0003 valpet_3_datarec;
    Gvalpet0001 valpet1_datarec;
    Gvalpet0002 valpet2_datarec;
    Gvalpet0003 valpet3_datarec;
} Gvalrec;

typedef struct {
    Gfloat  low;        /* low range limit */
    Gfloat  high;       /* high range limit */
    Gchar  *title_string; /* the title string */
} Gvalpet0001;

```

## INQUIRE DEFAULT VALUATOR DEVICE DATA 3

### Description

The INQUIRE DEFAULT VALUATOR DEVICE DATA 3 function returns the default values for the specified valuator-class logical input device on the specified workstation.

Note that DEC GKS uses PET 1 as the default value. This function does not return a title string or any device- or implementation-dependent data, or both.

Before calling this function, you must initialize the Gdefval3 structure. The field *\*integers* of structure Gintlist (*data*→*pets*→*integers*) must point to an array of type Gint. The size *bufsize* of the array should be large enough to hold all the PETs.

The information returned by this function depends on the value of the *data\_size* argument. If *data\_size* is equal to the sizeof (Gvalrec), DEC GKS returns all the state information. If *data\_size* is 0, DEC GKS returns all the information except the valuator data record.

---

## INQUIRE DISPLAY SPACE SIZE

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```

ginqdisplaysize (
    Gwstype      *type,      /* (I) Workstation type */
    Gdpsize      *dspsz,     /* (O) Returned display size data */
    Gint         *error      /* (O) Error indicator */
)

```

### Data Structures

```

typedef struct {          /* DISPLAY SIZE */
    Gdevunits  units;    /* device coordinate units (constant) */
    Gpoint     device;   /* device coordinate size */
    Gipoint    raster;   /* raster unit size */
} Gdpsize;

typedef struct {         /* COORDINATE POINT */
    Gfloat     x;        /* X coordinate */
    Gfloat     y;        /* Y coordinate */
} Gpoint;

typedef struct {        /* INTEGER POINT */
    Gint       x;        /* X coordinate */
    Gint       y;        /* Y coordinate */
} Gipoint;

```

### Constants

Data Type	Constant	Description
Gdevunits	GDC_METRES	Meters
	GDC_OTHER	Other units

### Description

The INQUIRE DISPLAY SPACE SIZE function returns, for the specified workstation type, the display size in device coordinates, a flag specifying whether the device coordinate units are returned in meters or in other units, and the display size in raster units.

By comparing a workstation's raster units with its maximum display coordinates, you can determine the resolution of the workstation surface, and how the device coordinates are mapped onto the pixels of the device.

### See Also

Example 7-4 for a program example using the INQUIRE DISPLAY SPACE SIZE function

## INQUIRE DISPLAY SPACE SIZE 3

---

## INQUIRE DISPLAY SPACE SIZE 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqdisplaysize3 (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gdssize3     *dspsz3,    /* (O) Returned display size data */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct { /* DISPLAY SIZE 3 */  
    Gdevunits  units; /* device coordinate units (constant) */  
    Gpoint3    device; /* device coordinate size */  
    Gipoint3   raster; /* raster unit size */  
} Gdssize3;  
  
    typedef struct { /* COORDINATE POINT */  
        Gfloat    x; /* X coordinate */  
        Gfloat    y; /* Y coordinate */  
        Gfloat    z; /* Z coordinate */  
    } Gpoint3;  
  
    typedef struct { /* INTEGER POINT */  
        Gint      x; /* X coordinate */  
        Gint      y; /* Y coordinate */  
        Gint      z; /* Z coordinate */  
    } Gipoint3;
```

### Constants

Data Type	Constant	Description
Gdevunits	GDC_METRES	Meters
	GDC_OTHER	Other units

### Description

The INQUIRE DISPLAY SPACE SIZE 3 function returns, for the specified workstation type, the display size in device coordinates, a flag specifying whether the device coordinate units are returned in meters or in other units, and the display size in raster units.

By comparing a workstation's raster units with its maximum display coordinates, you can determine the resolution of the workstation surface, and how the device coordinates are mapped onto the pixels of the device.

### See Also

Example 7-4 for a program example using the INQUIRE DISPLAY SPACE SIZE function

# INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

---

## INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmodsegattr (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gmodseg      *dyn,       /* (O) Dynamic modification values */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* DYNAMIC SEGMENT ATTRIBUTE MODIFICATION */  
    Gmodtype      transform; /* transformation (constant) */  
    Gmodtype      appear;   /* appearing (turning visible) */  
    Gmodtype      disappear; /* disappearing (turning invisible) */  
    Gmodtype      highlight; /* highlighting */  
    Gmodtype      priority;  /* priority */  
    Gmodtype      addition;  /* addition of primitives to segment */  
    Gmodtype      deletion;  /* deletion of segment */  
} Gmodseg;
```

### Constants

Data Type	Constant	Description
Gmodtype	GIRG	Implicit regeneration
	GIMM	Immediate regeneration

### Description

The INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES function returns information concerning the ability of the workstation to dynamically generate segment transformations, visibility changes, highlighting changes, priority changes, primitive additions, and segment deletions.

If the workstation can dynamically change the display surface, DEC GKS displays the results of the segment attribute changes immediately. If the workstation cannot dynamically change the display surface, and the implicit regeneration mode is set to SUPPRESSED, DEC GKS waits until the next update of the display surface to regenerate the primitives contained in segments. Implicit regeneration is described in the *DEC GKS User's Guide*.

If an implicit regeneration is required, all output primitives not contained in a segment are lost.



# INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

---

## INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmodwsattr (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gmodws       *dyn,       /* (O) Returned data */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* DYNAMIC WS ATTRIBUTE MODIFICATION (constant) */  
    Gmodtype      line;   /* polyline bundle representation changeable */  
    Gmodtype      mark;   /* polymarker bundle representation changeable */  
    Gmodtype      text;   /* text bundle representation changeable */  
    Gmodtype      fill;   /* fill area bundle representation changeable */  
    Gmodtype      pat;    /* pattern representation changeable */  
    Gmodtype      colour; /* color representation changeable */  
    Gmodtype      wstran; /* workstation transformation changeable */  
} Gmodws;
```

### Constants

Data Type	Constant	Description
Gmodtype	GIRG	Implicit regeneration
	GIMM	Immediate regeneration

### Description

The INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function returns information describing the ability of the workstation to dynamically alter the display surface following a modification of a workstation attribute.

If the workstation can dynamically change the display surface, DEC GKS displays the results of attribute changes immediately. If the workstation cannot dynamically change the display surface and the implicit regeneration mode is set to SUPPRESSED, DEC GKS waits until the next update of the display surface to regenerate the primitives contained in segments. Implicit regeneration is described in the *DEC GKS User's Guide*.

If an implicit regeneration is required, all output primitives not contained in a segment are lost.

## INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

### See Also

SET COLOUR REPRESENTATION  
SET FILL AREA REPRESENTATION  
SET PATTERN REPRESENTATION  
SET POLYLINE REPRESENTATION  
SET POLYMARKER REPRESENTATION  
SET TEXT REPRESENTATION

Example 7-4 for a program example using the INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function

---

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3

**Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```
ginqmodwsattr3 (
    Gwstype      *type,      /* (I) Workstation type */
    Gmodws3      *dyn3,      /* (O) Returned data */
    Gint         *error      /* (O) Error indicator */
)
```

**Data Structures**

```
typedef struct {          /* DYNAMIC WS ATTRIBUTE 3 MODIFICATION (constant) */
    Gmodtype line;        /* polyline bundle representation changeable */
    Gmodtype mark;        /* polymarker bundle representation changeable */
    Gmodtype text;        /* text bundle representation changeable */
    Gmodtype fill;        /* fill area bundle representation changeable */
    Gmodtype pat;         /* pattern representation changeable */
    Gmodtype colour;      /* color representation changeable */
    Gmodtype wstran;      /* workstation transformation changeable */
    Gmodtype view;        /* view transformation changeable */
    Gmodtype edge;        /* edge bundle representation changeable */
    Gmodtype hlhsr;       /* hlhsr mode changeable */
} Gmodws3;
```

**Constants**

Data Type	Constant	Description
Gmodtype	GIRG	Implicit regeneration
	GIMM	Immediate regeneration

**Description**

The INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3 function returns information describing the ability of the workstation to dynamically alter the display surface following a modification of a workstation attribute.

If the workstation can dynamically change the display surface, DEC GKS displays the results of attribute changes immediately. If the workstation cannot dynamically change the display surface and the implicit regeneration mode is set to SUPPRESSED, DEC GKS waits until the next update of the surface to regenerate the primitives contained in segments. Implicit regeneration is described in the *DEC GKS User's Guide*.

If an implicit regeneration is required, all output primitives not contained in a segment are lost.

## INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3

### See Also

SET COLOUR REPRESENTATION  
SET FILL AREA REPRESENTATION  
SET PATTERN REPRESENTATION  
SET POLYLINE REPRESENTATION  
SET POLYMARKER REPRESENTATION  
SET TEXT REPRESENTATION

Example 7-4 for a program example using the INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function

---

## INQUIRE EDGE COLOUR INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqedgecolourind (  
    Gint    *index,      /* (0) Edge color index */  
    Gint    *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE EDGE COLOUR INDEX function queries the GKS state list and returns the current edge color index.

## INQUIRE EDGE FACILITIES

---

## INQUIRE EDGE FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqedgefacil (  
    Gwstype      *type,          /* (I) Workstation type */  
    Gint         bufsize,       /* (I) Size of integer list */  
    Gint         *num_et,       /* (O) Number of available edge types */  
    Gedgefac     *fac,          /* (O) Returned edge facilities */  
    Gint         *error         /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* EDGE FACILITIES */  
    Gintlist      *types;      /* list of edge types (constant) */  
    Gint          n_widths;    /* number of edge widths */  
    Gfloat        nom_width;   /* nominal edge width */  
    Gfloat        min_width;   /* minimum edge width */  
    Gfloat        max_width;   /* maximum edge width */  
    Gint          n_indices;   /* number of predefined edge indexes */  
} Gedgefac;  
  
    typedef struct {      /* INTEGER LIST */  
        Gint          number; /* number of integers in list */  
        Gint          *integers; /* list of integers */  
    } Gintlist;
```

### Constants

Data Type	Constant	Description
Edge types	GED_SOLID	Solid edge
	GED_DASHED	Dashed edge
	GED_DOTTED	Dotted edge
	GED_DASHDOT	Dashed-dotted edge

### Description

The INQUIRE EDGE FACILITIES function returns the edge flag; the number of edge types; a list of edge types; the number of edge widths; the nominal, minimum, and maximum edge widths; and the number of predefined edge index values.

If the number of edge widths returned is 0, the workstation supports a continuous range of edge widths.

---

## INQUIRE EDGE FLAG

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqedgeflag (
    Gedge_f   *edge_f,   /* (0) Edge flag (constant) */
    Gint      *error     /* (0) Error indicator */
)
```

### Constants

Data Type	Constant	Description
Gedge_f	GEDGE_OFF	Edge off
	GEDGE_ON	Edge on

### Description

The INQUIRE EDGE FLAG function returns the value for the *current edge flag* entry in the GKS state list.

This flag enables the display of subsequent fill area set output primitives when the current edge ASF has been set to INDIVIDUAL by the function SET ASPECT SOURCE FLAGS 3. If this ASF is set to BUNDLED, the current edge flag has no effect.

## INQUIRE EDGE REPRESENTATION

---

### INQUIRE EDGE REPRESENTATION

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqedgerep (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      index,      /* (I) Edge index */  
    Ginttype  type,       /* (I) Return type (constant) */  
    Gedgebundl *rep,      /* (O) Edge representation */  
    Gint      *error      /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* EDGE BUNDLE */  
    Gedge_f  flag;       /* edge flag (constant) */  
    Gint     type;       /* edge type (constant) */  
    Gfloat   width;      /* edge width scale factor */  
    Gint     colour;     /* edge color index */  
} Gedgebundl;
```

#### Constants

Data Type	Constant	Description
Ginttype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gedge_f	GEDGE_OFF	Edge off.
	GEDGE_ON	Edge on.
Edge types	GED_SOLID	Solid edge.
	GED_DASHED	Dashed edge.
	GED_DOTTED	Dotted edge.
	GED_DASHDOT	Dashed-dotted edge.

#### Description

The INQUIRE EDGE REPRESENTATION function returns the values associated with the given edge index value.

If the specified edge index is not in the edge bundle table on the specified workstation, and the type of returned values is REALIZED, this function returns the edge representation associated with edge index 1.



---

## INQUIRE EDGETYPE

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqedgetype (  
    Gint    *edgetype,    /* (0) Edge type (constant) */  
    Gint    *error        /* (0) Error indicator */  
)
```

### Constants

Defined Argument	Constant	Description
edgetype	GED_SOLID	Solid edge.
	GED_DASHED	Dashed edge.
	GED_DOTTED	Dotted edge.
	GED_DASHDOT	Dashed-dotted edge.

### Description

The INQUIRE EDGETYPE function queries the GKS state list and returns the current edge type.

## INQUIRE EDGEWIDTH SCALE FACTOR

---

## INQUIRE EDGEWIDTH SCALE FACTOR

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqewidth (
    Gfloat    *scale,      /* (0) Edge width scale factor */
    Gint      *error      /* (0) Error indicator */
)
```

### Description

The INQUIRE EDGEWIDTH SCALE FACTOR function queries the GKS state list and returns the current edgewidth scale factor.

---

## INQUIRE FILL AREA COLOUR INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqfillcolourind (  
    Gint    *index,      /* (0) Fill area color index */  
    Gint    *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE FILL AREA COLOUR INDEX function returns the value for the *current fill area color index* entry in the GKS state list.

The value of the *current fill area color index* entry controls the display of subsequent fill area set output primitives when the current fill color index ASF has been set to INDIVIDUAL by the function SET ASPECT SOURCE FLAGS 3. If this ASF is set to BUNDLED, the current fill area colour index has no effect.

### See Also

SET ASPECT SOURCE FLAGS 3  
SET FILL AREA COLOUR INDEX

# INQUIRE FILL AREA FACILITIES

---

## INQUIRE FILL AREA FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqfillfacil (  
    Gwstype    *type,        /* (I) Workstation type */  
    Gint       bufsize,     /* (I) Size of integer list */  
    Gint       *num_hs,     /* (O) Number of available hatch styles */  
    Gflfac     *fac,        /* (O) Returned fill facilities */  
    Gint       *error       /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {                /* FILL AREA FACILITIES */  
    Gint      n_interiors;     /* number of interior styles */  
    Gflinter  *interiors;     /* list of available interior styles (constant) */  
    Gintlist  *hatches;       /* list of available hatch styles */  
    Gint      predefined;     /* number of predefined bundles */  
} Gflfac;  
  
typedef struct {                /* INTEGER LIST */  
    Gint      number;         /* number of integers in list */  
    Gint      *integers;     /* list of integers */  
} Gintlist;
```

### Constants

Data Type	Constant	Description
Gflinter	GHOLLOW	Hollow interior
	GSOLID	Solid interior
	GPATTERN	Patterned interior
	GHATCH	Hatched interior

### Description

The INQUIRE FILL AREA FACILITIES function returns the number of available interior styles, the list of available interior styles, the number of hatch styles, the list of available hatch styles, and the number of fill area indexes available for a given workstation type.

---

## INQUIRE FILL AREA INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqfillind (  
    Gint    *index,      /* (0) Fill area index */  
    Gint    *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE FILL AREA INDEX function returns the value for the *current fill area index* entry in the GKS state list.

The fill area bundle table contains entries for the fill area interior style, fill area style index, and fill area color index attribute values. When calling FILL AREA, DEC GKS uses the bundle table only if the corresponding ASF has been set to BUNDLED.

### See Also

FILL AREA  
SET FILL AREA INDEX

## INQUIRE FILL AREA INTERIOR STYLE

---

### INQUIRE FILL AREA INTERIOR STYLE

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqfillintstyle (  
    Gflinter    *style,    /* (0) Fill area interior style (constant) */  
    Gint        *error     /* (0) Error indicator */  
)
```

#### Constants

Data Type	Constant	Description
Gflinter	GHOLLOW	Hollow interior
	GSOLID	Solid interior
	GPATTERN	Patterned interior
	GHATCH	Hatched interior

#### Description

The INQUIRE FILL AREA INTERIOR STYLE function queries the GKS state list and returns the current value for the fill area interior style index. The fill area interior style index determines whether a fill area will be drawn hollow, filled with a single color, or filled with a pattern or hatch design.

#### See Also

SET FILL AREA INTERIOR STYLE

## INQUIRE FILL AREA REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqfillrep (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      index,      /* (I) Fill area index */
    Ginttype  type,       /* (I) Return type (constant) */
    Gflbundl  *rep,       /* (O) Returned fill representation */
    Gint      *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* FILL AREA BUNDLE */
    Gflinter  inter;     /* fill area interior style (constant) */
    Gint      style;     /* fill area style index */
    Gint      colour;    /* fill area color index */
} Gflbundl;
```

### Constants

Data Type	Constant	Description
Ginttype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gflinter	GHOLLOW	Hollow interior.
	GSOLID	Solid interior.
	GPATTERN	Patterned interior.
	GHATCH	Hatched interior.

### Description

The INQUIRE FILL AREA REPRESENTATION function returns the values associated with the given fill area index value.

### See Also

SET FILL AREA REPRESENTATION

## INQUIRE FILL AREA STYLE INDEX

---

### INQUIRE FILL AREA STYLE INDEX

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqfillstyleind (  
    Gint    *index,    /* (0) Fill area style index */  
    Gint    *error     /* (0) Error indicator */  
)
```

#### Description

The INQUIRE FILL AREA STYLE INDEX function returns the *current fill area style index* entry in the GKS state list.

This function returns the value of the specific pattern or hatch style used to fill the interior of a polygonal fill area. If the interior style is hollow or solid, the current style index is ignored in a call to FILL AREA. If the interior style is pattern, a pattern index value is returned by this function. If the interior style is hatch, a hatch style value is returned by this function. For device-dependent hatch styles, the hatch style index is always a negative number.

#### See Also

FILL AREA  
SET FILL AREA STYLE INDEX



## INQUIRE GENERALIZED DRAWING PRIMITIVE

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```

ginqgdp (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         function,       /* (I) GDP identifier */
    Ggdpfac      *fac,          /* (O) Returned GDP facility data */
    Gint         *error          /* (O) Error indicator */
)
    
```

### Data Structures

```

typedef struct {          /* GDP FACILITIES */
    Gint     n_attrs;     /* number of attributes used */
    Gattrs   *attrs;     /* list of attributes used (constant) */
} Ggdpfac;
    
```

### Constants

Data Type	Constant	Description
Gattrs	GPOLYLINE	Polyline bundle attributes
	GPOLYMARKER	Polymarker bundle attributes
	GTEXT	Text bundle attributes
	GFILLAREA	Fill area bundle attributes
	GEDGE	Edge bundle attributes

### Description

The INQUIRE GENERALIZED DRAWING PRIMITIVE function returns the number of attribute sets, and the list of those attribute sets that are associated with the specified two-dimensional GDP identifier for a given workstation type.

## INQUIRE GENERALIZED DRAWING PRIMITIVE 3

---

### INQUIRE GENERALIZED DRAWING PRIMITIVE 3

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqgdp3 (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gint         function,   /* (I) GDP identifier */  
    Ggdpfac     *fac,       /* (O) Returned GDP 3 facility data */  
    Gint         *error      /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* GDP FACILITIES */  
    Gint     n_attrs;     /* number of attributes used */  
    Gattrs   *attrs;     /* list of attributes used (constant) */  
} Ggdpfac;
```

#### Constants

Data Type	Constant	Description
Gattrs	GPOLYLINE	Polyline bundle attributes
	GPOLYMARKER	Polymarker bundle attributes
	GTEXT	Text bundle attributes
	GFILLAREA	Fill area bundle attributes
	GEDGE	Edge bundle attributes

#### Description

The INQUIRE GENERALIZED DRAWING PRIMITIVE 3 function returns the number of attribute sets, and the list of those attribute sets that are associated with the specified three-dimensional GDP identifier for a given workstation type.

---

## INQUIRE HLHSR FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqhlhsrfac (
    Gwstype    *type,          /* (I) Workstation type */
    Gint       bufsize_types, /* (I) Size of integer list of identifiers */
    Gint       bufsize_modes, /* (I) Size of integer list of modes */
    Gint       *num_types,    /* (O) Number of available HLHSR identifiers */
    Gint       *num_modes,    /* (O) Number of available HLHSR modes */
    Ghlhsrfac  *fac,         /* (O) Returned facilities data */
    Gint       *error         /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* HLHSR FACILITIES */
    Gintlist  *types;     /* list of available types */
    Gintlist  *modes;     /* list of available modes */
} Ghlhsrfac;

typedef struct {         /* INTEGER LIST */
    Gint      number;     /* number of integers in list */
    Gint      *integers; /* list of integers */
} Gintlist;
```

### Description

The INQUIRE HLHSR FACILITIES function returns the number of available hidden line and hidden surface removal (HLHSR) identifiers and HLHSR modes, and a list of available HLHSR identifiers or HLHSR modes for the specified workstation type.

## INQUIRE HLHSR MODE

---

## INQUIRE HLHSR MODE

### Operating States

WSOP, WSAC, WGOP

### Syntax

```
ginqhlhsrmode (  
    Gint      ws,          /* (I) Workstation identifier */  
    Ghlhsrb   *mode,      /* (O) Update state, current and request mode */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* HLHSR MODE */  
    Ghlhsrus   state;     /* HLHSR update state */  
    Ghlhsrm   req_mode;   /* requested HLHSR mode */  
    Ghlhsrm   cur_mode;   /* current HLHSR mode */  
} Ghlhsrb;  
  
typedef Gwstus Ghlhsrus; /* HLHSR update state (constant)*/  
typedef Gint Ghlhsrm;   /* HLHSR MODE */
```

### Constants

Data Type	Constant	Description
Gwstus	GNOTPENDING	Action not pending
	G_PENDING	Action pending

### Description

The INQUIRE HLHSR MODE function returns the hidden line and hidden surface removal (HLHSR) update state, and the requested and current HLHSR modes.

If an HLHSR mode change was requested but not yet provided at the time of the call to this function, the HLHSR update state is PENDING.

## INQUIRE INPUT QUEUE OVERFLOW

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
gininputoverflow (
    Gqueue *overflow, /* (0) Workstation identifier, input class,
                       device number */
    Gint *error /* (0) Error indicator */
)
```

### Data Structures

```
typedef struct { /* QUEUE INFORMATION */
    Gint ws; /* workstation identifier */
    Giclass class; /* event class (constant) */
    Gint devno; /* logical input device number */
} Gqueue;
```

### Constants

Data Type	Constant	Description
Giclass	GNCLASS	No input class
	GLOCATOR	Locator input class
	GSTROKE	Stroke input class
	GVALUATOR	Valuator input class
	GCHOICE	Choice input class
	GPICK	Pick input class
	GSTRING	String input class
	GVIEWPORT	Viewport input class

### Description

The INQUIRE INPUT QUEUE OVERFLOW function queries the GKS error state list, and if the input queue has overflowed since the start of the session or since the last call to this function, it returns the identification of the logical input device that caused the overflow. The information is then removed from the GKS error state list.

### See Also

AWAIT EVENT  
FLUSH DEVICE EVENTS

## INQUIRE LEVEL OF GKS

---

### INQUIRE LEVEL OF GKS

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqllevelgks (  
    Glevel    *level,    /* (0) Level of GKS (constant).  DEC GKS is  
                        level 2c. */  
    Gint      *error     /* (0) Error indicator. */  
)
```

#### Constants

Data Type	Constant	Description
Glevel	GLMA	Minimal output, no input
	GLMB	Minimal output, request input
	GLMC	Minimal output, full input
	GL0A	All primitives and attributes, no input
	GL0B	All primitives and attributes, request input
	GL0C	All primitives and attributes, full input
	GL1A	Basic segmentation with full output, no input
	GL1B	Basic segmentation with full output, request input
	GL1C	Basic segmentation with full output, full input
	GL2A	Workstation-independent segment storage, no input
	GL2B	Workstation-independent segment storage, request input
	GL2C	Workstation-independent segment storage, full input

#### Description

The INQUIRE LEVEL OF GKS function returns the DEC GKS implementation level.

---

## INQUIRE LINETYPE

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqlinetype (
    Gint      *linetype,    /* (0) Line type (constant) */
    Gint      *error        /* (0) Error indicator */
)
```

### Constants

Data Type	Constant	Description
Line types	GLN_SOLID	Solid line
	GLN_DASHED	Dashed line
	GLN_DOTTED	Dotted line
	GLN_DASHDOT	Dashed-dotted line

### Description

The INQUIRE LINETYPE function returns the value for the *current polyline line type* entry in the GKS state list as solid, dashed, dotted, dashed-dotted, or any one of the device-dependent types.

Every workstation capable of output (DEC GKS category OUTPUT or OUTIN) defines at least four line types. For more information concerning possible polyline type values, see the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.

### See Also

SET LINETYPE

## INQUIRE LINEWIDTH SCALE FACTOR

---

### INQUIRE LINEWIDTH SCALE FACTOR

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqlinewidth (  
    Gfloat    *lnwidth,    /* (0) Line width scale factor */  
    Gint      *error       /* (0) Error indicator */  
)
```

#### Description

The INQUIRE LINEWIDTH SCALE FACTOR function returns the value for the *current polyline line width scale factor* entry in the GKS state list.

DEC GKS calculates line width as the nominal line width, multiplied by the line width scale factor. The line width scale factor is a real number that is passed to the SET LINEWIDTH SCALE FACTOR function. The workstation maps the value to the nearest available line width defined by the workstation.

#### See Also

SET LINEWIDTH SCALE FACTOR



# INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

---

## INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqavailgdp (  
    Gwstype    *type,          /* (I) Workstation type. */  
    Gint       max_gdps,      /* (I) Size of supplied buffer. */  
    Gint       start,        /* (I) Starting point of inquiry. Set it to  
                             0 to inquire from the beginning of the  
                             list. */  
    Gintlist   *gdps,        /* (O) List of available GDPs. */  
    Gint       *actual_gdps,  /* (O) Total number of GDPs available. */  
    Gint       *error        /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gint    *integers;   /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES function returns the number of available two-dimensional GDPs and a list of the GDP identifiers for the given workstation type.

## INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3

---

### INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqavailgdp3 (  
    Gwstype      *type,          /* (I) Workstation type. */  
    Gint         max_gdps,       /* (I) Size of supplied buffer. */  
    Gint         start,         /* (I) Starting point of inquiry. Set it to  
                                0 to inquire from the beginning of the  
                                list. */  
    Gintlist     *gdps,         /* (O) List of available GDPs. */  
    Gint         *actual_gdps,  /* (O) Total number of GDPs available. */  
    Gint         *error         /* (O) Error indicator. */  
)
```

#### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint     number;      /* number of integers in list */  
    Gint     *integers;   /* list of integers */  
} Gintlist;
```

#### Description

The INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 function returns the number of available three-dimensional GDPs and a list of the GDP identifiers for the given workstation type.

# INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

---

## INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqavailwstypes (  
    Gint      bufsize,      /* (I) Number of types that buffer can hold. */  
    Gint      start,        /* (I) Inquiry starting point in list. Set it to  
                            0 to inquire from the beginning of the  
                            list. */  
    Gintlist  *wstypes,     /* (O) Buffer to hold returned list. */  
    Gint      *actual_types, /* (O) Total number of available workstation  
                            types. */  
    Gint      *error        /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint      number;     /* number of integers in list */  
    Gint      *integers;  /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE LIST OF AVAILABLE WORKSTATION TYPES function queries the DEC GKS description table and returns the requested list of workstation types. You can use a returned type value as a workstation type argument to the OPEN WORKSTATION function, and you can pass it to inquiry functions that take a workstation type argument.

### See Also

OPEN WORKSTATION

## INQUIRE LIST OF COLOUR INDICES

---

## INQUIRE LIST OF COLOUR INDICES

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqcolourindices (  
    Gint    ws,           /* (I) Workstation identifier. */  
    Gint    max_indices, /* (I) Number of indexes that fit in buffer */  
    Gint    start,       /* (I) Starting point of inquiry. Set it to  
                          0 to inquire from the beginning of the  
                          list. */  
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */  
    Gint    *actual_indices, /* (O) Number of color indexes available. */  
    Gint    *error       /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gint    *integers;   /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE LIST OF COLOUR INDICES function returns the number and the list of defined color index values.

### See Also

SET COLOUR REPRESENTATION

---

**INQUIRE LIST OF EDGE INDICES**

**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```

ginqedgeindices (
    Gint    ws,           /* (I) Workstation identifier. */
    Gint    max_indices, /* (I) Number of indexes that fit in buffer. */
    Gint    start,       /* (I) Starting point of inquiry. Set it to 0
                        /* to inquire from the beginning of the list. */
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */
    Gint    *actual_indices, /* (O) Number of edge indexes available. */
    Gint    *error       /* (O) Error indicator. */
)
    
```

**Data Structures**

```

typedef struct {           /* INTEGER LIST */
    Gint    number;       /* number of integers in list */
    Gint    *integers;   /* list of integers */
} Gintlist;
    
```

**Description**

The INQUIRE LIST OF EDGE INDICES function returns the number and list of defined edge index values.

**See Also**

SET EDGE REPRESENTATION

## INQUIRE LIST OF FILL AREA INDICES

---

## INQUIRE LIST OF FILL AREA INDICES

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqfillindices (  
    Gint    ws,                /* (I) Workstation identifier. */  
    Gint    max_indices,      /* (I) Number of indexes that fit in buffer. */  
    Gint    start,            /* (I) Starting point of inquiry. Set it to 0  
                               to inquire from the beginning of the  
                               list. */  
    Gintlist *indices,        /* (O) Buffer holding returned indexes. */  
    Gint    *actual_indices, /* (O) Number of fill area indexes available. */  
    Gint    *error            /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gint    *integers;   /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE LIST OF FILL AREA INDICES function returns the number and list of defined fill area index values.

### See Also

SET FILL AREA REPRESENTATION

# INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

---

## INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqntrannum (  
    Gint    max_ntrans,    /* (I) Number that can fit in buffer. */  
    Gint    start,        /* (I) Starting point of inquiry. Set it to 0  
                           to inquire from the beginning of the list. */  
    Gintlist *ntrans,     /* (O) Buffer holding transformation numbers. */  
    Gint    *actual_ntrans, /* (O) Total number of transformations  
                           available. */  
    Gint    *error        /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gint    *integers;   /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS function returns the list of all defined normalization transformations in order of input viewport priority.

### See Also

SELECT NORMALIZATION TRANSFORMATION  
SET VIEWPORT  
SET WINDOW

## INQUIRE LIST OF PATTERN INDICES

---

### INQUIRE LIST OF PATTERN INDICES

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqpatindices (  
    Gint    ws,           /* (I) Workstation identifier. */  
    Gint    max_indices, /* (I) Number of indexes that fit in buffer. */  
    Gint    start,       /* (I) Starting point of inquiry. Set it to  
                          0 to inquire from the beginning of the  
                          list. */  
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */  
    Gint    *actual_indices, /* (O) Number of pattern indexes available. */  
    Gint    *error       /* (O) Error indicator. */  
)
```

#### Data Structures

```
typedef struct {           /* INTEGER LIST */  
    Gint    number;       /* number of integers in list */  
    Gint    *integers;    /* list of integers */  
} Gintlist;
```

#### Description

The INQUIRE LIST OF PATTERN INDICES function returns the number and the list of defined pattern index values.

#### See Also

SET PATTERN REPRESENTATION



---

**INQUIRE LIST OF POLYLINE INDICES**

**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginqlineindices (
    Gint    ws,           /* (I) Workstation identifier. */
    Gint    max_indices, /* (I) Number of indexes that fit in buffer. */
    Gint    start,       /* (I) Starting point of inquiry. Set it to 0
                          /* to inquire from the beginning of the list. */
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */
    Gint    *actual_indices, /* (O) Number of line indexes available. */
    Gint    *error       /* (O) Error indicator. */
)
```

**Data Structures**

```
typedef struct {           /* INTEGER LIST */
    Gint    number;       /* number of integers in list */
    Gint    *integers;    /* list of integers */
} Gintlist;
```

**Description**

The INQUIRE LIST OF POLYLINE INDICES function returns the number and list of defined polyline index values.

**See Also**

SET POLYLINE REPRESENTATION

## INQUIRE LIST OF POLYMARKER INDICES

---

### INQUIRE LIST OF POLYMARKER INDICES

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqmarkerindices (  
    Gint      ws,           /* (I) Workstation identifier. */  
    Gint      max_indices, /* (I) Number of indexes that fit in buffer. */  
    Gint      start,       /* (I) Starting point of inquiry. Set it to  
                           0 to inquire from the beginning of the  
                           list. */  
    Gintlist  *indices,    /* (O) Buffer holding returned indexes. */  
    Gint      *actual_indices, /* (O) Number of marker indexes available. */  
    Gint      *error       /* (O) Error indicator. */  
)
```

#### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint      number;    /* number of integers in list */  
    Gint      *integers; /* list of integers */  
} Gintlist;
```

#### Description

The INQUIRE LIST OF POLYMARKER INDICES function returns the number and list of defined polymarker index values.

#### See Also

SET POLYMARKER REPRESENTATION

---

**INQUIRE LIST OF TEXT INDICES**

**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginqtextindices (
    Gint    ws,           /* (I) Workstation identifier. */
    Gint    max_indices, /* (I) Number of indexes that fit in buffer. */
    Gint    start,       /* (I) Starting point of inquiry. Set it to 0
                          to inquire from the beginning of the
                          list. */
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */
    Gint    *actual_indices, /* (O) Number of line indexes available. */
    Gint    *error       /* (O) Error indicator. */
)
```

**Data Structures**

```
typedef struct { /* INTEGER LIST */
    Gint    number; /* number of integers in list */
    Gint    *integers; /* list of integers */
} Gintlist;
```

**Description**

The INQUIRE LIST OF TEXT INDICES function returns the number and list of defined text index values.

**See Also**

SET TEXT REPRESENTATION

## INQUIRE LIST OF VIEW INDICES

---

### INQUIRE LIST OF VIEW INDICES

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqviewindices (  
    Gint    ws,           /* (I) Workstation identifier. */  
    Gint    max_indices, /* (I) Number of indexes that fit in buffer. */  
    Gint    start,       /* (I) Starting point of inquiry. Set it to 0  
                        to inquire from the beginning of the  
                        list. */  
    Gintlist *indices,   /* (O) Buffer holding returned indexes. */  
    Gint    *actual_indices, /* (O) Number of view indexes available. */  
    Gint    *error       /* (O) Error indicator. */  
)
```

#### Data Structures

```
typedef struct { /* INTEGER LIST */  
    Gint    number; /* number of integers in list */  
    Gint    *integers; /* list of integers */  
} Gintlist;
```

#### Description

The INQUIRE LIST OF VIEW INDICES function returns the number and list of defined view index values.

#### See Also

SET VIEW REPRESENTATION 3

---

**INQUIRE LOCATOR DEVICE STATE**
**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```

ginqlocst (
    Gint    ws,          /* (I) Workstation identifier */
    Gint    dev,        /* (I) Locator device number */
    Gintqtype type,    /* (I) Return type (constant) */
    Gint    bufsize,   /* (I) Locator data record size (in bytes) */
    Gint    *state_size, /* (O) Required locator data record size */
    Glocst  *state,    /* (O) Returned data structure */
    Gint    *error     /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* LOCATOR STATE */
    Gimode mode;         /* mode (constant) */
    Gesw   esw;         /* echo switch (constant) */
    Gloc   loc;         /* locator data */
    Gint   pet;         /* prompt and echo type */
    Glimit e_area;     /* echo area */
    Glocrec record;    /* locator data record */
} Glocst;

    typedef struct {          /* LOCATOR DATA */
        Gint   transform;    /* normalization transformation number */
        Gpoint position;    /* locator position */
    } Gloc;

        typedef struct {          /* COORDINATE POINT */
            Gfloat x;        /* X coordinate */
            Gfloat y;        /* Y coordinate */
        } Gpoint;

            typedef struct {          /* COORDINATE LIMITS */
                Gfloat xmin;    /* X minimum limit */
                Gfloat xmax;    /* X maximum limit */
                Gfloat ymin;    /* Y minimum limit */
                Gfloat ymax;    /* Y maximum limit */
            } Glimit;

```

## INQUIRE LOCATOR DEVICE STATE

```
typedef union { /* LOCATOR DATA RECORD */
  Glocpet_0001 locpet_1_datarec;
  Glocpet_0002 locpet_2_datarec;
  Glocpet_0003 locpet_3_datarec;
  Glocpet_0004 locpet_4_datarec;
  Glocpet_0005 locpet_5_datarec;
  Glocpet_0006 locpet_6_datarec;
  Glocpet_0007 locpet_7_datarec;
  Glocpet_0008 locpet_8_datarec;
  Glocpet_0009 locpet_9_datarec;
  Glocpet_0010 locpet_10_datarec;
  Glocpet_0011 locpet_11_datarec;
  Glocpet_0012 locpet_12_datarec;
  Glocpet0001 locpet1_datarec;
  Glocpet0002 locpet2_datarec;
  Glocpet0003 locpet3_datarec;
  Glocpet0004 locpet4_datarec;
  Glocpet0005 locpet5_datarec;
  Glocpet0006 locpet6_datarec;
} Glocrec;

typedef Glocpetneg0001 Glocpet_0001;

typedef struct {
  Gfloat box_x; /* size of the box in x */
  Gfloat box_y; /* size of the box in y */
  Gchar *data; /* device/implementation dependent data */
} Glocpetneg0001;

typedef Glocpetneg0002 Glocpet_0002;

typedef Glocpet0005 Glocpetneg0002;

typedef struct {
  Gpfcf pfcf; /* polyline/fill area control flag
              (constant) */
  Gacf acf; /* attribute control flag
            (constant) */

  union {
    Glnattr ln; /* polyline attributes */
    Gflattr fl; /* fill area attributes */
  } attr;
  Gchar *data; /* device/implementation dependent
              data */
} Glocpet0005;

typedef struct { /* POLYLINE ATTRIBUTES */
  Gasf type; /* line type ASF (constant) */
  Gasf width; /* line width ASF */
  Gasf colour; /* line color ASF */
  Gint line; /* line index */
  Glnbundl bundl; /* line bundle */
} Glnattr;

typedef struct { /* POLYLINE BUNDLE */
  Gint type; /* line type (constant) */
  Gfloat width; /* linewidth scale factor */
  Gint colour; /* polyline colour index */
} Glnbundl;

typedef struct { /* FILL AREA ATTRIBUTES */
  Gasf inter; /* fill area interior style ASF */
  Gasf style; /* fill area style index ASF */
  Gasf colour; /* fill area color ASF */
  Gint fill; /* fill area index */
  Gflbundl bundl; /* fill area bundle */
} Gflattr;
```

## INQUIRE LOCATOR DEVICE STATE

```
typedef struct {          /* FILL AREA BUNDLE */
    Gflinter inter;      /* fill area interior style
                          (constant) */
    Gint style;          /* fill area style index */
    Gint colour;         /* fill area colour index */
} Gflbund1;

typedef Glocpetneg0003 Glocpet_0003;

typedef struct {
    Gacf acf;            /* attribute control flag (constant) */
    union {
        struct {
            Gpoint point1; /* point 1 for echo */
            Gpoint point2; /* point 2 for echo */
        } echo;
        struct {
            Glnattr ln;     /* polyline attributes */
            Gpoint point1; /* point 1 for echo */
            Gpoint point2; /* point 2 for echo */
        } lnecho;
    } attr;
    Gchar *data;         /* device/implementation dependent data */
} Glocpetneg0003;

typedef Glocpetneg0004 Glocpet_0004;

typedef struct {
    Gacf acf;            /* attribute control flag */
    Glnattr ln;          /* polyline attributes */
    Gchar *data;         /* device/implementation dependent data */
} Glocpetneg0004;

typedef Glocpetneg0005 Glocpet_0005;
    typedef Glocpetneg0004 Glocpetneg0005;
typedef Glocpetneg0006 Glocpet_0006;
    typedef Glocpetneg0003 Glocpetneg0006;
typedef Glocpetneg0003 Glocpetneg0006;
typedef Glocpetneg0007 Glocpet_0007;
    typedef Glocpetneg0003 Glocpetneg0007;
typedef Glocpetneg0008 Glocpet_0008;
    typedef Glocpetneg0003 Glocpetneg0008;
typedef Glocpetneg0009 Glocpet_0009;
    typedef Glocpetneg0004 Glocpetneg0009;
typedef Glocpetneg0010 Glocpet_0010;
    typedef Glocpetneg0004 Glocpetneg0010;
typedef Glocpetneg0011 Glocpet_0011;
    typedef Glocpet0001 Glocpetneg0011;
    typedef struct {      /* LOCATOR prompt and echo types */
        Gchar *data;     /* device/implementation dependent data */
    } Glocpet0001;
typedef Glocpetneg0012 Glocpet_0012;
    typedef Glocpetneg0004 Glocpetneg0012;
typedef Glocpet0001 Glocpet0002;
typedef Glocpet0001 Glocpet0003;
```

## INQUIRE LOCATOR DEVICE STATE

```
typedef struct {
    Gacf      acf;          /* attribute control flag */
    Glnattr   ln;          /* polyline attributes */
    Gchar     *data;       /* device/implementation dependent data */
} Glocpet0004;

typedef struct {
    Gchar     *title_string; /* the title string */
} Glocpet0006;
```

---

### Note

---

The locator data record is contained in the *record* field of the *Glocrec* structure.

---

## Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled.
	GNOECHO	Echo disabled.
Gpfcf	GPF_POLYLINE	Data record polyline control flag.
	GPF_FILLAREA	Data record fill area control flag.
Gacf	GCURRENT	Input data record current values.
	GSPECIFIED	Input data record specified values.
Gasf	GBUNDLED	Bundled attributes.
	GINDIVIDUAL	Individual attributes.
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.
Gflinter	GHOLLOW	Hollow interior.
	GSOLID	Solid interior.
	GPATTERN	Patterned interior.
	GHATCH	Hatched interior.

## Description

The INQUIRE LOCATOR DEVICE STATE function returns the current state of the given locator-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Glocrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the locator data record.



## INQUIRE LOCATOR DEVICE STATE

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE LOCATOR

SET LOCATOR MODE

Example 9-1 for a program example using the INQUIRE LOCATOR DEVICE STATE function

## INQUIRE LOCATOR DEVICE STATE 3

---

## INQUIRE LOCATOR DEVICE STATE 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqlcst3 (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gint    dev,         /* (I) Locator device number */  
    Ginqtype type,      /* (I) Return type (constant) */  
    Gint    bufsize,    /* (I) Locator data record size, in bytes */  
    Gint    *state_size, /* (O) Required locator data record size */  
    Glocst3 *state,     /* (O) Returned data structure */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

#### Glocst3

```
typedef struct { /* LOCATOR 3 STATE */  
    Gimode mode; /* locator mode (constant) */  
    Gesw esw; /* echo switch (constant) */  
    Gloc3 loc; /* locator data */  
    Gint pet; /* prompt and echo type */  
    Glimit3 e_vol; /* echo volume */  
    Glocrec record; /* locator data record */  
} Glocst3;  
  
    typedef struct { /* LOCATOR 3 DATA */  
        Gint transform; /* normalization transformation number */  
        Gint view; /* view index */  
        Gpoint3 position; /* locator position */  
    } Gloc3;  
  
        typedef struct { /* COORDINATE POINT */  
            Gfloat x; /* X coordinate */  
            Gfloat y; /* Y coordinate */  
            Gfloat z; /* Z coordinate */  
        } Gpoint3;  
  
        typedef struct { /* COORDINATE LIMITS */  
            Gfloat xmin; /* X minimum limit */  
            Gfloat xmax; /* X maximum limit */  
            Gfloat ymin; /* Y minimum limit */  
            Gfloat ymax; /* Y maximum limit */  
            Gfloat zmin; /* Z minimum limit */  
            Gfloat zmax; /* Z maximum limit */  
        } Glimit3;
```

## INQUIRE LOCATOR DEVICE STATE 3

```

typedef union {          /* LOCATOR DATA RECORD */
    Glocpet_0001        locpet_1_datarec;
    Glocpet_0002        locpet_2_datarec;
    Glocpet_0003        locpet_3_datarec;
    Glocpet_0004        locpet_4_datarec;
    Glocpet_0005        locpet_5_datarec;
    Glocpet_0006        locpet_6_datarec;
    Glocpet_0007        locpet_7_datarec;
    Glocpet_0008        locpet_8_datarec;
    Glocpet_0009        locpet_9_datarec;
    Glocpet_0010        locpet_10_datarec;
    Glocpet_0011        locpet_11_datarec;
    Glocpet_0012        locpet_12_datarec;
    Glocpet0001         locpet1_datarec;
    Glocpet0002         locpet2_datarec;
    Glocpet0003         locpet3_datarec;
    Glocpet0004         locpet4_datarec;
    Glocpet0005         locpet5_datarec;
    Glocpet0006         locpet6_datarec;
} Glocrec;

typedef Glocpetneg0001 Glocpet_0001;

typedef struct {
    Gfloat   box_x;      /* size of the box in x */
    Gfloat   box_y;      /* size of the box in y */
    Gchar    *data;      /* device/implementation dependent data */
} Glocpetneg0001;

typedef Glocpetneg0002 Glocpet_0002;

typedef Glocpet0005 Glocpetneg0002;

typedef struct {
    Gpfcf    pfcf;      /* polyline/fill area control flag
                        (constant) */
    Gacf     acf;      /* attribute control flag */
    union {
        Glnattr   ln;   /* polyline attributes */
        Gflattr   fl;   /* fill area attributes */
    } attr;
    Gchar    *data;      /* device/implementation dependent
                        data */
} Glocpet0005;

typedef struct {        /* POLYLINE ATTRIBUTES */
    Gasf     type;      /* line type ASF (constant) */
    Gasf     width;     /* line width ASF */
    Gasf     colour;    /* line color ASF */
    Gint     line;      /* line index */
    Glnbundl bundl;     /* line bundle */
} Glnattr;

typedef struct {        /* POLYLINE BUNDLE */
    Gint     type;      /* line type (constant) */
    Gfloat   width;     /* linewidth scale factor */
    Gint     colour;    /* polyline colour index */
} Glnbundl;

typedef struct {        /* FILL AREA ATTRIBUTES */
    Gasf     inter;     /* fill area interior style ASF */
    Gasf     style;     /* fill area style index ASF */
    Gasf     colour;    /* fill area color ASF */
    Gint     fill;      /* fill area index */
    Gflbundl bundl;     /* fill area bundle */
} Gflattr;

```

## INQUIRE LOCATOR DEVICE STATE 3

```
        typedef struct {          /* FILL AREA BUNDLE */
            Gflinter inter;        /* fill area interior style
                                   (constant) */
            Gint style;           /* fill area style index */
            Gint colour;         /* fill area colour index */
        } Gflbund1;
```

```
typedef Glocpetneg0003 Glocpet_0003;
```

```
    typedef struct {
        Gacf acf;                 /* attribute control flag (constant) */
        union {
            struct {
                Gpoint point1; /* point 1 for echo */
                Gpoint point2; /* point 2 for echo */
            } echo;
            struct {
                Glnattr ln;     /* polyline attributes */
                Gpoint point1; /* point 1 for echo */
                Gpoint point2; /* point 2 for echo */
            } lnecho;
        } attr;
        Gchar *data;            /* device/implementation dependent data */
    } Glocpetneg0003;
```

```
typedef Glocpetneg0004 Glocpet_0004;
```

```
    typedef struct {
        Gacf acf;                 /* attribute control flag */
        Glnattr ln;              /* polyline attributes */
        Gchar *data;            /* device/implementation dependent data */
    } Glocpetneg0004;
```

```
typedef Glocpetneg0005 Glocpet_0005;
```

```
    typedef Glocpetneg0004 Glocpetneg0005;
```

```
typedef Glocpetneg0006 Glocpet_0006;
```

```
    typedef Glocpetneg0003 Glocpetneg0006;
```

```
typedef Glocpetneg0003 Glocpetneg0006;
```

```
typedef Glocpetneg0007 Glocpet_0007;
```

```
    typedef Glocpetneg0003 Glocpetneg0007;
```

```
typedef Glocpetneg0008 Glocpet_0008;
```

```
    typedef Glocpetneg0003 Glocpetneg0008;
```

```
typedef Glocpetneg0009 Glocpet_0009;
```

```
    typedef Glocpetneg0004 Glocpetneg0009;
```

```
typedef Glocpetneg0010 Glocpet_0010;
```

```
    typedef Glocpetneg0004 Glocpetneg0010;
```

```
typedef Glocpetneg0011 Glocpet_0011;
```

```
    typedef Glocpet0001 Glocpetneg0011;
```

```
        typedef struct {          /* LOCATOR prompt and echo types */
            Gchar *data;         /* device/implementation dependent data */
        } Glocpet0001;
```

```
typedef Glocpetneg0012 Glocpet_0012;
```

```
    typedef Glocpetneg0004 Glocpetneg0012;
```

```
typedef Glocpet0001 Glocpet0002;
```

```
typedef Glocpet0001 Glocpet0003;
```

## INQUIRE LOCATOR DEVICE STATE 3

```
typedef struct {
    Gacf      acf;           /* attribute control flag */
    Glnattr   ln;          /* polyline attributes */
    Gchar     *data;       /* device/implementation dependent data */
} Glocpet0004;

typedef struct {
    Gchar     *title_string; /* the title string */
} Glocpet0006;
```

---

### Note

The locator data record is contained in the *record* field of the *Glocrec* structure.

---

## Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled
Gpfcf	GPF_POLYLINE	Data record polyline control flag.
	GPF_FILLAREA	Data record fill area control flag.
Gacf	GCURRENT	Input data record current values.
	GSPECIFIED	Input data record specified values.
Gasf	GBUNDLED	Bundled attributes.
	GINDIVIDUAL	Individual attributes.
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.
Gflinter	GHOLLOW	Hollow interior.
	GSOLID	Solid interior.
	GPATTERN	Patterned interior.
	GHATCH	Hatched interior.

## INQUIRE LOCATOR DEVICE STATE 3

### Description

The INQUIRE LOCATOR DEVICE STATE 3 function returns the current state of the given locator-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Glocrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the locator data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE LOCATOR 3

SET LOCATOR MODE

Example 9–1 for a program example using the INQUIRE LOCATOR DEVICE STATE function

---

## INQUIRE MARKER SIZE SCALE FACTOR

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmarkersize (  
    Gfloat      *mksize,      /* (0) Marker size scale factor */  
    Gint        *error        /* (0) Error indicator */  
)
```

### Description

The INQUIRE MARKER SIZE SCALE FACTOR function returns the value for the *current marker size scale factor* entry in the GKS state list for all polymarker types.

DEC GKS calculates marker size for all types (except the dot marker type) as the nominal marker size multiplied by the marker size scale factor. The marker size scale factor is a real number passed to the SET MARKER SIZE SCALE FACTOR function. The workstation maps the value to the nearest available marker size defined by the workstation. (The dot marker type is always the smallest dot that the workstation can produce.)

### See Also

SET MARKER SIZE SCALE FACTOR

## INQUIRE MARKER TYPE

---

### INQUIRE MARKER TYPE

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqmarkertype (  
    Gint          *markertype, /* (0) Polymarker type (constant) */  
    Gint          *error       /* (0) Error indicator */  
)
```

#### Constants

Defined Argument	Constant	Description
markertype	GMK_POINT	Dot
	GMK_PLUS	Plus sign
	GMK_STAR	Asterisk
	GMK_O	Circle
	GMK_X	Diagonal cross

#### Description

The INQUIRE MARKER TYPE function returns the value for the *current marker type* entry in the GKS state list as dots, plus signs, asterisks, circles, diagonal crosses, or any of the device-dependent types.

Every workstation capable of output (of DEC GKS category OUTPUT or OUTIN) defines at least five polymarker types. Additionally, there may be many more device-dependent types. For more information concerning the device-dependent polymarker types, see the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*. For a complete list of available polymarker types, see Appendix B.

#### See Also

SET MARKER TYPE



## INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

---

### INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqmaxwsstables (  
    Gwstype      *type,          /* (I) Workstation type */  
    Gwstables    *tables,       /* (O) Returned table sizes */  
    Gint         *error         /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* LENGTH OF WORKSTATION TABLES */  
    Gint    line;         /* polyline tables */  
    Gint    mark;        /* polymarker tables */  
    Gint    text;        /* text tables */  
    Gint    fill;        /* fill area tables */  
    Gint    pat;         /* pattern tables */  
    Gint    colour;     /* color tables */  
} Gwstables;
```

#### Description

The INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES function returns, for a specific workstation type, the maximum number of polyline bundles, polymarker bundles, text bundles, fill area bundles, pattern indexes, and color indexes table entries.

## INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3

---

### INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqmaxwsstables3 (  
    Gwstype      *type,          /* (I) Workstation type */  
    Gwstables3   *tables3,       /* (O) Returned table sizes */  
    Gint         *error          /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* LENGTH OF WORKSTATION TABLES 3 */  
    Gint    line;         /* polyline tables */  
    Gint    mark;        /* polymarker tables */  
    Gint    text;        /* text tables */  
    Gint    fill;        /* fill area tables */  
    Gint    pat;         /* pattern tables */  
    Gint    colour;      /* color tables */  
    Gint    view;        /* view tables */  
    Gint    edge;        /* edge tables */  
} Gwstables3;
```

#### Description

The INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3 function returns, for a specific workstation type, the maximum number of polyline bundles, polymarker bundles, text bundles, fill area bundles, pattern indexes, color indexes, view indexes, and edge indexes table entries.

# INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

---

## INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmaxntrannum (  
    Gint      *maxtran,    /* (0) Maximum normalization transformation  
                           number supported. Window and viewport  
                           boundaries can be associated with  
                           transformation numbers from 1 to  
                           maxtran. */  
    Gint      *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER function returns the maximum normalization transformation number supported by the GKS implementation being used. The maximum number for the DEC GKS software is 255. Normalization transformation number 0 is the unity transformation and cannot be changed.

### See Also

SELECT NORMALIZATION TRANSFORMATION  
SET VIEWPORT INPUT PRIORITY

## INQUIRE MORE SIMULTANEOUS EVENTS

---

### INQUIRE MORE SIMULTANEOUS EVENTS

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqmoreevents (  
    Gsimultev    *events,    /* (0) More simultaneous events flag (constant) */  
    Gint         *error      /* (0) Error indicator */  
)
```

#### Constants

Data Type	Constant	Description
Gsimultev	GNOMORE	No more events
	GMORE	More events

#### Description

The INQUIRE MORE SIMULTANEOUS EVENTS function queries the GKS state list to see if there are more events on the event input queue that were entered by the user firing a single trigger.

---

## INQUIRE NAME OF OPEN SEGMENT

### Operating State

SGOP

### Syntax

```
ginqnameopenseg (  
    Gint      *seg,          /* (0) Segment name */  
    Gint      *error         /* (0) Error indicator */  
)
```

### Description

The INQUIRE NAME OF OPEN SEGMENT function returns the identification number of the currently open segment.

### See Also

CREATE SEGMENT

# INQUIRE NORMALIZATION TRANSFORMATION

---

## INQUIRE NORMALIZATION TRANSFORMATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqtran (  
    Gint      xform,    /* (I) Normalization transformation number */  
    Gtran     *tran,    /* (O) WC window and NDC viewport limits returned */  
    Gint      *error    /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* TRANSFORMATION */  
    Glimit     w;        /* window */  
    Glimit     v;        /* viewport */  
} Gtran;  
  
typedef struct {          /* COORDINATE LIMITS */  
    Gfloat     xmin;     /* X minimum limit */  
    Gfloat     xmax;     /* X maximum limit */  
    Gfloat     ymin;     /* Y minimum limit */  
    Gfloat     ymax;     /* Y maximum limit */  
} Glimit;
```

### Description

The INQUIRE NORMALIZATION TRANSFORMATION function returns the boundaries of the normalization window and the normalization viewport associated with the specified normalization transformation number.

The maximum number of normalization transformations for the DEC GKS software is 255. Normalization transformation number 0 is the unity transformation and cannot be changed.

### See Also

SELECT NORMALIZATION TRANSFORMATION  
SET VIEWPORT  
SET WINDOW

---

## INQUIRE NORMALIZATION TRANSFORMATION 3

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqtran3 (
    Gint    xform,    /* (I) Normalization transformation number */
    Gtran3  *tran3,   /* (O) WC window and NDC viewport limits returned */
    Gint    *error    /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* NORMALIZATION TRANSFORMATION */
    Glimit3  w;          /* WC window limits */
    Glimit3  v;          /* NDC viewport limits */
} Gtran3;

typedef struct {          /* COORDINATE LIMITS */
    Gfloat   xmin;      /* X minimum limit */
    Gfloat   xmax;      /* X maximum limit */
    Gfloat   ymin;      /* Y minimum limit */
    Gfloat   ymax;      /* Y maximum limit */
    Gfloat   zmin;      /* Z minimum limit */
    Gfloat   zmax;      /* Z maximum limit */
} Glimit3;
```

### Description

The INQUIRE NORMALIZATION TRANSFORMATION 3 function returns the boundaries of the normalization window and the normalization viewport associated with the specified normalization transformation number.

The maximum number of normalization transformations for the DEC GKS software is 255. Normalization transformation number 0 is the unity transformation and cannot be changed.

### See Also

SELECT NORMALIZATION TRANSFORMATION  
 SET VIEWPORT 3  
 SET WINDOW 3

## INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

---

## INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqnumavailinput (  
    Gwstype    *type,      /* (I) Workstation type */  
    Gnumdev    *num,       /* (O) Number of devices */  
    Gint       *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* INPUT DEVICES */  
    Gint    locator;     /* locators */  
    Gint    stroke;      /* strokes */  
    Gint    valuator;    /* valuator */  
    Gint    choice;      /* choices */  
    Gint    pick;        /* picks */  
    Gint    string;      /* strings */  
} Gnumdev;
```

### Description

The INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES function returns the number of logical input devices in each class for the given workstation type.



## INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

---

### INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqnumsegpri (  
    Gwstype    *type,      /* (I) Workstation type. */  
    Gint       *numpri,    /* (O) Number of priorities. If this function  
                           writes the value 0 to this argument, the  
                           device supports an infinite number of  
                           priorities. */  
    Gint       *error      /* (O) Error indicator. */  
)
```

#### Description

The INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED function returns the number of segment priorities supported for the specified workstation type.

If the returned number of segment priorities supported is 0, the workstation supports an infinite number of segment priorities.

## INQUIRE OPERATING STATE VALUE

---

### INQUIRE OPERATING STATE VALUE

#### Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqpst (  
    Gopst    *state    /* (0) Operating state (constant) */  
)
```

#### Constant

Data Type	Constant	Description
Gopst	GGKCL	GKS is closed.
	GGKOP	GKS is open.
	GWSOP	At least one workstation is open.
	GWSAC	At least one workstation is active.
	GSGOP	A segment is being opened.

#### Description

The INQUIRE OPERATING STATE VALUE function returns the DEC GKS operating state.

#### See Also

OPEN GKS

---

## INQUIRE PATTERN FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpatfacil (  
    Gwstype    *type,          /* (I) Workstation type */  
    Gint       bufsize,       /* (I) Not used */  
    Gint       *fac_size,     /* (O) Not used */  
    Gint       *num_pat_indexes, /* (O) Number of predefined pattern indexes */  
    Gint       *error         /* (O) Error indicator */  
)
```

### Description

The INQUIRE PATTERN FACILITIES function returns the number of pattern indexes available for the specified workstation type.

## INQUIRE PATTERN HEIGHT VECTOR

---

### INQUIRE PATTERN HEIGHT VECTOR

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqpatheight (  
    Gpoint    *heightvec,    /* (0) Pattern height vector */  
    Gint      *error         /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat    x;      /* X coordinate */  
    Gfloat    y;      /* Y coordinate */  
} Gpoint;
```

#### Description

The INQUIRE PATTERN HEIGHT VECTOR function queries the GKS state list and returns the current text path height.

---

## INQUIRE PATTERN REFERENCE POINT

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpatrefpt (  
    Gpoint    *prp,        /* (0) Pattern reference point */  
    Gint      *error       /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat    x; /* X coordinate */  
    Gfloat    y; /* Y coordinate */  
} Gpoint;
```

### Description

The INQUIRE PATTERN REFERENCE POINT function returns the value for the geometric attribute *current pattern reference point* entry in the GKS state list.

This attribute represents the starting point for a pattern used to fill the designated area. DEC GKS uses this value for all subsequent calls to FILL AREA until another value is specified.

## INQUIRE PATTERN REFERENCE POINT AND VECTORS

---

## INQUIRE PATTERN REFERENCE POINT AND VECTORS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpatrefptvector (  
    Gpoint3      *ref_pt,      /* (0) Pattern reference point */  
    Gvector3     *dx,          /* (0) Pattern reference vector 1 */  
    Gvector3     *dy,          /* (0) Pattern reference vector 2 */  
    Gint         *error        /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat x; /* X coordinate */  
    Gfloat y; /* Y coordinate */  
    Gfloat z; /* Z coordinate */  
} Gpoint3;  
  
typedef struct { /* COORDINATE POINTS */  
    Gfloat x; /* X coordinate */  
    Gfloat y; /* Y coordinate */  
    Gfloat z; /* Z coordinate */  
} Gvector3;
```

### Description

The INQUIRE PATTERN REFERENCE POINT AND VECTORS function returns the value for the geometric attributes *current pattern reference point* and *current pattern reference vectors* entries in the GKS state list.

The point attribute represents the starting point for a pattern used to fill the designated area. DEC GKS uses this value for all subsequent calls to the FILL AREA function until another value is specified. When the currently selected fill area interior style is PATTERN, the vectors attribute is used in conjunction with the current pattern width and height vectors to display the FILL AREA and FILL AREA SET output primitives.

## INQUIRE PATTERN REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqpatrep (
    Gint    ws,          /* (I) Workstation identifier */
    Gint    index,      /* (I) Pattern index */
    Gintype type,       /* (I) Return type (constant) */
    Gptbundl *rep,      /* (O) Returned pattern representation */
    Gint    *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* PATTERN BUNDLE */
    Gipoint size;        /* pattern array size */
    Gint *array;         /* pattern array */
} Gptbundl;

typedef struct {         /* INTEGER POINT */
    Gint x;              /* X coordinate */
    Gint y;              /* Y coordinate */
} Gipoint;
```

### Constants

Data Type	Constant	Description
Gintype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.

### Description

The INQUIRE PATTERN REPRESENTATION function returns the values associated with the given pattern index value.

### See Also

SET PATTERN REPRESENTATION  
SET PATTERN SIZE

## INQUIRE PATTERN WIDTH VECTOR

---

### INQUIRE PATTERN WIDTH VECTOR

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqpatwidth (  
    Gpoint    *widthvec,    /* (0) Pattern width vector */  
    Gint      *error        /* (0) Error indicator */  
)
```

#### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat    x; /* X coordinate */  
    Gfloat    y; /* Y coordinate */  
} Gpoint;
```

#### Description

The INQUIRE PATTERN WIDTH VECTOR function queries the GKS state list and returns the current pattern width vector.



---

**INQUIRE PICK DEVICE STATE**
**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginpickst (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      dev,        /* (I) Pick device number */
    Gintqtype type,      /* (I) Return type (constant) */
    Gint      bufsize,   /* (I) Allocated record buffer size */
    Gint      *state_size, /* (O) Required record buffer size */
    Gpickst   *state,    /* (O) Returned data structure */
    Gint      *error      /* (O) Error indicator */
)

```

**Data Structures**

```
typedef struct {          /* PICK STATE */
    Gimode    mode;      /* mode (constant) */
    Gesw      esw;       /* echo switch (constant) */
    Gpick     pick;      /* pick data */
    Gint      pet;       /* prompt and echo type */
    Glimit    e_area;    /* echo area */
    Gpickrec  record;    /* pick data record */
} Gpickst;

    typedef struct {      /* PICK DATA */
        Gpstat    status; /* pick status (constant) */
        Gint      seg;    /* pick segment */
        Gint      pickid; /* pick identifier */
    } Gpick;

    typedef struct {      /* COORDINATE LIMITS */
        Gfloat    xmin;   /* X minimum limit */
        Gfloat    xmax;   /* X maximum limit */
        Gfloat    ymin;   /* Y minimum limit */
        Gfloat    ymax;   /* Y maximum limit */
    } Glimit;

    typedef union {       /* PICK DATA RECORD */
        Gpickpet0001    pickpet1_datarec;
        Gpickpet0002    pickpet2_datarec;
        Gpickpet0003    pickpet3_datarec;
    } Gpickrec;

    typedef struct {
        Gfloat    aperture; /* pick aperture in device coordinates */
        Gchar     *data;     /* device/implementation dependent data */
    } Gpickpet0001;

    typedef Gpickpet0001 Gpickpet0002;
    typedef Gpickpet0001 Gpickpet0003;

```

# INQUIRE PICK DEVICE STATE

## Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled.
	GNOECHO	Echo disabled.
Gpstat	GP_OK	Return the initial measure.
	GP_NOPICK	Return no pick.

## Description

The INQUIRE PICK DEVICE STATE function returns the current state of the given pick-class logical input device.

The information returned by this function depends on the value of the *bufsize*. If *bufsize* is equal to the `sizeof(Gpickrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the pick data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

## See Also

INITIALIZE PICK  
SET PICK MODE

Example 9-2 for a program example using the INQUIRE PICK DEVICE STATE function

---

**INQUIRE PICK DEVICE STATE 3**
**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginpickst3 (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      dev,        /* (I) Pick device number */
    Gintqtype type,      /* (I) Return type (constant) */
    Gint      bufsize,   /* (I) Allocated record buffer size */
    Gint      *state_size, /* (O) Required record buffer size */
    Gpickst3  *state,    /* (O) Returned data structure */
    Gint      *error      /* (O) Error indicator */
)

```

**Data Structures**

```
typedef struct {          /* PICK 3 STATE */
    Gimode    mode;      /* pick mode (constant) */
    Gesw      esw;       /* echo switch (constant) */
    Gpick     pick;      /* pick data */
    Gint      pet;       /* prompt and echo type */
    Glimit3   e_vol;     /* echo volume */
    Gpickrec  record;    /* pick data record */
} Gpickst3;

    typedef struct {      /* PICK DATA */
        Gpstat    status; /* pick status (constant) */
        Gint      seg;    /* pick segment */
        Gint      pickid; /* pick identifier */
    } Gpick;

    typedef struct {      /* COORDINATE LIMITS */
        Gfloat    xmin;   /* X minimum limit */
        Gfloat    xmax;   /* X maximum limit */
        Gfloat    ymin;   /* Y minimum limit */
        Gfloat    ymax;   /* Y maximum limit */
        Gfloat    zmin;   /* Z minimum limit */
        Gfloat    zmax;   /* Z maximum limit */
    } Glimit3;

    typedef union {       /* PICK DATA RECORD */
        Gpickpet001 pickpet1_datarec;
        Gpickpet002 pickpet2_datarec;
        Gpickpet003 pickpet3_datarec;
    } Gpickrec;

    typedef struct {
        Gfloat    aperture; /* pick aperture in device coordinates */
        Gchar     *data;    /* device/implementation dependent data */
    } Gpickpet0001;

    typedef Gpickpet0001 Gpickpet0002;
    typedef Gpickpet0001 Gpickpet0003;

```

## INQUIRE PICK DEVICE STATE 3

### Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled.
	GNOECHO	Echo disabled.
Gpstat	GP_OK	Return the initial measure.
	GP_NOPICK	Return no pick.

### Description

The INQUIRE PICK DEVICE STATE 3 function returns the current state of the given pick-class logical input device.

The information returned by this function depends on the value of the *bufsize*. If *bufsize* is equal to the `sizeof(Gpickrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the pick data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE PICK 3

SET PICK MODE

Example 9–2 for a program example using the INQUIRE PICK DEVICE STATE function

---

## INQUIRE PIXEL

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqpixel (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gpoint  *point,     /* (I) WC point */  
    Gint    *pix,       /* (O) Color index */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct { /* COORDINATE POINT */  
    Gfloat  x; /* X coordinate */  
    Gfloat  y; /* Y coordinate */  
} Gpoint;
```

### Description

The INQUIRE PIXEL function returns the color index of an individual pixel on the display surface.

The specified point is transformed by the current normalization and workstation transformations, and by using a view index of 0, to a pixel on the display surface. The color index associated with this pixel is returned. If the color index of the pixel cannot be ascertained, the value -1 is returned for the pixel.

## INQUIRE PIXEL ARRAY

---

## INQUIRE PIXEL ARRAY

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqpixelarray (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gpoint    *point,     /* (I) 2D point in WC */  
    Gidim     *dim,       /* (I) Dimensions of color index array */  
    Gint      bufsize,    /* (I) Requested size */  
    Gcovalid  *covalid,   /* (O) Flag for invalid color index values  
                          (constant)*/  
    Gintlist  *pxarray,   /* (O) Returned pixel array */  
    Gint      *actual,    /* (O) Actual length of array */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* COORDINATE POINT */  
    Gfloat    x;          /* X coordinate */  
    Gfloat    y;          /* Y coordinate */  
} Gpoint;  
  
typedef struct {          /* DIMENSION IN INTEGER VALUES */  
    Gint      x_dim;     /* x dimension */  
    Gint      y_dim;     /* y dimension */  
} Gidim;  
  
typedef struct {          /* INTEGER LIST */  
    Gint      number;    /* number of integers in list */  
    Gint      *integers; /* list of integers */  
} Gintlist;
```

### Constants

Data type	Constant	Description
Gcovalid	GABSENT	Color array contains no invalid indexes.
	GPRESSENT	Color array contains invalid indexes.

### Description

The INQUIRE PIXEL ARRAY function returns the color indexes of pixels in a rectangular region on the screen.

The specified point is transformed by the current normalization and workstation transformations, and by using a view index of 0, to a pixel on the display surface. The color indexes of the array of pixels whose upper left-hand corner is the transformed point are returned. If the color index of a particular pixel cannot be ascertained, the value -1 is returned for that pixel.

**See Also**

SET COLOUR REPRESENTATION

## INQUIRE PIXEL ARRAY DIMENSIONS

---

## INQUIRE PIXEL ARRAY DIMENSIONS

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqpixelarraydim (  
    Gint    ws,          /* (I) Workstation identifier */  
    Grect   *rect,      /* (I) WC of parallelogram */  
    Gidim   *dim,       /* (O) Returned dimensions of structure */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* COORDINATE RECTANGLE */  
    Gpoint  ul;          /* upper left corner */  
    Gpoint  lr;          /* lower right corner */  
} Grect;  
  
    typedef struct {     /* COORDINATE POINT */  
        Gfloat  x;       /* X coordinate */  
        Gfloat  y;       /* Y coordinate */  
    } Gpoint;  
  
typedef struct {         /* DIMENSION IN INTEGER VALUES */  
    Gint    x_dim;      /* X dimension */  
    Gint    y_dim;      /* Y dimension */  
} Gidim;
```

### Description

The INQUIRE PIXEL ARRAY DIMENSIONS function returns the number of pixels in the X and Y axis of a rectangular portion of the display surface.

The two specified points determine the rectangular portion of the display surface. By transforming the two specified points by the current normalization and workstation transformations, and by using a view index of 0, a rectangle is mapped onto the display surface. No clipping is applied. The number of columns and rows of pixels whose positions lie within the rectangle is returned.



---

**INQUIRE POLYLINE COLOUR INDEX****Operating States**

GKOP, WSOP, WSAC, SGOP

**Syntax**

```
ginqlinecolourind (  
    Gint    *index,      /* (0) Line color index */  
    Gint    *error       /* (0) Error indicator */  
)
```

**Description**

The INQUIRE POLYLINE COLOUR INDEX function returns the value for the *current polyline color index* entry in the GKS state list.

**See Also**

SET POLYLINE COLOUR INDEX  
SET COLOUR REPRESENTATION

## INQUIRE POLYLINE FACILITIES

---

## INQUIRE POLYLINE FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqlinefacil (  
    Gwstype  *type,          /* (I) Workstation type */  
    Gint     bufsize,       /* (I) Size of allocated buffer */  
    Gint     *num_lt,       /* (O) Number of available line types */  
    Glnfac   *fac,          /* (O) Returned line facilities */  
    Gint     *error         /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* POLYLINE FACILITIES */  
    Gintlist  *types;     /* list of available line types (constant) */  
    Gint      widths;     /* number of line widths */  
    Gfloat    nom_width;  /* nominal width */  
    Gfloat    min_width;  /* minimum width */  
    Gfloat    max_width;  /* maximum width */  
    Gint      predefined; /* number of predefined bundles */  
} Glnfac;  
  
typedef struct {          /* INTEGER LIST */  
    Gint      number;     /* number of integers in list */  
    Gint      *integers;  /* list of integers */  
} Gintlist;
```

### Constants

Data Type	Constant	Description
Line types	GLN_SOLID	Solid line
	GLN_DASHED	Dashed line
	GLN_DOTTED	Dotted line
	GLN_DASHDOT	Dashed-dotted line

### Description

The INQUIRE POLYLINE FACILITIES function queries the workstation description table and returns the number of line types, a list of line types, the number of line widths, the nominal, minimum, and maximum line widths, and the number of predefined polyline index values.

If the number of line widths returned is 0, the workstation supports a continuous range of line widths.

---

## INQUIRE POLYLINE INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqlineind (  
    Gint    *index,      /* (0) Line index */  
    Gint    *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE POLYLINE INDEX function returns the value for the *current polyline index* entry in the GKS state list.

The polyline bundle table contains entries for the polyline color index, polyline type, and polyline width scale factor attribute values. When calling POLYLINE, DEC GKS uses the bundle table only if the corresponding ASF has been set to BUNDLED.

### See Also

SET POLYLINE INDEX

# INQUIRE POLYLINE REPRESENTATION

---

## INQUIRE POLYLINE REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqlinerep (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      index,      /* (I) Polyline index */  
    Ginttype  type,       /* (I) Return type (constant) */  
    Glnbundl  *rep,       /* (O) Returned polyline representation */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* POLYLINE BUNDLE */  
    Gint      type;       /* line type (constant) */  
    Gfloat    width;      /* line width scale factor */  
    Gint      colour;     /* polyline color index */  
} Glnbundl;
```

### Constants

Data Type	Constant	Description
Ginttype	GSET	Use the exact state list values.
	REALIZED	Use the values approximated by the graphics handler.
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.

### Description

The INQUIRE POLYLINE REPRESENTATION function returns the values associated with the given polyline index value.

If the specified polyline index is not in the polyline bundle table on the specified workstation, and the specified type of returned values is REALIZED, the representation for polyline index 1 is returned.

### See Also

SET POLYLINE REPRESENTATION

---

## INQUIRE POLYMARKER COLOUR INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmarkercolourind (  
    Gint    *index,      /* (0) Marker color index */  
    Gint    *error       /* (0) Error indicator */  
)
```

### Description

The INQUIRE POLYMARKER COLOUR INDEX function queries the GKS state list and returns the value for the *current polymarker color index* entry.

All output polymarkers are shown in the color corresponding to the index value, provided the ASF corresponding to the polymarker color is set to INDIVIDUAL.

### See Also

SET COLOUR REPRESENTATION  
SET POLYMARKER COLOUR INDEX

# INQUIRE POLYMARKER FACILITIES

---

## INQUIRE POLYMARKER FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmarkerfacil (  
    Gwstype      *type,          /* (I) Workstation type */  
    Gint         bufsize,       /* (I) Size of allocated buffer */  
    Gint         *num_mt,       /* (O) Number of available polymarker types */  
    Gmkfac       *fac,          /* (O) Returned marker facilities */  
    Gint         *error         /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* POLYMARKER FACILITIES */  
    Gintlist  *types;     /* list of available marker types (constant) */  
    Gint      sizes;     /* number of marker sizes */  
    Gfloat    nom_size;  /* nominal size */  
    Gfloat    min_size;  /* minimum size */  
    Gfloat    max_size;  /* maximum size */  
    Gint      predefined; /* number of predefined bundles */  
} Gmkfac;  
  
typedef struct {          /* INTEGER LIST */  
    Gint      number;    /* number of integers in list */  
    Gint      *integers; /* list of integers */  
} Gintlist;
```

### Constants

Data Type	Constant	Description
Marker types	GMK_POINT	Dot
	GMK_PLUS	Plus sign
	GMK_STAR	Asterisk
	GMK_O	Circle
	GMK_X	Diagonal cross

### Description

The INQUIRE POLYMARKER FACILITIES function returns the number of marker types; a list of marker types; the nominal, minimum, and maximum marker sizes; and the number of predefined polymarker index values.

If the returned marker size is 0, the workstation supports a continuous range of marker sizes.

---

## INQUIRE POLYMARKER INDEX

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqmarkerind (  
    Gint    *index,    /* (0) Marker index */  
    Gint    *error     /* (0) Error indicator */  
)
```

### Description

The INQUIRE POLYMARKER INDEX function returns the value for the *current polymarker index* entry in the GKS state list.

The polymarker bundle table contains entries for the polymarker color index, polymarker type, and polymarker size scale factor attribute values. When calling POLYMARKER, DEC GKS uses the bundle table only if the corresponding ASF has been set to BUNDLED.

# INQUIRE POLYMARKER REPRESENTATION

---

## INQUIRE POLYMARKER REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqmarkerrep (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      index,       /* (I) Polymarker index */  
    Ginttype  type,        /* (I) Return type (constant) */  
    Gmkbundl  *rep,        /* (O) Polymarker representation */  
    Gint      *error       /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* POLYMARKER BUNDLE */  
    Gint      type;       /* marker type (constant) */  
    Gfloat    size;       /* marker size scale factor */  
    Gint      colour;     /* polymarker color index */  
} Gmkbundl;
```

### Constants

Data Type	Constant	Description
Ginttype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Marker types	GMK_POINT	Dot.
	GMK_PLUS	Plus sign.
	GMK_STAR	Asterisk.
	GMK_O	Circle.
	GMK_X	Diagonal cross.

### Description

The INQUIRE POLYMARKER REPRESENTATION function returns the values associated with the given polymarker index value.

If the specified polymarker index is not in the polymarker bundle table on the specified workstation, and the specified type of returned values is REALIZED, the representation for polymarker index 1 is returned.

### See Also

SET POLYMARKER REPRESENTATION



---

## INQUIRE PREDEFINED COLOUR REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpredcolourep (  
    Gwstype      *type,      /* (I) Workstation identifier */  
    Gint         index,      /* (I) Color index */  
    Gcobundl     *rep,       /* (O) Returned color representation */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {      /* COLOR BUNDLE */  
    Gfloat red;       /* red intensity */  
    Gfloat green;     /* green intensity */  
    Gfloat blue;      /* blue intensity */  
} Gcobundl;
```

### Description

The INQUIRE PREDEFINED COLOUR REPRESENTATION function returns the predefined color component values associated with the color model in the workstation state list for the specified color index.

### See Also

INQUIRE COLOUR MODEL FACILITIES

# INQUIRE PREDEFINED EDGE REPRESENTATION

---

## INQUIRE PREDEFINED EDGE REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqprededgerep (  
    Gwstype      *type,      /* (I) Workstation identifier */  
    Gint         index,      /* (I) Edge index */  
    Gedgebundl   *rep,       /* (O) Returned edge representation */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* EDGE BUNDLE */  
    Gedge_f  flag;        /* edge flag (constant) */  
    Gint     type;        /* edge type (constant) */  
    Gfloat   width;       /* edge width scale factor */  
    Gint     colour;      /* edge color index */  
} Gedgebundl;
```

### Constants

Data Type	Constant	Description
Gedge_f	GEDGE_OFF	Edge off
	GEDGE_ON	Edge on
Edge types	GED_SOLID	Solid edge
	GED_DASHED	Dashed edge
	GED_DOTTED	Dotted edge
	GED_DASHDOT	Dashed-dotted edge

### Description

The INQUIRE PREDEFINED EDGE REPRESENTATION function returns the values for the predefined edge flag, edge type, edge width scale factor, and edge color index associated with the specified edge index.

### See Also

SET COLOUR REPRESENTATION

---

## INQUIRE PREDEFINED FILL AREA REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpredfillrep (  
    Gwstype      *type,      /* (I) Workstation identifier */  
    Gint         index,      /* (I) Fill area index */  
    Gflbundl     *rep,       /* (O) Returned fill area representation */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {      /* FILL AREA BUNDLE */  
    Gflinter inter;   /* fill area interior style (constant) */  
    Gint      style;   /* fill area style index */  
    Gint      colour;  /* fill area color index */  
} Gflbundl;
```

### Constants

Data Type	Constant	Description
Gflinter	GHOLLOW	Hollow interior
	GSOLID	Solid interior
	GPATTERN	Patterned interior
	GHATCH	Hatched interior

### Description

The INQUIRE PREDEFINED FILL AREA REPRESENTATION function returns the predefined values for the interior style, style index, and fill area color index associated with a specific fill area index for a given workstation type.

### See Also

SET COLOUR REPRESENTATION  
SET FILL AREA STYLE INDEX

## INQUIRE PREDEFINED PATTERN REPRESENTATION

---

### INQUIRE PREDEFINED PATTERN REPRESENTATION

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqpredpatrep (  
    Gwstype    *type,    /* (I) Workstation identifier */  
    Gint       index,    /* (I) Pattern index */  
    Gptbundl   *rep,     /* (O) Returned pattern representation */  
    Gint       *error    /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* PATTERN BUNDLE */  
    Gipoint  size;        /* pattern array size */  
    Gint     *array;      /* pattern array */  
} Gptbundl;  
  
typedef struct {         /* INTEGER POINT */  
    Gint     x;          /* X coordinate */  
    Gint     y;          /* Y coordinate */  
} Gipoint;
```

#### Description

The INQUIRE PREDEFINED PATTERN REPRESENTATION function returns a description of the specific pattern by returning the pattern dimensions and the array of color indexes that comprises the pattern.

## INQUIRE PREDEFINED POLYLINE REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```

ginqpredlinerep (
    Gwstype    *type,    /* (I) Workstation identifier */
    Gint       index,    /* (I) Polyline index */
    Glnbundl   *rep,     /* (O) Returned polyline representation */
    Gint       *error    /* (O) Error indicator */
)
    
```

### Data Structures

```

typedef struct {          /* POLYLINE BUNDLE */
    Gint    type;        /* line type (constant) */
    Gfloat  width;       /* line width scale factor */
    Gint    colour;     /* polyline color index */
} Glnbundl;
    
```

### Constants

Data Type	Constant	Description
Line types	GLN_SOLID	Solid line
	GLN_DASHED	Dashed line
	GLN_DOTTED	Dotted line
	GLN_DASHDOT	Dashed-dotted line

### Description

The INQUIRE PREDEFINED POLYLINE REPRESENTATION function returns the predefined values for the line type, color index, and line width associated with a specific polyline index for a given workstation type.

# INQUIRE PREDEFINED POLYMARKER REPRESENTATION

---

## INQUIRE PREDEFINED POLYMARKER REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpredmarkerrep (  
    Gwstype    *type,    /* (I) Workstation identifier */  
    Gint       index,    /* (I) Polymarker index */  
    Gmkbundl   *rep,     /* (O) Polymarker representation */  
    Gint       *error    /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* POLYMARKER BUNDLE */  
    Gint    type;         /* marker type (constant) */  
    Gfloat  size;        /* marker size scale factor */  
    Gint    colour;      /* polymarker color index */  
} Gmkbundl;
```

### Constants

Data Type	Constant	Description
Marker types	GMK_POINT	Dot
	GMK_PLUS	Plus sign
	GMK_STAR	Asterisk
	GMK_O	Circle
	GMK_X	Diagonal cross

### Description

The INQUIRE PREDEFINED POLYMARKER REPRESENTATION function returns the predefined values for the marker type, color index, and marker size scale factor associated with a specific polymarker index for a given workstation type.

## INQUIRE PREDEFINED TEXT REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```

ginqpredtextrep (
    Gwstype    *type,      /* (I) Workstation identifier */
    Gint       index,     /* (I) Text index */
    Gtxbundl   *rep,      /* (O) Returned text representation */
    Gint       *error     /* (O) Error indicator */
)
    
```

### Data Structures

```

typedef struct {          /* TEXT BUNDLE */
    Gtxfp    fp;         /* font and precision */
    Gfloat   exp;        /* character expansion */
    Gfloat   space;      /* character spacing */
    Gint     colour;     /* text color */
} Gtxbundl;

typedef struct {         /* TEXT FONT AND PRECISION */
    Gint     font;       /* text font */
    Gtxprec  prec;      /* text precision (constant) */
} Gtxfp;
    
```

### Constants

Data Type	Constant	Description
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.

### Description

The INQUIRE PREDEFINED TEXT REPRESENTATION function returns the predefined values for the text font and precision, character expansion factor, character spacing, and text color index associated with the specific text index for a given workstation type.

# INQUIRE PREDEFINED VIEW REPRESENTATION

---

## INQUIRE PREDEFINED VIEW REPRESENTATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqpredviewrep (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gint         index,     /* (I) Predefined view index value */  
    Gviewbundl  *rep,       /* (O) Returned view data structure */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* VIEW REPRESENTATION BUNDLE */  
    Gfloat   vo_matrix[4][4]; /* view orientation matrix */  
    Gfloat   vm_matrix[4][4]; /* view mapping matrix */  
    Glimit3  vc_limits;      /* clipping limits */  
    Gclip    req_xy_clip;    /* X-Y clipping indicator (constant) */  
    Gclip    req_b_clip;    /* back clip indicator (constant) */  
    Gclip    req_f_clip;    /* front clip indicator (constant) */  
} Gviewbundl;  
  
typedef struct {        /* COORDINATE LIMITS */  
    Gfloat   xmin;        /* X minimum limit */  
    Gfloat   xmax;        /* X maximum limit */  
    Gfloat   ymin;        /* Y minimum limit */  
    Gfloat   ymax;        /* Y maximum limit */  
    Gfloat   zmin;        /* Z minimum limit */  
    Gfloat   zmax;        /* Z maximum limit */  
} Glimit3;
```

### Constants

Data Type	Constant	Description
Gclip	GNOCLIP	Clipping disabled
	GCLIP	Clipping enabled

### Description

The INQUIRE PREDEFINED VIEW REPRESENTATION function returns the predefined values for the view orientation matrix, view mapping matrix, view clipping limits in NPC points, XY clipping indicator, back clipping indicator, and the front clipping indicator associated with a specific view index for a given workstation type.



## INQUIRE SEGMENT ATTRIBUTES

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqsegattr (
    Gint      seg,          /* (I) Segment name */
    Gsegattr *segattr,    /* (O) Segment attributes */
    Gint      *error       /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* SEGMENT ATTRIBUTES VALUES */
    Gfloat  segtran[2][3]; /* segment transformation matrix */
    Gsegvis segvis;       /* segment visibility (constant) */
    Gseghi  seghi;       /* segment highlighting (constant) */
    Gfloat  segpri;      /* segment priority */
    Gsegdet segdet;      /* segment detectability (constant) */
} Gsegattr;
```

### Constants

Data Type	Constant	Description
Gsegvis	GVISIBLE	DEC GKS shows the segment on the workstation surface. This is the default value.
	GINVISIBLE	DEC GKS does not show the segment on the workstation surface.
Gseghi	GNORMAL	DEC GKS does not highlight the segment. This is the default value.
	GHIGHLIGHTED	DEC GKS highlights the segment, if visible.
Gsegdet	GUNDETECTABLE	Segment cannot be picked. This is the default value.
	GDETECTABLE	Segment, if visible, can be picked.

### Description

The INQUIRE SEGMENT ATTRIBUTES function returns the current attribute values of the specified segment: the two-dimensional segment transformation matrix, visibility, highlighting, priority, and detectability.

## INQUIRE SEGMENT ATTRIBUTES

### See Also

ACCUMULATE TRANSFORMATION MATRIX  
CREATE SEGMENT  
EVALUATE TRANSFORMATION MATRIX  
SET DETECTABILITY  
SET HIGHLIGHTING  
SET SEGMENT PRIORITY  
SET SEGMENT TRANSFORMATION  
SET VISIBILITY

## INQUIRE SEGMENT ATTRIBUTES 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqsegattr3 (
    Gint      seg,          /* (I) Segment name */
    Gsegattr3 *segattr3,   /* (O) Segment attributes */
    Gint      *error       /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {           /* SEGMENT 3 ATTRIBUTES */
    Gfloat  segtran[3][4]; /* segment transformation matrix */
    Gsegvis segvis;       /* segment visibility (constant) */
    Gseghi  seghi;        /* segment highlighting (constant) */
    Gfloat  segpri;       /* segment priority */
    Gsegdet segdet;       /* segment detectability (constant) */
} Gsegattr3;
```

### Constants

Data Type	Constant	Description
Gsegvis	GVISIBLE	DEC GKS shows the segment on the workstation surface. This is the default value.
	GINVISIBLE	DEC GKS does not show the segment on the workstation surface.
Gseghi	GNORMAL	DEC GKS does not highlight the segment. This is the default value.
	GHIGHLIGHTED	DEC GKS highlights the segment, if visible.
Gsegdet	GUNDETECTABLE	Segment cannot be picked. This is the default value.
	GDETECTABLE	Segment, if visible, can be picked.

### Description

The INQUIRE SEGMENT ATTRIBUTES 3 function returns the current attribute values of the specified segment: the three-dimensional segment transformation matrix, visibility, highlighting, priority, and detectability.

## INQUIRE SEGMENT ATTRIBUTES 3

### See Also

ACCUMULATE TRANSFORMATION MATRIX 3  
CREATE SEGMENT  
EVALUATE TRANSFORMATION MATRIX 3  
SET DETECTABILITY  
SET HIGHLIGHTING  
SET SEGMENT PRIORITY  
SET SEGMENT TRANSFORMATION 3  
SET VISIBILITY

---

## INQUIRE SET OF ACTIVE WORKSTATIONS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqactivews (  
    Gint      max_ids,      /* (I) Maximum number of workstations that fit  
                           in buffer. */  
    Gint      start,       /* (I) Starting position of inquiry. Set it to 0  
                           to inquire from the beginning of the list. */  
    Gintlist  *wsids,      /* (O) List of active workstations. */  
    Gint      *actual_ids, /* (O) Number of workstations actually active. */  
    Gint      *error       /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint      number;    /* number of integers in list */  
    Gint      *integers; /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE SET OF ACTIVE WORKSTATIONS function returns the number and the list of active workstations.

### See Also

ACTIVATE WORKSTATION

## INQUIRE SET OF ASSOCIATED WORKSTATIONS

---

## INQUIRE SET OF ASSOCIATED WORKSTATIONS

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqassocws (  
    Gint    seg,          /* (I) Segment identifier. */  
    Gint    max_size,    /* (I) Number of workstation identifiers that  
                        buffer can hold. */  
    Gint    start,       /* (I) Starting position of the inquiry. Set it  
                        to 0 to inquire from the start of the list. */  
    Gint    *actual_size, /* (O) Total number of associated workstations. */  
    Gintlist *assocws,    /* (O) List of associated workstations. */  
    Gint    *error       /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gintlist *integers;  /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE SET OF ASSOCIATED WORKSTATIONS function returns the number and list of workstations associated with the specified segment.

### See Also

ASSOCIATE SEGMENT WITH WORKSTATION  
CREATE SEGMENT

---

## INQUIRE SET OF OPEN WORKSTATIONS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqopenws (  
    Gint      max_ids,      /* (I) Maximum number of workstations that fit in  
                           buffer. */  
    Gint      start,       /* (I) Starting position of inquiry. Set it to 0  
                           to inquire from the beginning of the list. */  
    Gintlist  *wsids,      /* (O) List of open workstations. */  
    Gint      *actual_ids, /* (O) Number of workstations actually open. */  
    Gint      *error       /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint      number;     /* number of integers in list */  
    Gint      *integers;  /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE SET OF OPEN WORKSTATIONS function returns the number and the list of open workstations.

### See Also

OPEN WORKSTATION

## INQUIRE SET OF SEGMENT NAMES IN USE

---

## INQUIRE SET OF SEGMENT NAMES IN USE

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqsegnames (  
    Gint      max_segnames,    /* (I) Maximum number of names buffer can  
                               hold. */  
    Gint      start,          /* (I) Starting point of inquiry. Set it to  
                               0 to inquire from the beginning of the  
                               list. */  
    Gintlist  *segnames,      /* (O) List of segment names. */  
    Gint      *actual_segnames, /* (O) Total number of segment names in  
                               use. */  
    Gint      *error          /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {                /* INTEGER LIST */  
    Gint      number;          /* number of integers in list */  
    Gint      *integers;      /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE SET OF SEGMENT NAMES IN USE function returns the number and the names of all existing segments.

### See Also

CREATE SEGMENT



# INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

---

## INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqsegnamesws (  
    Gint    ws,                /* (I) Workstation identifier. */  
    Gint    max_segnames,     /* (I) Maximum number of names buffer can  
                               hold. */  
    Gint    start,           /* (I) Starting point of inquiry. Set it to  
                               0 to inquire from the beginning of the  
                               list. */  
    Gintlist *segnames,      /* (O) List of segment names. */  
    Gint    *actual_segnames, /* (O) Total number of segment names in  
                               use. */  
    Gint    *error           /* (O) Error indicator. */  
)
```

### Data Structures

```
typedef struct {          /* INTEGER LIST */  
    Gint    number;      /* number of integers in list */  
    Gint    *integers;   /* list of integers */  
} Gintlist;
```

### Description

The INQUIRE SET OF SEGMENT NAMES ON WORKSTATION function returns the number and list of segment names stored on the specified workstation.

### See Also

CREATE SEGMENT  
RENAME SEGMENT

## INQUIRE STRING DEVICE STATE

---

### INQUIRE STRING DEVICE STATE

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqstringst (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      dev,        /* (I) Locator device number */  
    Gint      bufsize,    /* (I) String data record size */  
    Gint      *state_size, /* (O) Required string data record size */  
    Gstringst *state,     /* (O) Returned data structure */  
    Gint      *error      /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {          /* STRING STATE */  
    Gimode     mode;      /* mode (constant) */  
    Gesw       esw;       /* echo switch (constant) */  
    Gchar      *string;   /* string data */  
    Gint       pet;       /* prompt and echo type */  
    Glimit     e_area;    /* echo area */  
    Gstringrec record;    /* string data record */  
} Gstringst;  
  
typedef struct {         /* COORDINATE LIMITS */  
    Gfloat     xmin;      /* X minimum limit */  
    Gfloat     xmax;      /* X maximum limit */  
    Gfloat     ymin;      /* Y minimum limit */  
    Gfloat     ymax;      /* Y maximum limit */  
} Glimit;  
  
typedef union {          /* STRING DATA RECORD */  
    Gstringpet0001  
} Gstringrec;  
  
typedef struct {  
    Gint       bufsiz;    /* buffer size */  
    Gint       position;  /* initial cursor position */  
    Gchar      *title_string; /* title string */  
} Gstringpet0001;
```

---

#### Note

---

The string data buffer (*state.string*) must be initialized to point to a buffer with a length of 256 characters. The device-dependent data record will be initialized by the GKS system.

---

## Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled

## Description

The INQUIRE STRING DEVICE STATE function returns the current state of the given string-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gstringrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the string data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

## See Also

INITIALIZE STRING

SET STRING MODE

Example 9-3 for a program example using the INQUIRE STRING DEVICE STATE function

## INQUIRE STRING DEVICE STATE 3

---

## INQUIRE STRING DEVICE STATE 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqstringst3 (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      dev,        /* (I) Locator device number */  
    Gint      bufsize,    /* (I) String data record size */  
    Gint      *state_size, /* (O) Required string data record size */  
    Gstringst3 *state,    /* (O) Returned data structure */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* STRING 3 STATE */  
    Gimode     mode;      /* string mode (constant) */  
    Gesw       esw;      /* echo switch (constant) */  
    Gchar      *string;   /* string data */  
    Gint       pet;       /* prompt and echo type */  
    Glimit3    e_vol;     /* echo volume */  
    Gstringrec record;    /* string data record */  
} Gstringst3;  
  
typedef struct {          /* COORDINATE LIMITS */  
    Gfloat     xmin;     /* X minimum limit */  
    Gfloat     xmax;     /* X maximum limit */  
    Gfloat     ymin;     /* Y minimum limit */  
    Gfloat     ymax;     /* Y maximum limit */  
    Gfloat     zmin;     /* Z minimum limit */  
    Gfloat     zmax;     /* Z maximum limit */  
} Glimit3;  
  
typedef union {          /* STRING DATA RECORD */  
    Gstringpet0001 stringpet1_datarec;  
} Gstringrec;  
  
typedef struct {  
    Gint      bufsiz;     /* buffer size */  
    Gint      position;   /* initial cursor position */  
    Gchar      *title_string; /* title string */  
} Gstringpet0001;
```

---

#### Note

The string data buffer (*state.string*) must be initialized to point to a buffer with a length of 256 characters. The device-dependent data record will be initialized by the GKS system.

---

## Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled

## Description

The INQUIRE STRING DEVICE STATE 3 function returns the current state of the given string-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gstringrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the string data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

## See Also

INITIALIZE STRING 3

SET STRING MODE

Example 9-3 for a program example using the INQUIRE STRING DEVICE STATE function

# INQUIRE STROKE DEVICE STATE

---

## INQUIRE STROKE DEVICE STATE

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
gingstrokest (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      dev,        /* (I) Stroke device number */  
    Ginqtype  type,       /* (I) Return type (constant) */  
    Gint      bufsize,    /* (I) Allocated record buffer size */  
    Gint      *state_size, /* (O) Required record buffer size */  
    Gstrokest *state,     /* (O) Returned data structure */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* STROKE STATE */  
    Gimode    mode;      /* mode (constant) */  
    Gesw      esw;      /* echo switch (constant) */  
    Gstroke   stroke;    /* stroke data */  
    Gint      pet;       /* prompt and echo type */  
    Glimit    e_area;    /* echo area */  
    Gstrokest record;    /* stroke data record */  
} Gstrokest;  
  
    typedef struct {      /* STROKE DATA */  
        Gint      transform; /* normalization transformation number */  
        Gint      n_points;  /* number of points in stroke */  
        Gpoint    *points;   /* points in stroke */  
    } Gstroke;  
  
        typedef struct { /* COORDINATE POINT */  
            Gfloat    x;    /* X coordinate */  
            Gfloat    y;    /* Y coordinate */  
        } Gpoint;  
  
        typedef struct { /* COORDINATE LIMITS */  
            Gfloat    xmin; /* X minimum limit */  
            Gfloat    xmax; /* X maximum limit */  
            Gfloat    ymin; /* Y minimum limit */  
            Gfloat    ymax; /* Y maximum limit */  
        } Glimit;  
  
        typedef union { /* STROKE DATA RECORD */  
            Gstrokepet0001 strokepet1_datarec;  
            Gstrokepet0002 strokepet2_datarec;  
            Gstrokepet0003 strokepet3_datarec;  
            Gstrokepet0004 strokepet4_datarec;  
        } Gstrokestrec;  
  
        typedef struct {  
            Gint      bufsiz; /* input buffer size */  
            Gint      editpos; /* editing position */  
            Gpoint    interval; /* X, Y interval */  
            Gfloat    time; /* time interval */  
            Gchar     *data; /* device/implementation dependent data */  
        } Gstrokepet0001;
```

## INQUIRE STROKE DEVICE STATE

```

typedef Gstrokepet0001 Gstrokepet0002;

typedef struct {
    Gint      bufsiz;      /* input buffer size */
    Gint      editpos;     /* editing position */
    Gpoint    interval;   /* X, Y interval */
    Gfloat    time;       /* time interval */
    Gacf      acf;        /* attribute control flag (constant) */
    Gmkatrr  mk;          /* marker attributes */
    Gchar     *data;      /* device/implementation dependent data */
} Gstrokepet0003;

    typedef struct {      /* POLYMARKER ATTRIBUTES */
        Gasf      type;    /* marker type ASF (constant) */
        Gasf      size;    /* marker width ASF (constant) */
        Gasf      colour;  /* marker color ASF (constant) */
        Gint      mark;    /* marker index */
        Gmkbundl  bundl;   /* marker bundle */
    } Gmkatrr

        typedef struct {  /* POLYMARKER BUNDLE */
            Gint      type; /* marker type (constant) */
            Gfloat    size; /* marker size scale factor */
            Gint      colour; /* polyline color index */
        } Gmkbundl;

typedef struct {
    Gint      bufsiz;      /* input buffer size */
    Gint      editpos;     /* editing position */
    Gpoint    interval;   /* X, Y interval */
    Gfloat    time;       /* time interval */
    Gacf      acf;        /* attribute control flag */
    Glnattr  ln;          /* line attributes */
    Gchar     *data;      /* device/implementation dependent data */
} Gstrokepet0004;

    typedef struct {      /* POLYLINE ATTRIBUTES */
        Gasf      type;    /* line type ASF */
        Gasf      width;   /* line width ASF */
        Gasf      colour;  /* line color ASF */
        Gint      line;    /* line index */
        Glnbundl  bundl;   /* line bundle */
    } Glnattr;

        typedef struct {  /* POLYLINE BUNDLE */
            Gint      type; /* line type (constant) */
            Gfloat    width; /* line width scale factor */
            Gint      colour; /* polyline color index */
        } Glnbundl;

```

### Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled.
	GNOECHO	Echo disabled.

## INQUIRE STROKE DEVICE STATE

Gacf	GCURRENT GSPECIFIED	Input data record current values. Input data record specified values.
Gasf	GBUNDLED GINDIVIDUAL	Bundled attributes. Individual attributes.
Marker types	GMK_POINT GMK_PLUS GMK_STAR GMK_O GMK_X	Dot. Plus sign. Asterisk. Circle. Diagonal cross.
Line types	GLN_SOLID GLN_DASHED GLN_DOTTED GLN_DASHDOT	Solid line. Dashed line. Dotted line. Dashed-dotted line.

### Description

The INQUIRE STROKE DEVICE STATE function returns the current state of the given stroke-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gstroke)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the stroke data record.

Before calling this function, you must initialize the following data structure elements:

- The *n\_points* field of structure `Gstroke` (`state→stroke.n_points`) must contain the maximum number of points that can be returned. The number of points should be large enough to hold all the entries. On output, this element contains the number of points returned.
- The *\*points* field of structure `Gstroke` (`state→stroke.points`) must point to an array of type `Gpoint` with *n\_points* elements.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE STROKE  
SET STROKE MODE

Example 9–3 for a program example using an INQUIRE . . . DEVICE STATE function



---

**INQUIRE STROKE DEVICE STATE 3**
**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```

gingstrokest3 (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      dev,        /* (I) Stroke device number */
    Ginttype  type,       /* (I) Return type (constant) */
    Gint      bufsize,    /* (I) Allocated record buffer size */
    Gint      *state_size, /* (O) Required record buffer size */
    Gstrokest3 *state,    /* (O) Returned data structure */
    Gint      *error      /* (O) Error indicator */
)

```

**Data Structures**

```

typedef struct {          /* STROKE 3 STATE */
    Gimode     mode;      /* locator mode (constant) */
    Gesw       esw;       /* echo switch (constant) */
    Gstroke3   stroke;    /* stroke data */
    Gint       pet;       /* prompt and echo type */
    Glimit3    e_vol;     /* echo volume */
    Gstroke3   record;    /* stroke data record */
} Gstrokest3;

    typedef struct {      /* STROKE 3 DATA */
        Gint      transform; /* normalization transformation number */
        Gint      view;     /* view index used in transformation */
        Gint      n_points;  /* number of points in stroke */
        Gpoint3   *points;   /* points in stroke */
    } Gstroke3;

        typedef struct {  /* COORDINATE POINT */
            Gfloat     x;   /* X coordinate */
            Gfloat     y;   /* Y coordinate */
            Gfloat     z;   /* Z coordinate */
        } Gpoint3;

            typedef struct { /* COORDINATE LIMITS */
                Gfloat     xmin; /* X minimum limit */
                Gfloat     xmax; /* X maximum limit */
                Gfloat     ymin; /* Y minimum limit */
                Gfloat     ymax; /* Y maximum limit */
                Gfloat     zmin; /* Z minimum limit */
                Gfloat     zmax; /* Z maximum limit */
            } Glimit3;

                typedef union { /* STROKE DATA RECORD */
                    Gstrokepet0001 strokepet1_datarec;
                    Gstrokepet0002 strokepet2_datarec;
                    Gstrokepet0003 strokepet3_datarec;
                    Gstrokepet0004 strokepet4_datarec;
                } Gstroke3rec;

```

## INQUIRE STROKE DEVICE STATE 3

```
typedef struct {
    Gint    bufsiz;    /* input buffer size */
    Gint    editpos;  /* editing position */
    Gpoint  interval; /* X, Y interval */
    Gfloat  time;     /* time interval */
    Gchar   *data;    /* device/implementation dependent data */
} Gstrokepet0001;

    typedef struct { /* COORDINATE POINT */
        Gfloat  x;    /* X coordinate */
        Gfloat  y;    /* Y coordinate */
    } Gpoint;

typedef Gstrokepet0001 Gstrokepet0002;

typedef struct {
    Gint    bufsiz;    /* input buffer size */
    Gint    editpos;  /* editing position */
    Gpoint  interval; /* X, Y interval */
    Gfloat  time;     /* time interval */
    Gacf    acf;      /* attribute control flag (constant) */
    Gmattr  mk;       /* marker attributes */
    Gchar   *data;    /* device/implementation dependent data */
} Gstrokepet0003;

    typedef struct { /* POLYMARKER ATTRIBUTES */
        Gasf    type;    /* marker type ASF (constant) */
        Gasf    size;    /* marker width ASF (constant) */
        Gasf    colour; /* marker color ASF (constant) */
        Gint    mark;    /* marker index */
        Gmbndl  bundl;   /* marker bundle */
    } Gmattr;

        typedef struct { /* POLYMARKER BUNDLE */
            Gint    type;    /* marker type (constant) */
            Gfloat  size;    /* marker size scale factor */
            Gint    colour; /* polymarker color index */
        } Gmbndl;

typedef struct {
    Gint    bufsiz;    /* input buffer size */
    Gint    editpos;  /* editing position */
    Gpoint  interval; /* X, Y interval */
    Gfloat  time;     /* time interval */
    Gacf    acf;      /* attribute control flag */
    Glnattr ln;       /* line attributes */
    Gchar   *data;    /* device/implementation dependent data */
} Gstrokepet0004;

    typedef struct { /* POLYLINE ATTRIBUTES */
        Gasf    type;    /* line type ASF */
        Gasf    width;   /* line width ASF */
        Gasf    colour; /* line color ASF */
        Gint    line;    /* line index */
        Glnbndl bundl;   /* line bundle */
    } Glnattr;

        typedef struct { /* POLYLINE BUNDLE */
            Gint    type;    /* line type (constant) */
            Gfloat  width;   /* line width scale factor */
            Gint    colour; /* polyline color index */
        } Glnbndl;
```

## Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gimode	GREQUEST	Request mode.
	GSAMPLE	Sample mode.
	GEVENT	Event mode.
Gesw	GECHO	Echo enabled.
	GNOECHO	Echo disabled.
Gacf	GCURRENT	Input data record current values.
	GSPECIFIED	Input data record specified values.
Gasf	GBUNDLED	Bundled attributes.
	GINDIVIDUAL	Individual attributes.
Marker types	GMK_POINT	Dot.
	GMK_PLUS	Plus sign.
	GMK_STAR	Asterisk.
	GMK_O	Circle.
	GMK_X	Diagonal cross.
Line types	GLN_SOLID	Solid line.
	GLN_DASHED	Dashed line.
	GLN_DOTTED	Dotted line.
	GLN_DASHDOT	Dashed-dotted line.

## Description

The INQUIRE STROKE DEVICE STATE 3 function returns the current state of the given stroke-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gstroke3)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the stroke data record.

Before calling this function, you must initialize the following data structure elements:

- The *n\_points* field of structure `Gstroke3` (`state→stroke.n_points`) must contain the maximum number of points that can be returned. The number of points should be large enough to hold all the entries. On output, this element contains the number of points returned.
- The *\*points* field of structure `Gstroke3` (`state→stroke.points`) must point to an array of type `Gpoint3` with *n\_points* elements.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

## INQUIRE STROKE DEVICE STATE 3

### See Also

INITIALIZE STROKE 3

SET STROKE MODE

Example 9-3 for a program example using an INQUIRE . . . DEVICE STATE function

---

## INQUIRE TEXT ALIGNMENT

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqtextalign (
    Gtxalign *align, /* (0) Text alignment */
    Gint *error /* (0) Error indicator */
)
```

### Data Structures

```
typedef struct { /* TEXT ALIGNMENT */
    Gtxhor hor; /* horizontal component (constant) */
    Gtxver ver; /* vertical component (constant) */
} Gtxalign;
```

### Constants

Data Type	Constant	Description
Gtxhor	GAH_NORMAL	Normal
	GAH_LEFT	Left
	GAH_CENTRE	Center
	GAH_RIGHT	Right
Gtxver	GAV_NORMAL	Normal
	GAV_TOP	Top
	GAV_CAP	Cap
	GAV_HALF	Half
	GAV_BASE	Base
	GAV_BOTTOM	Bottom

### Description

The INQUIRE TEXT ALIGNMENT function returns the value for the *current text alignment* entry in the GKS state list.

### See Also

SET TEXT ALIGNMENT

## INQUIRE TEXT COLOUR INDEX

---

### INQUIRE TEXT COLOUR INDEX

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqtextcolourind (  
    Gint    *index,    /* (0) Text color index */  
    Gint    *error     /* (0) Error indicator */  
)
```

#### Description

The INQUIRE TEXT COLOUR INDEX function returns the value for the *current text color index* entry in the GKS state list.

#### See Also

SET TEXT COLOUR INDEX

---

## INQUIRE TEXT EXTENT

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqtextextent (
    Gint      ws,          /* (I) Workstation identifier */
    Gpoint    *position,  /* (I) Text position */
    Gchar     *string,    /* (I) Character string */
    Gextent   *extent,    /* (O) Concatenation point and parallelogram */
    Gint      *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct { /* COORDINATE POINT */
    Gfloat  x;    /* X coordinate */
    Gfloat  y;    /* Y coordinate */
} Gpoint;

typedef struct { /* TEXT EXTENT */
    Gpoint  concat; /* concatenation point */
    Gpoint  bottom_left; /* corner 1 */
    Gpoint  bottom_right; /* corner 2 */
    Gpoint  top_right; /* corner 3 */
    Gpoint  top_left; /* corner 4 */
} Gextent;
```

### Description

The INQUIRE TEXT EXTENT function returns the concatenation point and text extent parallelogram for the specified text string.

The text extent is calculated using the currently selected text attributes. The text extent is returned as the four corner points of the enclosing parallelogram. The four points are in counterclockwise order. The concatenation point can be used as the text origin of a subsequent text origin. However, text string concatenation is not a meaningful combination of text path and text alignment.

### See Also

SET CHARACTER HEIGHT  
 SET CHARACTER UP VECTOR  
 SET TEXT ALIGNMENT  
 SET TEXT FONT AND PRECISION  
 TEXT

## INQUIRE TEXT EXTENT 3

---

## INQUIRE TEXT EXTENT 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqtexttext3 (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gpoint3   *position,   /* (I) Text position */  
    Gpoint3   *dirn_vec1,  /* (I) Vector 1 defining plane */  
    Gpoint3   *dirn_vec2,  /* (I) Vector 2 defining plane */  
    Gchar     *string,     /* (I) Character string */  
    Gextent3  *extent,     /* (O) Concatenation point and parallelogram */  
    Gint      *error       /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* COORDINATE POINT */  
    Gfloat     x;         /* X coordinate */  
    Gfloat     y;         /* Y coordinate */  
    Gfloat     z;         /* Z coordinate */  
} Gpoint3;  
  
typedef struct {          /* TEXT EXTENT 3 */  
    Gpoint3    concat;    /* concatenation point */  
    Gpoint3    corner_1;  /* corner 1 */  
    Gpoint3    corner_2;  /* corner 2 */  
    Gpoint3    corner_3;  /* corner 3 */  
    Gpoint3    corner_4;  /* corner 4 */  
} Gextent3;
```

### Description

The INQUIRE TEXT EXTENT 3 function returns the concatenation point and text extent parallelogram for the specified text string.

The text extent is calculated using the currently selected text attributes. The text direction vectors are specified as input arguments to the TEXT 3 function. These vectors, together with the text position, define the text plane. The text extent is returned as the four corner points of the enclosing parallelogram. The four points are in counterclockwise order. The concatenation point can be used as the text origin of a subsequent text origin. However, text string concatenation is not a meaningful combination of text path and text alignment.

### See Also

SET CHARACTER HEIGHT  
SET CHARACTER UP VECTOR  
SET TEXT ALIGNMENT  
SET TEXT FONT AND PRECISION  
TEXT 3



## INQUIRE TEXT FACILITIES

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqtextfacil (
    Gwstype      *type,          /* (I) Workstation type */
    Gint         bufsize,       /* (I) Size of allocated buffer */
    Gint         *num_fpp,      /* (O) Number of available font precision pairs */
    Gtxfac       *fac,         /* (O) Returned text facilities */
    Gint         *error         /* (O) Error indicator */
)
```

### Data Structures

```
typedef struct {
    Gint         fps;           /* TEXT FACILITIES */
                                /* number of fonts and precisions */
    Gtxfp       *fp_list;      /* list of available fonts and precisions */
    Gint         heights;      /* number of character heights */
    Gfloat       min_ht;       /* minimum height */
    Gfloat       max_ht;       /* maximum height */
    Gint         expansions;    /* number of character expansion factors */
    Gfloat       min_exp;      /* minimum expansion factor */
    Gfloat       max_exp;      /* maximum expansion factor */
    Gint         predefined;    /* number of predefined bundles */
} Gtxfac;

typedef struct {
    Gint         font;         /* TEXT FONT AND PRECISION */
                                /* text font */
    Gtxprec      prec;        /* text precision (constant) */
} Gtxfp;
```

### Constants

Data Type	Constant	Description
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.

### Description

The INQUIRE TEXT FACILITIES function returns the number of text font and precision pairs, a list of text font and precision pairs, the number of available character heights, the number of available character expansion factors, the minimum and maximum character expansion factors, and the number of text indexes available for the specified workstation type.

## INQUIRE TEXT FACILITIES

If the number of character heights or character expansion factors is 0, then the workstation supports a continuous range of character heights or character expansion factors.

---

## INQUIRE TEXT FONT AND PRECISION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqtextfontprec (
    Gtxfp    *fp,      /* (0) Text font and precision */
    Gint     *error    /* (0) Error indicator */
)
```

### Data Structures

```
typedef struct {          /* TEXT FONT AND PRECISION */
    Gint     font;       /* text font */
    Gtxprec  prec;      /* text precision (constant) */
} Gtxfp;
```

### Constants

Data Type	Constant	Description
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.

### Description

The INQUIRE TEXT FONT AND PRECISION function returns the value for the *current text font and precision* entry in the GKS state list.

This value is used to display subsequent TEXT output primitives, created when the current text font and precision ASF entry in the GKS state list is INDIVIDUAL. This value does not affect the display of subsequent TEXT output primitives, created when the current text font and precision ASF entry in the GKS state list is BUNDLED.

Text font and precision is a single text aspect; a particular text font can be available at some, but not necessarily all, precisions. The text precision value determines the fidelity with which the other text aspects are used. The values of text precision, in order of increasing fidelity, are string, character, and stroke.

### See Also

SET TEXT FONT AND PRECISION

## INQUIRE TEXT INDEX

---

### INQUIRE TEXT INDEX

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqtextind (  
    Gint    *index,    /* (0) Text index */  
    Gint    *error     /* (0) Error indicator */  
)
```

#### Description

The INQUIRE TEXT INDEX function returns the value for the *current text index* entry in the GKS state list.

The text bundle table contains entries for text font and precision, character expansion factor, character spacing, and text color index attribute values. When calling TEXT, DEC GKS uses the bundle table only if the corresponding ASF has been set to BUNDLED.

#### See Also

SET TEXT INDEX

---

## INQUIRE TEXT PATH

### Operating States

GKOP, WSOP, WSAC, GSOP

### Syntax

```
ginqtextpath (
    Gtxpath    *path,    /* (0) Text path (constant) */
    Gint       *error    /* (0) Error indicator */
)
```

### Constants

Data Type	Constant	Description
Gtxpath	GTP_RIGHT	Text string reads from left to right.
	GTP_LEFT	Text string reads from right to left.
	GTP_UP	Text string reads from bottom to top.
	GTP_DOWN	Text string reads from top to bottom.

### Description

The INQUIRE TEXT PATH function returns the value for the geometric attribute *current text path* entry in the GKS state list.

### See Also

SET TEXT PATH

# INQUIRE TEXT REPRESENTATION

---

## INQUIRE TEXT REPRESENTATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqtextrep (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gint      index,      /* (I) Text index */  
    Ginqtype  type,       /* (I) Return type (constant) */  
    Gtxbundl  *rep,       /* (O) Returned text representation */  
    Gint      *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* TEXT BUNDLE */  
    Gtxfp      fp;        /* font and precision */  
    Gfloat     exp;       /* character expansion */  
    Gfloat     space;     /* character spacing */  
    Gint       colour;    /* text color */  
} Gtxbundl;  
  
typedef struct {         /* TEXT FONT AND PRECISION */  
    Gint       font;     /* text font */  
    Gtxprec    prec;     /* text precision (constant) */  
} Gtxfp;
```

### Constants

Data Type	Constant	Description
Ginqtype	GSET	Use the exact state list values.
	GREALIZED	Use the values approximated by the graphics handler.
Gtxprec	GP_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GP_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GP_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.

### Description

The INQUIRE TEXT REPRESENTATION function returns the values currently associated with the specified text index value. Although the text font and precision values are set individually, the function returns them as a pair of values.

If the specified text index is not in the text bundle table on the specified workstation, and the specified type of returned values is REALIZED, the representation for text index 1 is used.

**See Also**

SET TEXT REPRESENTATION

## INQUIRE VALUATOR DEVICE STATE

---

## INQUIRE VALUATOR DEVICE STATE

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqvalst (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gint    dev,         /* (I) Locator device number */  
    Gint    bufsize,     /* (I) Allocated record buffer size */  
    Gint    *state_size, /* (O) Required record buffer size */  
    Gvalst  *state,      /* (O) Returned data structure */  
    Gint    *error       /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* VALUATOR STATE */  
    Gimode    mode;      /* mode (constant) */  
    Gesw      esw;       /* echo switch (constant) */  
    Gfloat    val;       /* valuator data */  
    Gint      pet;       /* prompt and echo type */  
    Glimit    e_area;    /* echo area */  
    Gvalrec   record;    /* valuator data record */  
} Gvalst;  
  
typedef struct {         /* COORDINATE LIMITS */  
    Gfloat    xmin;     /* X minimum limit */  
    Gfloat    xmax;     /* X maximum limit */  
    Gfloat    ymin;     /* Y minimum limit */  
    Gfloat    ymax;     /* Y maximum limit */  
} Glimit;  
  
typedef union {         /* VALUATOR DATA RECORD */  
    Gvalpet_0001    valpet_1_datarec;  
    Gvalpet_0002    valpet_2_datarec;  
    Gvalpet_0003    valpet_3_datarec;  
    Gvalpet0001     valpet1_datarec;  
    Gvalpet0002     valpet2_datarec;  
    Gvalpet0003     valpet3_datarec;  
} Gvalrec;  
  
typedef Gvalpetneg0001 Gvalpet_0001;  
    typedef Gvalpet0001 Gvalpetneg0001;  
        typedef struct {  
            Gfloat    low;          /* low range limit */  
            Gfloat    high;         /* high range limit */  
            Gchar    *title_string; /* the title string */  
        } Gvalpet0001;  
typedef Gvalpetneg0002 Gvalpet_0002;  
    typedef Gvalpetneg0001 Gvalpetneg0002;  
typedef Gvalpetneg0003 Gvalpet_0003;  
    typedef Gvalpetneg0001 Gvalpetneg0003;  
typedef Gvalpet0001 Gvalpet0002;
```



## INQUIRE VALUATOR DEVICE STATE

```
typedef Gvalpet0001 Gvalpet0003;
```

### Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled

### Description

The INQUIRE VALUATOR DEVICE STATE function returns the current state of the given valuator-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gvalrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the valuator data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE VALUATOR

SET VALUATOR MODE

Example 9-4 for a program example using the INQUIRE VALUATOR DEVICE STATE function

## INQUIRE VALUATOR DEVICE STATE 3

---

## INQUIRE VALUATOR DEVICE STATE 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqvalst3 (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gint    dev,        /* (I) Locator device number */  
    Gint    bufsize,    /* (I) Allocated record buffer size */  
    Gint    *state_size, /* (O) Required record buffer size */  
    Gvalst3 *state,     /* (O) Returned data structure */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* VALUATOR 3 STATE */  
    Gimode    mode;      /* locator mode (constant) */  
    Gesw      esw;       /* echo switch (constant) */  
    Gfloat    val;       /* valuator data */  
    Gint      pet;       /* prompt and echo type */  
    Glimit3   e_vol;     /* echo volume */  
    Gvalrec   record;    /* stroke data record */  
} Gvalst3;  
  
    typedef struct {      /* COORDINATE LIMITS */  
        Gfloat    xmin;   /* X minimum limit */  
        Gfloat    xmax;   /* X maximum limit */  
        Gfloat    ymin;   /* Y minimum limit */  
        Gfloat    ymax;   /* Y maximum limit */  
        Gfloat    zmin;   /* Z minimum limit */  
        Gfloat    zmax;   /* Z maximum limit */  
    } Glimit3;  
  
    typedef union {       /* VALUATOR DATA RECORD */  
        Gvalpet_0001    valpet_1_datarec;  
        Gvalpet_0002    valpet_2_datarec;  
        Gvalpet_0003    valpet_3_datarec;  
        Gvalpet0001     valpet1_datarec;  
        Gvalpet0002     valpet2_datarec;  
        Gvalpet0003     valpet3_datarec;  
    } Gvalrec;  
  
    typedef Gvalpetneg0001 Gvalpet_0001;  
        typedef Gvalpet0001 Gvalpetneg0001;  
            typedef struct {  
                Gfloat    low;          /* low range limit */  
                Gfloat    high;         /* high range limit */  
                Gchar    *title_string; /* the title string */  
            } Gvalpet0001;  
    typedef Gvalpetneg0002 Gvalpet_0002;  
        typedef Gvalpetneg0001 Gvalpetneg0002;  
    typedef Gvalpetneg0003 Gvalpet_0003;  
        typedef Gvalpetneg0001 Gvalpetneg0003;
```

## INQUIRE VALUATOR DEVICE STATE 3

```
typedef Gvalpet0001 Gvalpet0002;  
typedef Gvalpet0001 Gvalpet0003;
```

### Constants

Data Type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode
Gesw	GECHO	Echo enabled
	GNOECHO	Echo disabled

### Description

The INQUIRE VALUATOR DEVICE STATE 3 function returns the current state of the given valuator-class logical input device.

The information returned by this function depends on the value of the *bufsize* argument. If *bufsize* is equal to the `sizeof(Gvalrec)`, DEC GKS returns all the state information. If *bufsize* is equal to 0, DEC GKS returns all the information except the valuator data record.

Note that the title string or device- or implementation-specific data, or both, is returned as a pointer to the information stored in DEC GKS.

### See Also

INITIALIZE VALUATOR 3  
SET VALUATOR MODE

Example 9–4 for a program example using the INQUIRE VALUATOR DEVICE STATE function

## INQUIRE VIEW FACILITIES

---

### INQUIRE VIEW FACILITIES

#### Operating States

GKOP, WSOP, WSAC, SGOP

#### Syntax

```
ginqviewfac (  
    Gwstype    *type,          /* (I) Workstation type */  
    Gint       *num_indices,   /* (I) Number of predefined view indexes */  
    Gint       *error          /* (O) Error indicator */  
)
```

#### Description

The INQUIRE VIEW FACILITIES function returns the number of predefined view indexes for the specified workstation type.

## INQUIRE VIEW REPRESENTATION 3

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqviewrep3 (
    Gint      ws,          /* (I) Workstation identifier */
    Gint      index,      /* (I) View index */
    Gviewus   *xform_state, /* (O) Transformation state (constant) */
    Gviewbundl *req_rep,  /* (O) Requested view representation */
    Gviewbundl *cur_rep,  /* (O) Current view representation */
    Gint      *error      /* (O) Error indicator */
)
```

### Data Structures

```
typedef Gwstus Gviewus;          /* view update state (constant) */
typedef struct {                  /* VIEW REPRESENTATION BUNDLE */
    Gfloat      vo_matrix[4][4];  /* view orientation matrix */
    Gfloat      vm_matrix[4][4];  /* view mapping matrix */
    Glimit3     vc_limits;        /* clipping limits */
    Gclip       req_xy_clip;      /* X-Y clipping indicator (constant) */
    Gclip       req_b_clip;      /* back clip indicator (constant) */
    Gclip       req_f_clip;      /* front clip indicator (constant) */
} Gviewbundl;

    typedef struct {              /* COORDINATE LIMITS */
        Gfloat      xmin;        /* X minimum limit */
        Gfloat      xmax;        /* X maximum limit */
        Gfloat      ymin;        /* Y minimum limit */
        Gfloat      ymax;        /* Y maximum limit */
        Gfloat      zmin;        /* Z minimum limit */
        Gfloat      zmax;        /* Z maximum limit */
    } Glimit3;
```

### Constants

Data Type	Constant	Description
Gviewus	GPENDING	Transformation is pending
	GNOTPENDING	Transformation is not pending
Gclip	GNOCLIP	Clipping disabled
	GCLIP	Clipping enabled

### Description

The INQUIRE VIEW REPRESENTATION 3 function queries the workstation state list and returns the view representation for the given view index. If a viewing transformation change was requested but not yet provided at the time of the call to this function, the viewing transformation update state is PENDING.

## **INQUIRE VIEW REPRESENTATION 3**

### **See Also**

SET VIEW REPRESENTATION 3

## INQUIRE WORKSTATION CATEGORY

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqwscategory (
    Gwstype  *type,      /* (I) Workstation type */
    Gwscat   *cat,      /* (O) Workstation category (constant) */
    Gint     *error     /* (O) Error indicator */
)
```

### Constants

Data Type	Constant	Description
Gwscat	GOUTPUT	Output category
	GINPUT	Input category
	GOUTIN	Output/input category
	GWISS	Workstation independent segment storage
	GMO	Metafile output category
	GMI	Metafile input category

### Description

The INQUIRE WORKSTATION CATEGORY function returns the workstation category for the specified workstation type.

# INQUIRE WORKSTATION CLASSIFICATION

---

## INQUIRE WORKSTATION CLASSIFICATION

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqwsclass (  
    Gwstype      *type,      /* (I) Workstation type */  
    Gwsclass     *class,     /* (O) Workstation classification (constant) */  
    Gint         *error      /* (O) Error indicator */  
)
```

### Constants

Data Type	Constant	Description
Gwsclass	GVECTOR	Vector class
	GRASTER	Raster class
	GOTHER	Other class

### Description

The INQUIRE WORKSTATION CLASSIFICATION function returns the workstation classification for the specified workstation type.

You can use the workstation classification to determine the validity of other DEC GKS return values. For example, if you are working on a device other than one that uses raster units to define pixel dimensions, the function INQUIRE DISPLAY SPACE SIZE will not return valid values to the raster unit arguments.



---

## INQUIRE WORKSTATION CONNECTION AND TYPE

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqwsconntype (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gint    bufsize,    /* (I) Size of passed buffer */  
    Gint    *ct_size,   /* (O) Output size of structure ct */  
    Gwsct   *ct,        /* (O) Returned connection and type */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* WS CONNECTION AND TYPE */  
    Gconn    *conn;      /* workstation connection */  
    Gwstype   *type;    /* workstation type */  
} Gwsct;
```

### Description

The INQUIRE WORKSTATION CONNECTION AND TYPE function returns the logical name or environment variable associated with the physical device connection from the host to the workstation, and the workstation type.

### See Also

OPEN WORKSTATION

Example 4-2 for a program example using the INQUIRE WORKSTATION CONNECTION AND TYPE function

# INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

---

## INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqwsdeferupdatest (  
    Gint      ws,          /* (I) Workstation identifier */  
    Gwsdsurf *du,         /* (O) Deferral mode and update state */  
    Gint      *error       /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* WORKSTATION DEFERRAL AND UPDATE STATE */  
    Gdefmode  defmode;    /* deferral mode (constant) */  
    Gdspsurf  dspsurf;    /* display surface (constant) */  
    Girgmode  irgmode;    /* implicit regeneration mode (constant) */  
    Gnframe   nframe;     /* new frame action at update (constant) */  
} Gwsdsus;
```

### Constants

Data Type	Constant	Description
Gdefmode	GASAP	Generate images as soon as possible.
	GBNIG	Generate images before the next interaction globally or before a sample or event input occurs.
	GBNIL	Generate images before the next interaction locally or before a sample or event input occurs.
	GASTI	Generate images at some time. The exact time is determined by the workstation.
Gdspsurf	GEMPTY	Display surface is empty.
	GNOTEMPTY	Display surface is not empty.
Girgmode	GSUPPRESSED	Implicit regeneration is suppressed.
	GALLOWED	Implicit regeneration is allowed.
Gnframe	GNO	No new frame action on update.
	GYES	New frame action on update.

### Description

The INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES function returns the current deferral state mode, implicit regeneration mode, and workstation surface status. It also indicates whether a new frame is necessary to update the screen.

## INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

### See Also

SET DEFERRAL STATE

## INQUIRE WORKSTATION MAXIMUM NUMBERS

---

## INQUIRE WORKSTATION MAXIMUM NUMBERS

### Operating States

GKOP, WSOP, WSAC, SGOP

### Syntax

```
ginqwsmaxnum (  
    Gwsmax    *maxws,    /* (0) Maximum number of open, active and  
                        associated workstations */  
    Gint      *error     /* (0) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* WS MAXIMUM NUMBERS */  
    Gint    open;        /* number of open workstations */  
    Gint    active;     /* number of active workstations */  
    Gint    assoc;     /* number of associated workstations */  
} Gwsmax;
```

### Description

The INQUIRE WORKSTATION MAXIMUM NUMBERS function returns the maximum number of open workstations, active workstations, and workstations that can be associated with a segment.

---

**INQUIRE WORKSTATION STATE**

**Operating States**

WSOP, WSAC, SGOP

**Syntax**

```
ginqwsst (
    Gint      ws,          /* (I) Workstation identifier */
    Gwsstate  *state,     /* (O) Workstation state (constant) */
    Gint      *error      /* (O) Error indicator */
)
```

**Constants**

Data Type	Constant	Description
Gwsstate	GINACTIVE	Workstation is not active.
	GACTIVE	Workstation is active.

**Description**

The INQUIRE WORKSTATION STATE function identifies the workstation state value as either GINACTIVE or GACTIVE.

**See Also**

ACTIVATE WORKSTATION

# INQUIRE WORKSTATION TRANSFORMATION

---

## INQUIRE WORKSTATION TRANSFORMATION

### Operating States

WSOP, WSAC, SGOP

### Syntax

```
ginqwstran (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gwsti   *wstran,    /* (O) Workstation transformation state and  
                        transformations */  
    Gint    *error      /* (O) Error indicator */  
)
```

### Data Structures

```
typedef struct {          /* WORKSTATION TRANSFORMATION INFORMATION */  
    Gwstus   wstus;      /* workstation transformation update state  
                        (constant) */  
  
    Gtran    request;    /* requested transformation */  
    Gtran    current;    /* current transformation */  
} Gwsti;  
  
typedef struct {         /* TRANSFORMATION */  
    Glimit   w;          /* window */  
    Glimit   v;          /* viewport */  
} Gtran;  
  
typedef struct {        /* COORDINATE LIMITS */  
    Gfloat   xmin;      /* X minimum limit */  
    Gfloat   xmax;      /* X maximum limit */  
    Gfloat   ymin;      /* Y minimum limit */  
    Gfloat   ymax;      /* Y maximum limit */  
} Glimit;
```

### Constants

Data Type	Constant	Description
Gwstus	GNOTPENDING	Action not pending
	G_PENDING	Action pending

### Description

The INQUIRE WORKSTATION TRANSFORMATION function returns the requested and current workstation windows and workstation viewports and a flag indicating whether a workstation transformation is pending.

If a workstation transformation change was requested but not yet provided at the time of the call to this function, the workstation transformation update state is PENDING.

## INQUIRE WORKSTATION TRANSFORMATION

### See Also

SET WORKSTATION VIEWPORT  
SET WORKSTATION WINDOW

## INQUIRE WORKSTATION TRANSFORMATION 3

---

### INQUIRE WORKSTATION TRANSFORMATION 3

#### Operating States

WSOP, WSAC, SGOP

#### Syntax

```
ginqwstran3 (  
    Gint    ws,          /* (I) Workstation identifier */  
    Gwsti3  *wstran,     /* (O) Workstation transformation state and  
                        transformations */  
    Gint    *error       /* (O) Error indicator */  
)
```

#### Data Structures

```
typedef struct {  
    Gwstus  wstus;       /* workstation transformation update state  
                        (constant) */  
    Gtran3  request;     /* requested transformation */  
    Gtran3  current;     /* current transformation */  
} Gwsti3;  
  
typedef struct { /* NORMALIZATION TRANSFORM */  
    Glimit3 w;          /* WC window limits */  
    Glimit3 v;          /* NDC viewport limits */  
} Gtran3;  
  
typedef struct { /* COORDINATE LIMITS */  
    Gfloat  xmin;       /* X minimum limit */  
    Gfloat  xmax;       /* X maximum limit */  
    Gfloat  ymin;       /* Y minimum limit */  
    Gfloat  ymax;       /* Y maximum limit */  
    Gfloat  zmin;       /* Z minimum limit */  
    Gfloat  zmax;       /* Z maximum limit */  
} Glimit3;
```

#### Constants

Data Type	Constant	Description
Gwstus	GNOTPENDING	Action not pending
	GPENDING	Action pending

#### Description

The INQUIRE WORKSTATION TRANSFORMATION 3 function returns the requested and current workstation windows and workstation viewports, and a flag indicating whether a workstation transformation is pending.

If a workstation transformation change was requested but not yet provided at the time of the call to this function, the workstation transformation update state is PENDING.



**See Also**

SET WORKSTATION VIEWPORT 3  
SET WORKSTATION WINDOW 3



## **Error-Handling Functions**

**Insert tabbed divider here. Then discard this sheet.**



---

## Error-Handling Functions

The DEC GKS error-handling functions provide a method for you to control the generation of messages to the user, and a method of exit when a DEC GKS function call generates an error.

DEC GKS recognizes a number of error conditions. These error conditions are detected within DEC GKS functions, within procedures called by DEC GKS functions (such as calls to the graphics handler procedures), or within other areas of the application program.

For errors occurring in areas of the application program other than in DEC GKS function calls, either the application program regains control or program execution terminates abnormally. If the application program regains control, it can attempt to properly close DEC GKS or, failing that, attempt an emergency closure by calling the function EMERGENCY CLOSE GKS. If the program terminates abnormally, the results are unpredictable. In the worst case, you lose all graphic information produced before the error.

For errors detected within procedures called by DEC GKS, if the procedure does not generate a fatal error, then the DEC GKS error handlers may be able to process the error and you should be able to save graphic data. Otherwise, the application program regains control, or the application program is forced to call EMERGENCY CLOSE GKS, or in the worst case, you lose all graphic data produced before the error.

For errors detected within DEC GKS functions, DEC GKS performs the following tasks:

1. Sets the DEC GKS error state to ON to prohibit modification of DEC GKS variables
2. Calls ERROR HANDLING and passes the appropriate arguments
3. Performs function-specific error reaction or cleanup
4. Sets the DEC GKS error state to OFF

You can allow DEC GKS to call its own error handler, or you can provide an error handler of your own. An application-supplied, error-handling function can interpret information about the error and store data in a data area for subsequent analysis. Because application-supplied handlers do not have to generate the standard DEC GKS error messages, such a handler can change the format or the text of the messages sent to the user. Also, application-supplied handlers can decide whether to abort a program or to continue despite generated errors, if the errors are not fatal.

## Error-Handling Functions

A fatal error occurs within DEC GKS when internal data structures are corrupted, or when accurate and meaningful execution of DEC GKS functions is no longer possible. When a fatal error occurs, DEC GKS executes the current error handler and then terminates execution of the application.

The GKS standard dictates that every error-handling function, whether it be the DEC GKS supplied function or an application-supplied function, accept the following information from DEC GKS upon error generation:

- The GKS error number corresponding to the appropriate error condition (see Appendix A, DEC GKS Error Messages)
- The name of the GKS function that generated the error condition
- The name of the error file specified in the application program in the call to OPEN GKS

To implement an application-supplied, error-handling function, define the ERROR HANDLING function in your application. Use the name *gerrorhand* and the listed order and type of the three arguments. After you link the application, DEC GKS will use the new error handler when errors occur.

### 12.1 Function Descriptions

This section describes the DEC GKS error functions in detail.

---

### EMERGENCY CLOSE GKS

#### Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

#### Syntax

gemergencyclosegks ( )

#### Description

The EMERGENCY CLOSE GKS function attempts to perform a rapid and orderly closure of DEC GKS. Usually, this function is called for error conditions detected outside DEC GKS. If possible, the call to this function closes any open segments, updates all active workstations, deactivates those workstations, closes all open workstations, and then closes DEC GKS.

#### See Also

Example 12-1 for a program example using the EMERGENCY CLOSE GKS function

## ERROR HANDLING

---

### ERROR HANDLING

#### Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

#### Syntax

```
gerrorhand (  
    Gint    errnum,      /* (I) Error message number */  
    Gint    func_id,    /* (I) Name of function that detected the error */  
    Gfile   *errfile    /* (I) GKS error file */  
)
```

#### Description

The ERROR HANDLING function calls the ERROR LOGGING function and allows continued program execution. By default, DEC GKS calls this function when it encounters an error condition.

To implement an application-supplied error handler, call this function in your application. After you link the application, DEC GKS calls the application error handler when an error occurs.

#### See Also

EMERGENCY CLOSE GKS

ERROR LOGGING

OPEN GKS

Example 12-1 for a program example using the ERROR HANDLING function



---

## ERROR LOGGING

### Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

### Syntax

```
gerrorlog (  
    Gint    error,          /* (I) Error message number */  
    Gint    func_id,       /* (I) Name of function that detected the error */  
    Gfile   *errfile       /* (I) GKS error file */  
)
```

### Description

The ERROR LOGGING function writes the standard DEC GKS error message, which includes the number of the error and the text of the message, to the error file and returns to the procedure or function from which it was called.

The error handler function supplied by DEC GKS (ERROR HANDLING) automatically calls the ERROR LOGGING function. An application-supplied error handler can call this function if necessary.

### See Also

ERROR HANDLING

Example 12–1 for a program example using the ERROR LOGGING function

## Error-Handling Functions

### 12.2 Program Example

## 12.2 Program Example

Example 12–1 illustrates the use of the error-handling functions. This program writes the error message to the file ERROR.DAT, and closes GKS and the workstation.

### Example 12–1 Creating an Error Handler

```
/*
 * This program sets up a new error handler, creates an error,
 * then logs the error to an error file. This program writes the error
 * message to the file ERROR.DAT, and closes GKS and the workstation.
 *
 * Key functions:
 *
 * ERROR HANDLING
 * ERROR LOGGING
 * EMERGENCY CLOSE GKS
 */

#include <stdio.h>
#include <gks.h>          /* C binding definitions file */

main ()
{
    Gconn    default_conid;
    Gwstype  default_wstype;
    Gpoint   tx;
    Gchar    *string      = {"test"};
    FILE     *errorfile, *fopen();
    Gint     ws_id        = 1;

    /* Open GKS and the workstation environments. */
    default_conid = GWC_DEF;
    default_wstype = GWS_DEF;

    errorfile = fopen ("error.dat", "w");
    gopengks (errorfile, 0);
    gopenws (ws_id, &default_conid, &default_wstype);

    /*
     * Cause an error (the workstation isn't active yet). The message will
     * not be displayed on the screen; it will be written to the file
     * ERROR.DAT. */
    tx.x = tx.y = 0.5;
    gtext (&tx, string);

    /* Release the GKS and workstation environments. */
    gclosews (ws_id);
    gclosegks ();
}
```

(continued on next page)

#### Example 12–1 (Cont.) Creating an Error Handler

```
/* User-defined error handler. */
int gerrorhand ( Gint  error_number,
                 Gint  function_number,
                 Gfile *error_file_name )
{
    Gint  ws_not_active = 5;
/* Tell the user the error handler was called. */
    printf (" ** Aborting from a severe error. Please check the file
            error.dat ** \n");
/* Write the error message to a log file. */
    gerrorlog (error_number, function_number, error_file_name);
/* If ERROR_5, abort. */
    if (error_number == ws_not_active)
        {
            gemergencyclosegks( );
            exit(1);
        }
}
```



## **Error Messages**

**Insert tabbed divider here. Then discard this sheet.**



# A

---

## DEC GKS Error Messages

This appendix lists each of the DEC GKS error messages and the number associated with each message. Each of the standard errors also has a constant associated with it. The constants can be used to write more readable code for examining the function completion states. Nonstandard errors must be referred to numerically. The available constants for the function completion states are listed in Appendix B. Consider the following example:

```

    .
    .
    .
/* Include the error symbol definitions file. */

/* Check for success. */
    if ( NO_ERROR = gopenws( ws_id, &default_conid, &default_wstype) )
        .
        .
        .
/* Check for an invalid workstation identifier. */
    if ( EWSIDINV = gactivatews( ws_id ) )
        .
        .
        .

```

Each of the condition status codes corresponds to the number of the appropriate DEC GKS error message. The NO\_ERROR code is of severity *success*; all the codes with positive numbers are of severity *error*; and all negative errors are implementation-specific messages of severity *error* or *fatal error*.

Some of the DEC GKS specific error messages substitute program information in the message text. In this appendix, the portion of the text to be substituted is shown as \*\*\*\*.

The following sections describe the DEC GKS error messages by category and in numerical order.

### A.1 Operating State Errors

Table A-1 lists the errors that result when you call a function that is not permitted in the current operating state. For a list of the functions that you can or cannot call in a given DEC GKS operating state, see Chapter 4.

# DEC GKS Error Messages

## A.1 Operating State Errors

**Table A-1 Operating State Errors**

<b>Error</b>	<b>Error Message</b>
0	Successful completion of routine **** User Action: None.
1	GKS not in proper state: GKS shall be in the state GKCL in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call CLOSE GKS before the current DEC GKS state changes to GKCL.)
2	GKS not in proper state: GKS shall be in the state GKOP in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call either the function OPEN GKS or CLOSE WORKSTATION before the DEC GKS state changes to GKOP.)
3	GKS not in proper state: GKS shall be in the state WSAC in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call either the function ACTIVATE WORKSTATION or CLOSE SEGMENT before the DEC GKS state changes to WSAC.)
4	GKS not in proper state: GKS shall be in the state SGOP in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function CREATE SEGMENT before the DEC GKS state changes to SGOP.)
5	GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function ACTIVATE WORKSTATION before the DEC GKS state changes to WSAC.)
6	GKS not in proper state: GKS shall be in the state WSOP or in the state WSAC in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function OPEN WORKSTATION before the DEC GKS state changes to WSOP.)
7	GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function OPEN WORKSTATION before the DEC GKS state changes to WSOP.)
8	GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function OPEN GKS before the DEC GKS state changes to GKOP.)



## A.2 Workstation Errors

Table A–2 lists the errors that result when you call a DEC GKS function with invalid or undefined arguments pertaining to workstations.

**Table A–2 Workstation Errors**

<b>Error</b>	<b>Error Message</b>
20	Specified workstation identifier is invalid in routine ****  User Action: Make sure you have opened a workstation associated with that identifier, that you are not trying to generate output to an inactive workstation, that the arguments are presented in the right order, and if you are using a variable to specify the workstation identifier, that the variable is declared to be an integer.
21	Specified connection identifier is invalid in routine ****  User Action: Check that the connection identifier is specified correctly. See the sections on specifying the connection identifier in in the programming specifics online manual, which is available on the kit. If you are using the default value, make sure the environment option has been defined.
22	Specified workstation type is invalid in routine ****  User Action: Check to make sure you passed either a DEC GKS constant or the corresponding integer value. The workstation type constants are listed in Appendix B. If you are using the default value, make sure the environment option has been defined.
23	Specified workstation type does not exist in routine ****  User Action: The implementation of GKS does not support the workstation type associated with the identifier you passed. Pass an identifier associated with a supported device. If you are using the default workstation type, you should use INQUIRE WORKSTATION CONNECTION AND TYPE to check if DEC GKS supports the currently defined workstation type.
24	Specified workstation is open in routine ****  User Action: Either remove the function call to OPEN WORKSTATION, or replace the incorrect workstation type argument.
25	Specified workstation is not open in routine ****  User Action: Call OPEN WORKSTATION and pass the appropriate workstation identifier.
26	Specified workstation cannot be opened in routine ****  User Action: Make sure you specify valid workstation types, bit masks, or environment options, and make sure the information corresponds to a supported, functional physical device.
27	Workstation Independent Segment Storage is not open in routine ****  User Action: You tried to copy, associate, or insert a segment from WISS to another workstation. Make sure you have opened WISS in a call to OPEN WORKSTATION, passing GWS_WISS as an argument.

(continued on next page)

## DEC GKS Error Messages

### A.2 Workstation Errors

Table A-2 (Cont.) Workstation Errors

Error	Error Message
28	Workstation Independent Segment Storage is already open in routine **** User Action: Either remove the function call to OPEN WORKSTATION, or replace the incorrect workstation type argument.
29	Specified workstation is active in routine **** User Action: Either remove the function call to ACTIVATE WORKSTATION, or replace the incorrect workstation identifier argument.
30	Specified workstation is not active in routine **** User Action: You tried to generate output on an inactive workstation. Call ACTIVATE WORKSTATION, passing the appropriate workstation.
31	Specified workstation is of category MO in routine **** User Action: You attempted to perform an operation that is not permissible on MO workstations. Either remove the function call, change the state of the MO workstation, or check to see if you passed the correct arguments to OPEN WORKSTATION.
32	Specified workstation is not of category MO in routine **** User Action: Open and activate an MO workstation.
33	Specified workstation is of category MI in routine **** User Action: You attempted to perform an operation that is not permissible on MI workstations. Either remove the function call, change the state of the MI workstation, or check to see if you passed the correct arguments to OPEN WORKSTATION.
34	Specified workstation is not of category MI in routine **** User Action: You tried to interpret a file that was not associated with an MI workstation. Open a workstation of category MI.
35	Specified workstation is of category INPUT in routine **** User Action: You attempted to perform an operation that is not permissible on workstations of category INPUT, such as generating output. Either remove the function call, change the workstation type to the needed category, or check to see if you passed the correct arguments to OPEN WORKSTATION.
36	Specified workstation is Workstation Independent Segment Storage in routine **** User Action: You attempted to perform an operation that is not permissible on workstations of category WISS, such as requesting input. Either remove the function call, check that the call uses the correct workstation identifier, or check to see if you passed the correct arguments to OPEN WORKSTATION.

(continued on next page)

**Table A-2 (Cont.) Workstation Errors**

<b>Error</b>	<b>Error Message</b>
37	Specified workstation is not of category OUTIN in routine **** User Action: Either remove the function call, open and activate an OUTIN workstation, or check to see if you passed the correct arguments to OPEN WORKSTATION.
38	Specified workstation is neither of category INPUT nor of category OUTIN in routine **** User Action: Either remove the function call, change the workstation type to the needed category, or check to see if you passed the correct arguments to OPEN WORKSTATION.
39	Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** User Action: You attempted to perform an operation that is only permissible on workstations of category OUTPUT or OUTIN, such as generating output. Either remove the function call, open and activate a workstation of the correct category, or check to see if you passed the correct arguments to OPEN WORKSTATION.
40	Specified workstation has no pixel store readback capability in routine **** User Action: You called one of the pixel inquiry functions for a device incapable of returning such information. Either remove the function call, or make sure you passed the correct workstation identifier.
41	Specified workstation type is not able to generate the specified generalized drawing primitive in routine **** User Action: Either remove the function call to GENERALIZED DRAWING PRIMITIVE, or make sure you passed the correct GDP identifier.
42	Maximum number of simultaneously open workstations would be exceeded in routine **** User Action: You must remove the function call to OPEN WORKSTATION. You can use INQUIRE WORKSTATION MAXIMUM NUMBERS to determine the maximum number of open workstations that DEC GKS supports.
43	Maximum number of simultaneously active workstations would be exceeded in routine **** User Action: You must remove the function call to ACTIVATE WORKSTATION. You can use INQUIRE WORKSTATION MAXIMUM NUMBERS to determine the maximum number of active workstations that DEC GKS supports.

### A.3 Transformation Function Errors

Table A-3 lists the errors that result when you call a DEC GKS transformation function with invalid or undefined arguments.

**Table A-3 Transformation Function Errors**

<b>Error</b>	<b>Error Message</b>
50	Transformation number is invalid in routine ****

(continued on next page)

## DEC GKS Error Messages

### A.3 Transformation Function Errors

**Table A-3 (Cont.) Transformation Function Errors**

Error	Error Message
	User Action: Make sure that either the arguments are specified in the correct order, the transformation number is not negative, or the transformation number is an integer.
51	<p>Rectangle definition is invalid in routine ****</p> <p>User Action: Either the normalization window or viewport is invalid. Make sure you have not reversed the order of the X and Y argument values, that your coordinate values form a valid rectangle, and that your coordinate values are real numbers.</p>
52	<p>Viewport is not within the Normalized Device Coordinate unit square in routine ****</p> <p>User Action: DEC GKS allows unclipped primitives to exceed the NDC unit square (<math>[1,0,0] \times [0,1,0] \times [0,0,1]</math>), but does not allow you to define a normalization viewport whose boundaries exceed this square. Redefine the function normalization viewport.</p>
53	<p>Workstation window is not within the Normalized Device Coordinate unit square in routine ****</p> <p>User Action: Redefine the function normalization viewport to be within the NDC square (<math>[1,0,0] \times [0,1,0] \times [0,0,1]</math>).</p>
54	<p>Workstation viewport is not within the display space in routine ****</p> <p>User Action: Make sure you have not reversed the order of the X, Y, and Z argument values, your coordinate values form a valid rectangle, and your coordinate values are real numbers. You can use the function INQUIRE DISPLAY SPACE SIZE to determine the maximum X, Y, and Z values of the device coordinate plane.</p>

### A.4 Attribute Function Errors

Table A-4 lists the errors that result when you call the DEC GKS attribute functions with invalid or undefined arguments.

**Table A-4 Attribute Function Errors**

Error	Error Message
60	<p>Polyline index is invalid in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and the index is an integer.</p>
61	<p>A representation for the specified polyline index has not been defined on this workstation in routine ****</p> <p>User Action: Use SET POLYLINE REPRESENTATION to define a representation for the index, or use another predefined index value.</p>
62	<p>A representation for the specified polyline index has not been predefined on this workstation in routine ****</p>

(continued on next page)

## DEC GKS Error Messages A.4 Attribute Function Errors

**Table A-4 (Cont.) Attribute Function Errors**

<b>Error</b>	<b>Error Message</b>
	User Action: Use SET POLYLINE REPRESENTATION to define a representation for the index, or use another predefined index value.
63	Specified line type is equal to zero in routine **** User Action: Make sure the order and the number of the arguments is correct. If you used an inquiry function to obtain a default line type, check the order of the arguments passed to the inquiry function.
64	Specified line type is not supported on this workstation in routine **** User Action: Change the line type specification. You can use the function INQUIRE POLYLINE FACILITIES to obtain a list of supported line types for a given workstation.
65	Line width scale factor is less than zero in routine **** User Action: Either change the scale factor, or check the order and the number of the specified arguments.
66	Polymarker index is invalid in routine **** User Action: Make sure the arguments are specified in the correct order and the index is an integer.
67	A representation for the specified polymarker index has not been defined on this workstation in routine **** User Action: Use SET POLYMARKER REPRESENTATION to define a representation for a given index, or use another predefined index value.
68	A representation for the specified polymarker index has not been predefined on this workstation in routine **** User Action: Use SET POLYMARKER REPRESENTATION to define a representation for a given index, or use another predefined index value.
69	Specified marker type is equal to zero in routine **** User Action: Make sure the order of the arguments is correct. If you used an inquiry function to obtain a default marker type, check the order of the arguments passed to the inquiry function.
70	Specified marker type is not supported on this workstation in routine **** User Action: Change the marker type specification. You can use the function INQUIRE POLYMARKER FACILITIES to obtain a list of supported line types for a given workstation.
71	Marker size scale factor is less than zero in routine **** User Action: Either change the scale factor, or check the order and the number of the specified arguments.

(continued on next page)

## DEC GKS Error Messages

### A.4 Attribute Function Errors

Table A-4 (Cont.) Attribute Function Errors

Error	Error Message
72	<p>Text index is invalid in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and the index is an integer.</p>
73	<p>A representation for the specified text index has not been defined on this workstation in routine ****</p> <p>User Action: Use SET TEXT REPRESENTATION to define a representation for the index value, or use another predefined index value.</p>
74	<p>A representation for the specified text index has not been predefined on this workstation in routine ****</p> <p>User Action: Use SET TEXT REPRESENTATION to define a representation for the index value, or use another predefined index value.</p>
75	<p>Text font is equal to zero in routine ****</p> <p>User Action: Either change the font number, or check the order and the number of the specified arguments. If you used an inquiry function to obtain a default value, check the order and the number of the arguments passed to the inquiry function.</p>
76	<p>Requested text font is not supported for the specified precision on this workstation in routine ****</p> <p>User Action: Lower the precision or change the font number.</p>
77	<p>Character expansion factor is less than or equal to zero in routine ****</p> <p>User Action: Either change the expansion factor value or check the order and the number of the arguments. If you used an inquiry function to obtain a default value, check the order and the number of the arguments passed to the inquiry function.</p>
78	<p>Character height is less than or equal to zero in routine ****</p> <p>User Action: Either change the height value, or check the order and the number of the arguments. If you used an inquiry function to obtain a default value, check the order and the number of the arguments passed to the inquiry function.</p>
79	<p>Length of character up vector is zero in routine ****</p> <p>User Action: Either change the character up vector, or check the order and the number of the arguments. If you used an inquiry function to obtain a default value, check the order and the number of the arguments passed to the inquiry function.</p>
80	<p>Fill area index is invalid in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and the index is an integer.</p>

(continued on next page)

## DEC GKS Error Messages A.4 Attribute Function Errors

**Table A-4 (Cont.) Attribute Function Errors**

<b>Error</b>	<b>Error Message</b>
81	<p>A representation for the specified fill area index has not been defined on this workstation in routine ****</p> <p>User Action: Use SET FILL AREA REPRESENTATION to define a representation for the given index value, or pass another predefined index value.</p>
82	<p>A representation for the specified fill area index has not been predefined on this workstation in routine ****</p> <p>User Action: Use SET FILL AREA REPRESENTATION to define a representation for the given index value, or pass another predefined index value.</p>
83	<p>Specified fill area interior style is not supported on this workstation in routine ****</p> <p>User Action: Change the interior style specification. You can use the function INQUIRE FILL AREA FACILITIES to obtain a list of supported interior styles for a given workstation.</p>
84	<p>Style (pattern or hatch) index is equal to zero in routine ****</p> <p>User Action: Either change the style index, or check the order and the number of the specified arguments. If you used an inquiry function to obtain a style index, check the order and the number of the arguments passed to the inquiry function.</p>
85	<p>Specified pattern index is invalid in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and the index is an integer.</p>
86	<p>Specified hatch style is not supported on this workstation in routine ****</p> <p>User Action: Either replace the hatch style index, or check the order and the number of the arguments. The inquiry function INQUIRE FILL AREA FACILITIES returns the list of available hatch style indexes.</p>
87	<p>Pattern size value is not positive in routine ****</p> <p>User Action: Either alter the size of the pattern, or check the order and the number of the arguments. If you used an inquiry function to obtain the size of the pattern, check the order and the number of the arguments passed to the inquiry function.</p>
88	<p>A representation for the specified pattern index has not been defined on this workstation in routine ****</p> <p>User Action: Use SET PATTERN REPRESENTATION to define a representation for the pattern index, or pass another predefined index to the function.</p>
89	<p>A representation for the specified pattern index has not been predefined on this workstation in routine ****</p> <p>User Action: Use SET PATTERN REPRESENTATION to define a representation for the pattern index, or pass another predefined index to the function.</p>

(continued on next page)

## DEC GKS Error Messages

### A.4 Attribute Function Errors

**Table A-4 (Cont.) Attribute Function Errors**

Error	Error Message
90	Interior style PATTERN is not supported on this workstation in routine **** User Action: Specify another interior style to SET FILL AREA INTERIOR STYLE.
91	Dimensions of color array are invalid in routine **** User Action: One or more of the arguments passed to CELL ARRAY are invalid. Make sure the color array is a two-dimensional array, that you have not specified more rows and columns in the cell array that exist from the offset point to the end of the array, and that the cell array contains integers representing colors supported on that workstation.
92	Color index is less than zero in routine **** User Action: Either remove the index, or check the order and the number of the arguments. If you used an inquiry function to obtain the color index value, check the order and the number of the arguments passed to the inquiry function.
93	Color index is invalid in routine **** User Action: Make sure the arguments are specified in the correct order and that the index is an integer.
94	A representation for the specified color index has not been defined on this workstation in routine **** User Action: Use SET COLOUR REPRESENTATION to define a color representation for the index value, or pass another predefined index value.
95	A representation for the specified color index has not been predefined on this workstation in routine **** User Action: Use SET COLOUR REPRESENTATION to define a color representation for the index value, or pass another predefined index value.
96	Color index is outside range of the current color model in routine **** User Action: Check the color model range.
97	Pick identifier is invalid in routine **** User Action: Either remove the call to SET PICK IDENTIFIER or make sure the pick identifier is an integer. If you obtained the pick identifier from an inquiry function, check the order and the number of the arguments passed to the inquiry function.
98	Specified colour model is not available on the workstation in routine **** User Action: Call the INQUIRE COLOUR MODEL FACILITIES function for the list of color models supported by your workstation.

### A.5 Output Function Errors

Table A-5 lists the errors that result when you call a DEC GKS output function with invalid or undefined arguments.



**Table A–5 Output Function Errors**

Error	Error Message
100	<p>Number of points is invalid in routine ****</p> <p>User Action: The number of points specified does not match the number of coordinate points passed. Either alter the specified number of points, or alter the number of coordinate values contained in the arrays passed as arguments.</p>
101	<p>Invalid code in string in routine ****</p> <p>User Action: Your text string contained characters that cannot be printed. Remove the characters.</p>
102	<p>Generalized drawing primitive identifier is invalid in routine ****</p> <p>User Action: Specify another identifier or check to see if the identifier is an integer value.</p>
103	<p>Content of generalized drawing primitive data record is invalid in routine ****</p> <p>User Action: Make sure you passed a correct data record size.</p>
104	<p>At least one active workstation is not able to generate the specified generalized drawing primitive in routine ****</p> <p>User Action: Deactivate the workstations that do not generate the GDPs, or redefine the GDP data record so that all the workstations can generate the primitive.</p>
105	<p>At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformation and clipping rectangle in routine ****</p> <p>User Action: Either redefine the current normalization transformation (creating a different clipping rectangle), or supply different WC points so that the GDP falls within the current clipping rectangle.</p>

## A.6 Segment Function Errors

Table A–6 lists the errors that result when you call a DEC GKS segment function with invalid or undefined arguments.

**Table A–6 Segment Function Errors**

Error	Error Message
120	<p>Specified segment name is invalid in routine ****</p> <p>User Action: Either check the number and the order of the arguments or make sure the segment name is an integer value. If you obtained the segment name from an inquiry function, check the order and the number of the arguments passed to the inquiry function.</p>
121	<p>Specified segment name is already in use in routine ****</p> <p>User Action: Either remove the call to CREATE SEGMENT or check to make sure you specified the correct segment name.</p>

(continued on next page)

## DEC GKS Error Messages

### A.6 Segment Function Errors

**Table A-6 (Cont.) Segment Function Errors**

Error	Error Message
122	<p>Specified segment does not exist in routine ****</p> <p>User Action: Either check the order and the number of the arguments or make sure you specified an integer value as a segment name. If you used an inquiry function to obtain the segment name, check the order and the number of the arguments passed to the inquiry function.</p>
123	<p>Specified segment does not exist on specified workstation in routine ****</p> <p>User Action: Either remove the function call, or if the segment exists in WISS, associate the segment with the appropriate workstation.</p>
124	<p>Specified segment does not exist on Workstation Independent Segment Storage in routine ****</p> <p>User Action: You attempted to copy, associate, or insert a segment that is not stored in WISS. Either remove the function call or check to see that you specified the correct segment name.</p>
125	<p>Specified segment is open in routine ****</p> <p>User Action: Either remove the call to CREATE SEGMENT or specify another segment name.</p>
126	<p>Segment priority is outside the range [0,1] in routine ****</p> <p>User Action: Change the specified segment priority. If you used an inquiry function to obtain the segment priority value, check the order and the number of the arguments passed to the inquiry function.</p>

## A.7 Input Function Errors

Table A-7 lists the errors that result when you call a DEC GKS input function with invalid or undefined arguments.

**Table A-7 Input Function Errors**

Error	Error Message
140	<p>Specified input device is not present on workstation in routine ****</p> <p>User Action: Make sure you specified the function that applies to the correct logical input device and the correct workstation identifier.</p>
141	<p>Input device is not in REQUEST mode in routine ****</p> <p>User Action: Use one of the SET MODE input functions to set the logical input device to request mode before using this logical input device.</p>
142	<p>Input device is not in SAMPLE mode in routine ****</p> <p>User Action: Use one of the SET MODE input functions to set the logical input device to sample mode before using this logical input device.</p>

(continued on next page)

**Table A-7 (Cont.) Input Function Errors**

<b>Error</b>	<b>Error Message</b>
143	<p>EVENT and SAMPLE input mode are not available at this level of GKS in routine ****</p> <p>User Action: This error code is required by the standard, but DEC GKS will never generate this error.</p>
144	<p>Specified prompt and echo type is not supported on this workstation in routine ****</p> <p>User Action: Make sure the order of the arguments is correct or change the prompt and echo value. If you obtained the prompt and echo type from an inquiry function, check the order and the number of the arguments passed to the inquiry function.</p>
145	<p>Echo area is outside display space in routine ****</p> <p>User Action: Make sure the specified coordinate points are real values that specify a valid rectangle on the display surface. If you used an inquiry function to obtain the echo area, check the order and the number of the arguments passed to the inquiry function.</p>
146	<p>Contents of input data record are invalid in routine ****</p> <p>User Action: Make sure you specified the correct size of the data record, that the elements of the data record are of the correct data type, and that you have chosen the correct corresponding prompt and echo type. If you used an inquiry function to obtain the data record, check the order and number of the arguments passed to the inquiry function. Also, make sure you have not specified input values that are not accepted by the particular device; you can check the device's capabilities by calling one of the DEFAULT DATA inquiry functions.</p>
147	<p>Input queue has overflowed in routine ****</p> <p>User Action: Check the input queue with greater frequency or flush the input queue.</p>
148	<p>Input queue has not overflowed since GKS was opened or the last invocation of INQUIRE INPUT QUEUE OVERFLOW in routine ****</p> <p>User Action: You called INQUIRE INPUT QUEUE OVERFLOW when the queue was not full, and had not been filled since the beginning of your application. Continue to generate events, if your application still requires input.</p>
149	<p>Input queue has overflowed, but associated workstation has been closed in routine ****</p> <p>User Action: You called INQUIRE INPUT QUEUE OVERFLOW when the queue was full, but since the workstation is closed, information about the overflow is not available. You can set the devices to request mode (removing their prompts from the workstation surface), and then you can either process reports from the queue until empty or you can flush the queue of all reports.</p>
150	<p>No input value of the correct class is in the current event report in routine ****</p> <p>User Action: Make sure you check the input class argument passed to AWAIT EVENT before you try to call the appropriate GET function.</p>

(continued on next page)

## DEC GKS Error Messages

### A.7 Input Function Errors

**Table A-7 (Cont.) Input Function Errors**

Error	Error Message
151	Timeout is invalid in routine **** User Action: Make sure the timer argument in AWAIT EVENT is a real value between 0.0 and 356,400 seconds, as specified in the format described in the AWAIT EVENT function description in Chapter 9.
152	Initial value is invalid in routine **** User Action: Either check to make sure you specified the correct value, or check the capabilities of the device to see if you requested a value unsupported by the device. If you obtained the value from an inquiry function, check the order and number of arguments specified to the inquiry function.
153	Number of points in the initial stroke is greater than the buffer size in routine **** User Action: Either increase the size of the buffer or reduce the number of points in the initial stroke.
154	Length of initial string is greater than the buffer size in routine **** User Action: Either increase the size of the buffer or decrease the size of the initial string.

## A.8 Metafile Function Errors

Table A-8 lists the errors that result when you call a DEC GKS metafile function with invalid or undefined arguments.

**Table A-8 Metafile Function Errors**

Error	Error Message
160	Item type is not allowed for user items in routine **** User Action: Use an item type greater than or equal to 101, or less than 0 for a user item.
161	Item length is invalid in routine **** User Action: The length of the data item was shorter than necessary for its type. Make sure DEC GKS does not truncate your record when reading the item from a GKSM.
162	No item is left in GKS Metafile input in routine **** User Action: You tried to read past the end of the GKSM. Do not attempt to read items past the item of type 0.
163	Metafile item is invalid in routine **** User Action: Your item data was incorrect. Make sure DEC GKS did not truncate the item while reading from a GKSM and that you specified correct sizes and types. Make sure you are not trying to interpret a user-defined record type. User-defined records have item numbers greater than 100.

(continued on next page)

**Table A–8 (Cont.) Metafile Function Errors**

Error	Error Message
164	Item type is not a valid GKS item in routine **** User Action: You tried to interpret an item of type less than 0 or greater than 100. Make sure DEC GKS did not truncate the item while reading from a GKSM and that you specified correct sizes and types.
165	Content of item data record is invalid for the specified item type in routine **** User Action: There was unexpected or incorrect information in the data record. Make sure you pass the correct storage area.
166	Maximum item data record length is invalid in routine **** User Action: Make sure the data length is not negative.
167	User item cannot be interpreted in routine **** User Action: Do not pass user items to DEC GKS for interpretation.
168	Specified function is not supported in this level of GKS in routine **** User Action: This error code is required by the standard, but DEC GKS will never generate this error.

## A.9 Escape Function Errors

Table A–9 lists the errors that result when you call a DEC GKS escape function with invalid or undefined arguments.

**Table A–9 Escape Function Errors**

Error	Error Message
180	Specified escape function is not supported in routine **** User Action: Check the escape function identifier to make sure that it is a valid integer representing an escape function, and make sure you specified the correct workstation identifier.
181	Specified escape function identifier is invalid in routine **** User Action: Make sure the escape function identifier is a valid integer value.
182	Contents of escape data record are invalid in routine **** User Action: Make sure you specified the correct size of the data record. Also, make sure the elements of the data record are declared to be the correct data type.

## A.10 Miscellaneous Errors

Table A–10 lists the DEC GKS miscellaneous errors.

## DEC GKS Error Messages

### A.10 Miscellaneous Errors

**Table A-10 Miscellaneous Errors**

Error	Error Message
200	Specified error file is invalid in routine **** User Action: Make sure your specified error handler exists and that it includes the three required parameters in its definition.

### A.11 System Errors

Table A-11 lists implementation-dependent system errors.

**Table A-11 System Errors**

Error	Error Message
300	Storage overflow has occurred in GKS in routine **** User Action: Increase the swap space or the process virtual memory limit.
301	Storage overflow has occurred in segment storage in routine **** User Action: Increase the swap space or the process virtual memory limit.
302	Input/Output error has occurred while reading in routine **** User Action: You specified an illegal metafile for a metafile input workstation. Make sure you specified a valid GKSM or GKS3 metafile, and that you correctly specified the connection identifier.
303	Input/Output error has occurred while writing in routine **** User Action: You specified an illegal metafile for a metafile output workstation. Make sure you specified a valid GKSM or GKS3 metafile, and that you correctly specified the connection identifier.
304	Input/Output error has occurred while sending data to a workstation in routine **** User Action: Submit an SPR.
305	Input/Output error has occurred while receiving data from a workstation in routine **** User Action: Submit an SPR.
306	Input/Output error has occurred during program library management in routine **** User Action: Submit an SPR.
307	Input/Output error has occurred while reading workstation description table in routine **** User Action: Submit an SPR.

(continued on next page)

**Table A-11 (Cont.) System Errors**

<b>Error</b>	<b>Error Message</b>
308	Arithmetic error has occurred in routine **** User Action: You either divided by 0 or caused data overflow. Check the arguments passed in the function call.
400	View up vector and view plane normal are collinear in routine **** User Action: Make sure the view up vector and view plane normal are not collinear.
401	View plane normal is a null vector in routine **** User Action: Make sure the view plane normal is not {0.0, 0.0, 0.0} (null vector).
402	View up vector is a null vector in routine **** User Action: Make sure the view up vector is not {0.0, 0.0, 0.0} (null vector).
403	Projection viewport limits are not within NPC range in routine **** User Action: Make sure the projection viewport limits are in the range 0.0 to 1.0.
404	Projection reference point is between the front and back clipping planes in routine **** User Action: Make sure the projection reference point is not between the front and back clipping planes.
405	Projection reference point is on the view plane in routine **** User Action: Make sure the projection reference point is not on the view plane.
406	Box definition is invalid in routine **** User Action: Make sure the defined box follows the rule XMIN < XMAX and YMIN < YMAX.
407	Viewport is not within NDC unit cube in routine **** User Action: Make sure the viewport limits are in the range 0.0 to 1.0.
408	Specified view index is invalid in routine **** User Action: Make sure the view index is a valid index.
409	A representation for the specified view index has not been defined on this workstation in routine **** User Action: Make sure you specified the view index before you call a function that requires the view index argument. Use the INQUIRE LIST OF VIEW INDICES function for a list of the defined view indexes.

(continued on next page)

## DEC GKS Error Messages

### A.11 System Errors

Table A-11 (Cont.) System Errors

Error	Error Message
410	<p>A representation for the specified view index has not been predefined on this workstation in routine ****</p> <p>User Action: Make sure the specified view index is predefined. Use the INQUIRE PREDEFINED VIEW REPRESENTATION function for a list of the predefined view indexes.</p>
411	<p>Workstation window limits are not within the NPC cube in routine ****</p> <p>User Action: Make sure the workstation window limits are in the range 0.0 to 1.0.</p>
412	<p>Back clipping plane is in front of front clipping plane in routine ****</p> <p>User Action: Make sure the back clipping plane is behind the front clipping plane.</p>
413	<p>View clipping limits not within NPC range in routine ****</p> <p>User Action: Make sure the view clipping limits are in the range 0.0 and 1.0.</p>
420	<p>Edge index is invalid in routine ****</p> <p>User Action: Make sure the edge index is greater than or equal to 1.</p>
421	<p>A representation for the specified edge index has not been defined on this workstation in routine ****</p> <p>User Action: Make sure the edge representation is predefined. Use the INQUIRE EDGE FACILITIES function for information on the predefined edge representations.</p>
422	<p>A representation for the specified edge index has not been predefined on this workstation in routine ****</p> <p>User Actions: Make sure the specified edge index has been predefined.</p>
423	<p>Edge type is equal to zero in routine ****</p> <p>User Actions: Make sure the edge type is one of the permitted values. See Appendix B for a list of edge type values.</p>
424	<p>Specified edge type is not supported on this workstation in routine ****</p> <p>User Action: Make sure the edge type is one of the permitted values. See the chapter on your workstation in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for a list of supported edge types.</p>
425	<p>Edge width scale factor is less than zero in routine ****</p> <p>User Action: Make sure the edge width scale factor is greater than or equal to 0.0.</p>
426	<p>Pattern reference vectors are collinear in routine ****</p> <p>User Action: Make sure the pattern reference vectors are not collinear.</p>

(continued on next page)



**Table A–11 (Cont.) System Errors**

<b>Error</b>	<b>Error Message</b>
427	Specified HLHSR mode not supported on workstation in routine **** User Action: Make sure the specified HLHSR mode is supported by your workstation. See the chapter on your workstation in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for a list of supported HLHSR modes.
428	Specified HLHSR identifier is invalid in routine **** User Action: Make sure the HLHSR identifier is greater than or equal to 0.
429	Specified HLHSR mode is invalid in routine **** User Action: Make sure the HLHSR mode is greater than or equal to 0.
430	The text direction vectors are collinear in routine **** User Action: Make sure the text direction vectors are not collinear.
431	List of point lists is invalid in routine **** User Action: Make sure the number of lists is greater than 0, and the number of points for a set is greater than or equal to 3.
432	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformation and clipping volume in routine **** User Action: Do nothing. This is an informational message.

## A.12 C Binding Errors

Table A–12 lists the errors that are specific to the C binding functions.

**Table A–12 C Binding Errors**

<b>Error</b>	<b>Error Message</b>
2200	Buffer overflow on input or inquiry function in routine **** User Action: Make the buffer bigger.

## A.13 Implementation-Specific Errors

All the DEC GKS specific errors are negative. These errors are either of severity "error" or "fatal error." Error numbers in the range –90 to –100 are fatal errors. If one of these errors occurs, submit a Software Performance Report (SPR) indicating the error number, corresponding message, and any relevant particulars. For more information concerning SPRs, see the DEC GKS installation guide.

Table A–13 lists the errors that are implementation specific.

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

**Table A-13 Implementation-Specific Errors**

Error	Error Message
-2	<p>Requested color map could not be created as specified in routine ****</p> <p>User Action: The specified color map is too large. Check to make sure you specified the correct color map size and type (either physical or virtual) and that you have not exceeded the limitations of your device.</p>
-3	<p>Invalid data in workstation description file in routine ****</p> <p>User Action: Make sure the format of your description file is valid for your particular workstation.</p>
-4	<p>Invalid bit mask in workstation type in routine ****</p> <p>User Action: Check to make sure you specified a bit mask workstation type value that is valid for your workstation, and that you are running your program on the expected type of workstation.</p>
-5	<p>Bad string addresses found writing choice data record in routine ****</p> <p>User Action: There is an illegal array of string pointers passed to the choice data record in the specified routine. Make sure you properly initialized the arrays containing string addresses and string lengths. Also, make sure you have declared a buffer to hold choice strings, and that your string address array contains addresses of the elements in your choice string array.</p>
-6	<p>Echo area is too narrow for data in routine ****</p> <p>User Action: The specified input echo area minimum and maximum X values are too close in proximity. Make sure you did not swap X and Y values, and that your specified X values are a greater distance from each other.</p>
-7	<p>Maximum number of representable choices exceeded in routine ****</p> <p>User Action: The number of requested choices is too large for the workstation type. You can use INQUIRE DEFAULT CHOICE DEVICE DATA to obtain the maximum choices available for your workstations, and then break your menu into two smaller menus.</p>
-8	<p>Echo area is too short for data in routine ****</p> <p>User Action: The specified input echo area minimum and maximum Y values are too close in proximity. Make sure you did not swap X and Y values, and that your specified Y values are a greater distance from each other.</p>
-9	<p>Binary format and integer number representation not supported in this implementation of GKS in routine ****</p> <p>User Action: You opened a metafile of an incompatible type. Check the metafile type.</p>
-10	<p>Invalid value specified for ASF in routine ****</p> <p>User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GBUNDLED (0) or GINDIVIDUAL (1).</p>

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
-11	<p>Invalid value specified for fill area interior style in routine ****</p> <p>User Action: Make sure you passed one of the values GHOLLOW (0), GSOLID (1), GPATTERN(2), or GHATCH (3).</p>
-12	<p>Invalid value specified for horizontal component of text alignment in routine ****</p> <p>User Action: Make sure you passed one of the values GAH_NORMAL (0), GAH_LEFT (1), GAH_CENTRE (2), or GAH_RIGHT (3).</p>
-13	<p>Invalid value specified for vertical component of text alignment in routine ****</p> <p>User Action: Make sure you passed one of the values GAV_NORMAL (0), GAV_TOP (1), GAV_CAP (2), GAV_HALF (3), GAV_BASE (4), or GAV_BOTTOM (5).</p>
-14	<p>Invalid value specified for text precision in routine ****</p> <p>User Action: Make sure you passed one of the values GP_STRING (0), GP_CHAR (1), or GP_STROKE (2).</p>
-15	<p>Invalid value specified for text path in routine ****</p> <p>User Action: Make sure you passed one of the values GTP_RIGHT (0), GTP_LEFT (1), GTP_UP (2), or GTP_DOWN (3).</p>
-16	<p>Echo switch is invalid in routine ****</p> <p>User Action: Make sure you passed the one of the values GNOECHO (0) or GECHO (1). Also, if you used an inquiry function to obtain the echo switch, check to see that the arguments to the inquiry function are specified in the correct order.</p>
-17	<p>Inquired device values not set or realized in routine ****</p> <p>User Action: Check the value type argument to make sure it is either GSET (0) or GREALIZED (1).</p>
-18	<p>The following error occurred when GKS was interpreting an item ****</p> <p>User Action: An error occurred while interpreting a metafile item. DEC GKS follows this error message with another message that signals the appropriate action.</p>
-19	<p>Invalid error status parameter specified in routine ****</p> <p>User Action: You passed an illegal error code to ERROR LOGGING. Make sure the error code passed to ERROR LOGGING is one of the codes described in this appendix.</p>
-20	<p>GKS not in proper state: GKS in the ERROR state in routine ****</p> <p>User Action: You attempted to execute a DEC GKS function other than an error-handling or inquiry function. Remove all calls to DEC GKS functions, other than inquiry and error-handling function calls, from your error-handling code.</p>

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-21	Function is not supported in this level of GKS in routine **** User Action: Remove the call to the unsupported function.
-22	Invalid segment transformation in routine **** User Action: Check your calls to EVALUATE TRANSFORMATION MATRIX and to ACCUMULATE TRANSFORMATION MATRIX to make sure you passed valid transformation components. Also, make sure you specified a transformation matrix to SET SEGMENT TRANSFORMATION or to INSERT SEGMENT.
-23	Invalid value specified for clipping flag in routine **** User Action: Make sure you passed either the value GNOCLIP (0) or GCLIP (1).
-24	Invalid value specified for viewport priority flag in routine **** User Action: Make sure you passed either the value GHIGHER (0) or GLOWER (1).
-25	Invalid value specified for update workstation flag in routine **** User Action: Make sure you passed either the value GPERFORM (0) or GPOSTPONE (1).
-26	Invalid value specified for deferral mode in routine **** User Action: Make sure you passed one of the values GASAP (0), GBNIG (1), GBNIL (2), or GASTI (3).
-27	Invalid value specified for regeneration mode in routine **** User Action: Make sure you passed either the value GSUPPRESSED (0) or GALLOWED (1).
-28	Invalid value specified for expansion factor in routine **** User Action: Check to make sure you specified a real number value greater than the value 0.0. The value 1.0 causes no expansion.
-29	Invalid data record size for specified prompt and echo type in routine **** User Action: Check to make sure you specified a data record of the correct size as determined by your chosen prompt and echo type.
-30	Cannot load workstation handler: error during image activation in routine **** User Action: DEC GKS could not activate your workstation handler's shareable image. Make sure your workstation handler is a valid, shareable image.
-31	Cannot load graphics handler: invalid DFT in routine **** User Action: Your device function tables are incompatible. You need to build your device function table again using the appropriate macro. For more information, see the <i>Building a Device Handler System for DEC GKS and DEC PHIGS</i> .

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-32	Font file for stroke precision text not found or unusable in routine **** User Action: DEC GKS could not activate your workstation handler's shareable image. See the appropriate device-specific chapter in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> to determine if the specified font is supported on your device. If you are not using a DEC GKS supported graphics handler, make sure your handler defines the proper environment options, and they reference a valid file.
-33	Array descriptor is not acceptable in routine **** User Action: An item in the array descriptor is either invalid or inconsistent. Make sure you have passed the array by descriptor and that you fill the descriptor with valid values. If you have, and you use an inquiry function to initialize the array variable, make sure all the arguments are specified to the inquiry function in the correct order. Also, if the array is passed to the CELL ARRAY function, make sure you have declared a two-dimensional array.
-34	String length less than or equal to 0 in routine **** User Action: You specified an invalid character string. Check the declaration, definition, or assignment statements involving the character variable.
-35	Kernel has detected an unexpected error from a device handler in routine **** User Action: The device handler encountered an error. DEC GKS follows this error message with another message that signals the appropriate action.
-36	Cannot load device handler: error during image activation in routine **** User Action: DEC GKS could not activate your device handler's shareable image. Make sure your device handler is a valid, shareable image. This error message is specific to handlers that affect a device (VAXstations) as opposed to a graphics language (PostScript).
-37	Error in device handler during event flag allocation in routine **** User Action: A graphics handler was unable to acquire all of its needed event flags. The application must release event flags for use by the graphics handler.
-38	Error in device handler, cannot allocate device in routine **** User Action: You used your graphics handler with an invalid physical device. Make sure you use the proper physical device or that you specify the correct workstation type value to OPEN WORKSTATION.
-39	Descriptor is not acceptable in routine **** User Action: Make sure you have passed the variable by descriptor. If you have and you use an inquiry function to initialize the variable, make sure all the arguments are specified to the inquiry function in the correct order.

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-40	Illegal device pointer in routine **** User Action: Check your handler code for null pointers or otherwise invalid pointers.
-41	Driver handler WDT is invalid in routine **** User Action: You illegally defined a workstation description table entry. Check your workstation description table definitions for your graphics handler.
-42	Logical name for the list of workstation types, GKS\$LIST_TYPES, could not be translated in routine **** User Action: You improperly defined the logical name. Make sure the translation of GKS\$LIST_TYPES is as expected.
-43	VAXstation Workstation Software is not present, workstation type is invalid in routine **** User Action: Check to make sure either that you specify the correct workstation type when opening a workstation other than a VAXstation, or that you passed a correct workstation type value to one of the workstation description table or state list inquiry functions. If you are working on a MicroVAX, make sure you install the VAXstation Workstation Software.
-44	Error trying to save or restore VT340 color map in routine **** User Action: Submit an SPR.
-45	Unable to decompose fill area or fill area set in routine **** User Action: Simplify the fill area by breaking it up into smaller fill areas until the error does not occur.
-46	No default connection identifier for specified workstation type in routine **** User Action: Define the environment option to be a valid GKS connection identifier.
-90	Internal GKS error: Bad memory address freed in routine **** User Action: DEC GKS memory data structures were corrupted. Submit an SPR.
-91	Internal GKS error: Invalid function pointer parameter in error handler in routine **** User Action: A DEC GKS internal data structure was corrupted. Submit an SPR.
-92	Internal GKS error: Insufficient virtual memory in routine **** User Action: DEC GKS was unable to allocate enough virtual memory. Check to make sure the problem is not caused by storing too much in segment storage or by defining a very large cell array. If you cannot reduce storage by checking segments and cell arrays, submit an SPR.
-93	Internal GKS error: Prompt and echo type not supported in routine ****

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
	User Action: Internal error. Submit an SPR.
-94	Internal GKS error: Corrupted segment memory in routine **** User Action: Internal error. Submit an SPR.
-95	Internal GKS error: Negative size passed to allocate memory in routine **** User Action: An invalid size was passed to the DEC GKS memory allocation routines. If you generate this error using a user-written graphics handler, make sure the value of the local storage area is a valid value.
-96	Internal GKS error: Illegal number of points to device handler for rectangular polygon in routine **** User Action: Internal error. Submit an SPR.
-97	Internal GKS error: Insufficient buffer size for translated logical name in routine **** User Action: Make sure none of the environment options is a string greater than 255 characters. If you cannot locate an error, submit an SPR.
-98	Internal GKS error: Too many translations of logical name in routine **** User Action: You may have recursively defined a logical name. Check the currently defined logical names to see if all are properly defined. If you cannot locate an error, submit an SPR.
-99	Internal GKS error: Unable to reduce number of points in fill area to requested limit in routine **** User Action: Internal error. Submit an SPR.
-100	Internal GKS error: Device handler received unexpected input access in routine **** User Action: Internal error. Submit an SPR.
-150	Edge index is less than zero in routine **** User Action: Make sure the edge index passed in the escape function GESC_ESEI is valid.
-151	Edge width scale factor is less than zero **** User Action: Make sure the edge width scale factor passed in the escape Set Edge Index (GESC_ESEW) is valid.
-154	A representation for the specified edge index has not been predefined on this workstation in routine **** User Action: Check the index of the edge being inquired about to make sure it is predefined.

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

<b>Error</b>	<b>Error Message</b>
-155	Display speed is less than zero in routine **** User Action: Pass a positive real value to the escape Set Speed (GESC_ESP).
-156	Loudness is outside range [0,1] in routine **** User Action: Pass a valid value to the escape Beep (GESC_EB) .
-157	Duration is less than zero in routine **** User Action: Make sure your duration value is greater than or equal to 0.
-158	GDP primitive is not defined by the supplied data in routine **** User Action: DEC GKS is unable to form the desired primitive. See the error message listing in the description of the GDP that generated the error (in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> ). This listing gives specific information concerning the primitive you attempted to draw.
-159	Arc type is invalid in routine **** User Action: See the error message listing in the description of the GDP that generated the error (in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> ). This listing gives specific information concerning the primitive you attempted to draw.
-160	Insufficient space in escape output data record arrays in routine **** User Action: You passed addresses of arrays that were too small to contain the data to be written to them. Pass addresses of larger array buffers in the last four components of the escape data record.
-161	Specified bounding box is too small in routine **** User Action: You specified text attributes that were too large to fill the text in the bounding box (the extent rectangle). Use a larger bounding box, or reduce the text height or the character expansion factor.
-162	Edge index is invalid in routine **** User Action: Make sure you use a valid edge index.
-163	Specified edge type is not supported on this workstation in routine **** User Action: Make sure you use a valid edge type.
-300	Invalid value specified for highlighting in routine **** User Action: Make sure you specify either GNORMAL (0) or GHIGHLIGHTED (1).
-301	Invalid value specified for visibility in routine **** User Action: Make sure you specify either GVISIBLE (0) or GINVISIBLE (1).

(continued on next page)



## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
-302	<p>Invalid value specified for detectability in routine ****</p> <p>User Action: Make sure you specify either GUNDETECTABLE (0) or GDETECTABLE (1).</p>
-303	<p>Input device cannot be activated due to conflict with another input device that is currently active in routine ****</p> <p>User Action: The input device echo area overlaps the input device echo area on the display. Because these two input devices could overwrite each other's echo on the display, they cannot be active at the same time. Typically, on a nonwindowing system output device, a choice, string, or valuator echo area cannot overlap the input echo of a locator, stroke, or pick device. On a windowing system, this error should not occur, because the input echos are drawn in separate windows. You should change the input echo areas and make sure that they do not overlap.</p>
-304	<p>Cannot set input device echo on due to conflict with other input devices active in the same echo area in routine ****</p> <p>User Action: Two or more input devices are already active, but one of them is in NOECHO mode and the application is attempting to put it into ECHO mode. You should change the input echo areas and make sure they do not overlap.</p>
-305	<p>Error parsing GKS\$[wstype]_INPUT_DEVICES logical name due to syntax error in routine ****</p> <p>User Action: Check the definition of the logical name with the syntax description in the Tektronix 41xx chapter in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i>.</p>
-306	<p>The definition of GKS\$HPGL_THRESHOLD is invalid (contains nonnumeric values in routine) ****</p> <p>User Action: Check the definition of GKS\$HPGL_THRESHOLD and redefine to range 0 to 1023.</p>
-450	<p>Valuator device could not be used—dials are not available on the PCM device in routine ****</p> <p>User Action: The hardware dial box is not available, either because it is not present or because it was not hooked up correctly. Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware dials will not be used. See the chapter on PCM button and dials in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i>.</p>
-451	<p>Choice device could not be used—buttons are not available on the PCM device in routine ****</p> <p>User Action: The hardware button box is not available, either because it is not present or because it was not hooked up correctly. Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware buttons will not be used. See the chapter on PCM button and dials in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i>.</p>

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
-452	PCM initialization failed—hardware dials and buttons not available in routine **** User Action: The PCM server is not running. It has either not been started or the PCM dial and button box is not connected. Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware buttons will not be used. See the chapter on PCM button and dials in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> .
-453	PCM configuration failed—valuator or choice device not initialized in routine **** User Action: Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware buttons will not be used. See the chapter on PCM button and dials in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> .
-1517	Error trying to open UID database User Action: Verify that the UID files gfx_decw.uid and gfx_decw_xx.uid (where xx is the language code) exist in either the specified system area or user path.
-1518	Error trying to fetch from UID database User Action: Verify that the name used in the DwtFetchWidget or DwtFetchWidgetOverride command matches that in the diagrams (in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> ), and has not been altered while editing the UIL files.
-2000	Illegal call to routine **** while GKS is in an error state User Action: Make sure GKS is not called while it is already in an error state. This may happen if you call GKS functions from the GKS error handler provided by the application.
-2001	Invalid value specified for ASF in routine **** User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GBUNDLED (0) or GINDIVIDUAL (1).
-2002	Invalid clipping indicator in routine **** User Action: Make sure you passed either the value GNOCLIP (0) or GCLIP (1).
-2003	Invalid normalization transformation priority in routine **** User Action: Make sure the arguments are specified in the correct order and that the normalization priority value is either GHIGHER (0) or GLOWER (1).
-2004	Invalid view index specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified view index is in the permitted range (greater than or equal to 0 and less than the maximum number of view representations).
-2005	Invalid metafile format found in routine ****

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
	User Action: Make sure the input file you are trying to interpret is a metafile that was generated by DEC GKS.
-2006	Null text direction vector found in routine **** User Action: Make sure the arguments are specified in the correct order and that the text direction vector is not {0.0, 0.0, 0.0} (null vector).
-2007	Invalid text path specified in routine **** User Action: Make sure you passed one of the values GTP_RIGHT (0), GTP_LEFT (1), GTP_UP (2), or GTP_DOWN (3).
-2008	Invalid text character alignment specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the text alignment specified is a permitted value (GAH_NORMAL (0), GAH_LEFT (1), GAH_CENTRE (2), or GAH_RIGHT (3) for the horizontal component, and GAV_NORMAL (0), GAV_TOP (1), GAV_CAP (2), GAV_HALF (3), GAV_BASE (4), or GAV_BOTTOM (5) for the vertical component).
-2009	Invalid fill area interior style specified in routine **** User Action: Make sure you passed one of the values GHOLLOW (0), GSOLID (1), GPATTERN(2), or GHATCH (3).
-2010	Invalid edge flag specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified edge flag is a permitted value (GEDGE_OFF (0) or GEDGE_ON (1)).
-2011	Invalid aspect source flag specified in routine **** User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GBUNDLED (0) or GINDIVIDUAL (1).
-2012	Null pattern reference vector found in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified pattern reference vector is not {0.0, 0.0, 0.0} (null vector).
-2013	Start value for inquiry is negative or too large in routine **** User Action: Make sure the arguments are specified in the correct order and that the start value is in the permitted range (greater than or equal to 0 and less than the maximum number).
-2014	Invalid clear control flag specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the clear control flag is a permitted value (GCONDITIONALLY (0) or GALWAYS (1)).

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-2015	Invalid update control flag specified in routine **** User Action: Make sure you passed one of the values GASAP (0), GBNIG (1), GBNIL (2), or GASTI (3).
-2016	Invalid deferral mode specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified deferral mode is a permitted value (GSUPPRESSED (0) or GALLOWED (1)).
-2017	Invalid implicit regeneration mode specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified implicit regeneration mode is a permitted value (GSUPPRESSED (0) or GALLOWED (1)).
-2018	Invalid view transformation priority in routine **** User Action: Make sure the arguments are specified in the correct order and that the view transformation priority is a permitted value (GHIGHER (0) or GLOWER (1)).
-2019	Invalid text precision specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified text precision is a permitted value (GP_STRING (0), GP_CHAR (1), or GP_STROKE (2)).
-2020	Invalid coordinate switch argument in routine **** User Action: Make sure the arguments are in the correct order and that the coordinate switch argument is a permitted value (GWC (0) or GNDC (1)).
-2021	Invalid projection type argument in routine **** User Action: Make sure the arguments are specified in the correct order and that the projection type argument is either GPARALLEL (0) or GPERSPECTIVE (1).
-2022	Front plane is too near back plane for this viewport in routine **** User Action: Make the distance between the front and back planes is larger.
-2023	Equal window z limits not allowed with unequal viewport z limits in routine **** User Action: Either make the window Z limits unequal, or make the viewport Z limits equal.
-2024	Invalid input device mode specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified input device mode is one of the following values: GREQUEST (0), GSAMPLE (1), or GEVENT (2).

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
-2025	<p>Invalid input device echo state specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the input device echo state is either GNOECHO (0) or GECHO (1).</p>
-2026	<p>Invalid polyline/fill area control flag in data record in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the polyline /fill area control flag is a permitted value (GPF_POLYLINE (0) or GPF_FILLAREA (1)).</p>
-2027	<p>Invalid attribute control flag in data record in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the attribute control flag in the data record is a permitted value (GCURRENT (0) or GSPECIFIED (1)).</p>
-2028	<p>Invalid data record size for specified prompt and echo type in routine ****</p> <p>User Action: Check the specified data record size against the data record required for the specified PET.</p>
-2029	<p>Invalid normalization transformation number specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified normalization transformation number is a permitted value.</p>
-2030	<p>NDC position(s) outside NDC unit cube in routine ****</p> <p>User Action: Make sure the specified NDC values are in the range 0.0 to 1.0.</p>
-2031	<p>Invalid input device class specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified input device class is one of the following values: GNCLASS (0), GLOCATOR (1), GSTROKE (2), GVALUATOR (3), GCHOICE (4), GPICK (5), GSTRING (6), or GVIEWPORT (7).</p>
-2032	<p>Invalid visibility flag specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified visibility flag is either GVISIBLE (0) or GINVISIBLE (1).</p>
-2033	<p>Invalid highlighting flag specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified highlighting flag is either GNORMAL (0) or GHIGHLIGHTED (1).</p>
-2034	<p>Invalid detectability flag specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified detectability flag is either GUNDETECTABLE (0) or GDETECTABLE (1).</p>

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-2036	Invalid matrix specified, matrix has a zero fourth row in routine **** User Action: Make sure the specified matrix does not have a zero fourth row.
-2037	Warning, the specified view matrix is not invertible in routine **** User Action: Specify a matrix that is invertible.
-2038	Warning, a 3D item cannot be written into GKSM (2D metafile output) in routine **** User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
-2039	Warning, the normalization transformation is 3D and cannot be used with GKSM (2D metafile output) in routine **** User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
-2040	Warning, the segment transformation is 3D and cannot be used with GKSM (2D metafile output) in routine **** User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
-2041	Warning, 3D inquiries are invalid with GKSM (2D metafile output) in routine **** User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
-2500	Insufficient buffer size for translated logical name in routine **** User Action: Make sure none of the environment options equals a string greater than 255 characters. If you cannot locate an error, submit an SPR.
-2501	Too many translations of logical name in routine **** User Action: You may have recursively defined a logical name. Check the currently defined logical names to see if all are properly defined. If you cannot locate an error, submit an SPR.
-2502	System error during logical name translation in routine **** User Action: Make sure the environment option has a valid assignment string or other environment option. The maximum length of assignment is 255 characters.
-2503	Error closing the error logging file in routine **** User Action: Make sure there is enough free disk space available.

(continued on next page)

## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
-2504	Image activation error during load of workstation handler in routine **** User Action: Make sure your workstation handler is a valid, shareable image. Reinstall DEC GKS.
-2505	Invalid WFT detected during load of workstation handler in routine **** User Action: You need to build your device function table again using the appropriate macro. For more information, see the <i>Building a Device Handler System for DEC GKS and DEC PHIGS</i> .
-2506	Invalid string descriptor found in routine **** User Action: Make sure you specify a valid string descriptor. See the file <code>gks_descrip.h</code> in the system library.
-2507	Insufficient buffer space to convert string descriptor in routine **** User Action: Submit an SPR.
-2508	Image activation error during load of device handler in routine **** User Action: Make sure your device handler is a valid, shareable image. This error message is specific to handlers that affect a device (VAXstations) as opposed to a graphics language (PostScript).
-2509	Invalid DFT detected during load of device handler in routine **** User Action: You need to build your device function table again using the appropriate macro. For more information, see the <i>Building a Device Handler System for DEC GKS and DEC PHIGS</i> .
-2510	System error during logical name creation in routine **** User Action: GKS is not able to create a process table logical. See your system manager.
-2511	Invalid array descriptor in routine **** User Action: Make sure you have passed the array by descriptor and that you fill the descriptor with valid values. If you have, and you use an inquiry function to initialize the array variable, make sure that all the arguments are specified to the inquiry function in the correct order. Also, if the array is passed to the CELL ARRAY function, make sure you have declared a two-dimensional array.
-2512	Invalid representation of integer in translated string in routine **** User Action: Make sure the environment options that require an integer define a valid integer (not another data type).
-2513	Error translating workstation type list in routine **** User Action: Make sure the translation of <code>GKS\$LIST_TYPES</code> is as expected.

(continued on next page)

## DEC GKS Error Messages

### A.13 Implementation-Specific Errors

Table A-13 (Cont.) Implementation-Specific Errors

Error	Error Message
-2514	Invalid device handler WDT detected in routine **** User Action: Check your workstation description table definitions for your graphics handler.
-2515	Device handler WDT has at least one bundle table of zero size in routine **** User Action: Check your workstation description table. All bundle tables must be defined in the workstation description table.
-2517	Invalid function name descriptor found in routine **** User Action: Submit an SPR.
-2518	Create operation failed on GKS error file in routine **** User Action: Check the validity of the error file specification and check the available disk space.
-2519	Open operation failed on GKS error file in routine **** User Action: Check the validity of the error file specification and check the available disk space.
-2520	Close operation failed on GKS error file in routine **** User Action: Check the validity of the error file specification and check the available disk space.
-2521	Delete operation failed on empty GKS error file in routine **** User Action: Check the validity of the error file specification, file references, file protection, and file owner.
-2522	Font file not found or unusable in routine **** User Action: See the appropriate device-specific chapter in the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> to determine if the specified font is supported on your device. If you are not using a DEC GKS supported graphics handler, make sure your handler defines the proper logical names, and that the logicals reference a valid file.
-2523	String length less than or equal to 0 in routine **** User Action: Check the declaration, definition, or assignment statements involving the character variable.
-2525	Invalid function identifier found in routine **** User Action: Submit an SPR.
-2528	Internal error converting time to binary form in routine ****

(continued on next page)



## DEC GKS Error Messages A.13 Implementation-Specific Errors

**Table A-13 (Cont.) Implementation-Specific Errors**

Error	Error Message
	User Action: Submit an SPR.
-2529	System error setting interval timer in routine **** User Action: Submit an SPR.
-2530	System error waiting for event flag in routine **** User Action: Submit an SPR.
-2531	System error canceling interval timer in routine **** User Action: Submit an SPR.
-2532	System error setting event flag in routine **** User Action: Submit an SPR.
-2533	System error getting event flag in routine **** User Action: Submit an SPR.
-2534	System error freeing event flag in routine **** User Action: Submit an SPR.
-2535	System error setting timer event handler in routine **** User Action: Submit an SPR.
-2536	System error resetting timer event handler in routine **** User Action: Submit an SPR.
-2537	Nonfatal input manager error detected in routine **** User Action: Submit an SPR.
-2538	Insufficient buffer space available in routine **** User Action: Submit an SPR.
-2539	Device handler error detected in routine **** User Action: DEC GKS follows this error message with another message that signals the appropriate action.

# DEC GKS Error Messages

## A.14 Fatal Errors

### A.14 Fatal Errors

Table A–14 lists the fatal DEC GKS and system errors.

**Table A–14 Fatal Errors**

Error	Error Message
–3000	Internal data error detected in routine **** User Action: Submit an SPR.
–3001	Error looping condition forced emergency close in routine **** User Action: Submit an SPR.
–3002	Fatal error condition forced emergency close in routine **** User Action: Submit an SPR.
–3003	Illegal call to native interface, bypassing FORTRAN binding, detected in routine **** User Action: Submit an SPR.
–3004	Illegal call to native interface, bypassing C binding, detected in routine **** User Action: Submit an SPR.
–3500	Memory zone allocation error, insufficient virtual memory in routine **** User Action: Increase the system page or swap file.
–3501	Memory zone deallocation error in routine **** User Action: Submit an SPR.
–3502	Memory allocation error, insufficient virtual memory in routine **** User Action: Increase the system page or swap file.
–3503	Memory deallocation error in routine **** User Action: Submit an SPR.
–3504	Memory reallocation error in routine **** User Action: Submit an SPR.
–3505	Float array allocation error, insufficient virtual memory in routine **** User Action: Submit an SPR.
–3506	Float array deallocation error in routine **** User Action: Submit an SPR.

(continued on next page)

Table A-14 (Cont.) Fatal Errors

---

Error	Error Message
-3508	Workstation handler error in routine **** User Action: Submit an SPR.
-3509	Fatal input manager error detected in routine **** User Action: Submit an SPR.
-3510	Inadequate size for the internal array containing the list of prompt and echo types in routine **** User Action: Submit an SPR.
IVP	DEC GKS Installation Verification Procedure failed User Action: See the installation guide for information on what to do if the IVP fails.

---



## **Constants**

**Insert tabbed divider here. Then discard this sheet.**



# B

## Constants

This appendix lists the DEC GKS constants defined for the C binding language interface, their values, and a short description of each. Using constants in your DEC GKS programs makes your code easier to read.

To use constants in your program, you must include a definitions file in your code. The language definition file for the C binding is gks.h.

### B.1 C Constants

Table B-1 lists the C constant names, their values, and their descriptions.

**Table B-1 C Constants**

Constant	Value	Description
<b>Aspect Source Flags:</b>		
GBUNDLED	0	Bundled
GINDIVIDUAL	1	Individual
<b>Attribute Control Flags:</b>		
GCURRENT	0	Input data record current values
GSPECIFIED	1	Input data record specified values
<b>Choice Input Prompt Flags:</b>		
GPROFF	0	Prompt off
GPRON	1	Prompt on
<b>Choice Status Types:</b>		
GC_OK	0	Input obtained
GC_NOCHOICE	1	Input is NOCHOICE
GC_NONE	2	No input obtained
<b>Clear Screen States:</b>		
GCONDITIONALLY	0	Clear conditionally
GALWAYS	1	Clear always

(continued on next page)

## Constants

### B.1 C Constants

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>Clipping Flags:</b>		
GCLIP	0	Clipping on
GNOCLIP	1	Clipping off
<b>Color Availability Flags:</b>		
GCOLOUR	0	Color
GMONOCHROME	1	Monochrome
<b>Color Models:</b>		
GCM_RGB	1	Red, green, blue color model
GCM_CIE	2	Commission Internationale de l'Eclairage color model
GCM_HSV	3	Hue, saturation, and value color model
GCM_HLS	4	Hue, lightness, and saturation color model
<b>Coordinate Switch:</b>		
GWC	0	World coordinates
GNDC	1	Normalized device coordinates
<b>Default Connection Identifier:</b>		
GWC_DEF	0	Default workstation connection identifier
GCONID_DEFAULT	0	Default workstation connection identifier
<b>Deferral Modes:</b>		
GASAP	0	As soon as possible
GBNIG	1	Before the next global interaction
GBNIL	2	Before the next local interaction
GASTI	3	At some time
<b>Detectability Flags:</b>		
GUNDETECTABLE	0	Set to undetectable
GDETECTABLE	1	Set to detectable
<b>Device Coordinate Units:</b>		
GDC_METRES	0	Meters
GDC_OTHER	1	Other units

(continued on next page)



**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>Display Surface States:</b>		
GEMPTY	0	Display surface empty
GNOTEMPTY	1	Display surface not empty
<b>Dynamic Modification States:</b>		
GIRG	0	Implicit regeneration necessary
GIMM	1	Immediate
<b>Echo States:</b>		
GECHO	0	Echo enabled
GNOECHO	1	Echo disabled
<b>Edge Flags:</b>		
GEDGE_OFF	0	Edge flag off
GEDGE_ON	1	Edge flag on
<b>Edge Types:</b>		
GED_SOLID	1	Edge type solid
GED_DASHED	2	Edge type dashed
GED_DOTTED	3	Edge type dotted
GED_DASHDOT	4	Edge type dash-dotted
GED_DASH_2_DOT	–1	Edge type double-dashed dotted
GED_DASH_3_DOT	–2	Edge type triple-dashed dotted
GED_LONG_DASH	–3	Edge type long-dashed
GED_LONG_SHORT_DASH	–4	Edge type long-short-dashed
GED_SPACED_DASH	–5	Edge type spaced-dashed
GED_SPACED_DOT	–6	Edge type spaced-dotted
GED_DOUBLE_DOT	–7	Edge type double dotted
GED_TRIPLE_DOT	–8	Edge type triple dotted
<b>Error Handling Modes:</b>		
GER_OFF	0	Error handling off
GER_ON	1	Error handling on

(continued on next page)

## Constants

### B.1 C Constants

**Table B-1 (Cont.) C Constants**

Constant	Value	Description
<b>Escape Function Identifiers:</b>		
ESC_ESP	-100	Set Display Speed
ESC_EP	-101	Generate Hardcopy of Workstation Surface
ESC_EB	-103	Beep
ESC_EPOPW	-106	Pop Workstation
ESC_EPSHW	-107	Push Workstation
ESC_ESEHM	-108	Set Error Handling Mode
ESC_ESVE	-109	Set Viewport Event
ESC_EAWC	-110	Associates Workstation Type and Connection ID
ESC_ESCL	-111	Software Clipping
ESC_ESWM	-150	Set Writing Mode
ESC_ESLC	-151	Set Line Cap Style
ESC_ESLJ	-152	Set Line Join Style
ESC_ESEC	-153	Set Edge Control Flag
ESC_ESET	-154	Set Edge Type
ESC_ESEW	-155	Set Edge Width Scale Factor
ESC_ESECI	-156	Set Edge Color Index
ESC_ESEI	-157	Set Edge Index
ESC_ESEA	-158	Set Edge Aspect Source Flag
ESC_EBTB	-160	Begin Transformation Block
ESC_EETB	-161	End Transformation Block
ESC_ESSHM	-162	Set Segment Highlighting Method
ESC_ESHM	-163	Set Highlighting Method
ESC_BT3	-164	Begin Transformation Block 3
ESC_ESER	-200	Set Edge Representation
ESC_ESWT	-202	Set Window Title
ESC_ESRS	-203	Set Reset String
ESC_ESCS	-204	Set Cancel String
ESC_ESES	-205	Set Enter String
ESC_ESIB	-206	Set Icon Bit Maps
ESC_EIWM	-251	Inquire Current Writing Mode
ESC_EILC	-252	Inquire Current Line Cap Style
ESC_EILJ	-253	Inquire Current Line Join Style
ESC_EIEA	-254	Inquire Current Edge Attributes
ESC_EIVD	-255	Inquire Viewport Data
ESC_EIS	-300	Inquire Current Display Speed
ESC_EILEI	-302	Inquire List of Edge Indexes

(continued on next page)

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>Escape Function Identifiers:</b>		
ESC_EISE	–303	Inquire Segment Extent
ESC_EIWID	–304	Inquire Window Identifiers
ESC_EISHM	–305	Inquire Segment Highlighting Method
ESC_EIHM	–306	Inquire Highlighting Method
ESC_EIPBI	–307	Inquire Pasteboard Identifier
ESC_EIMBI	–308	Inquire Menu Bar Identifier
ESC_EISHI	–309	Inquire Shell Identifier
ESC_EILE	–350	Inquire List of Available Escapes
ESC_EIDS	–351	Inquire Default Display Speed
ESC_EILCJ	–352	Inquire line Cap and Join Facilities
ESC_EIEF	–354	Inquire Edge Facilities
ESC_EIPER	–355	Inquire Predefined Edge Representation
ESC_EIMEB	–356	Inquire Maximum Number of Edge Bundles
ESC_EILH	–358	Inquire List of Highlighting Methods
ESC_IER	–359	Inquire Edge Representation
ESC_EMNW	–400	Evaluate NDC Mapping of a WC Point
ESC_EMDN	–401	Evaluate DC Mapping of an NDC Point
ESC_EMWN	–402	Evaluate WC Mapping of an NDC Point
ESC_EMND	–403	Evaluate NDC Mapping of a DC Point
ESC_EIGEX	–404	Inquire Extent of a GDP
ESC_ESDB	–500	Set Double Buffering
ESC_ESBPM	–501	Set Background Pixmap
ESC_EIDBM	–502	Inquire Double Buffer Pixmap
ESC_EIBGM	–503	Inquire Background Pixmap
<b>Fill Area Control Flags:</b>		
GPF_POLYLINE	0	Data record polyline control flag
GPF_FILLAREA	1	Data record fill area control flag
<b>Fill Area Interior Styles:</b>		
GHOLLOW	0	Interior style hollow
GSOLID	1	Interior style solid
GPATTERN	2	Interior style pattern
GHATCH	3	Interior style hatched

(continued on next page)

## Constants

### B.1 C Constants

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>GDP Bundle Types:</b>		
GPOLYLINE	0	GDP polyline bundle
GPOLYMARKER	1	GDP polymarker bundle
GTEXT	2	GDP text bundle
GFILLAREA	3	GDP fill area bundle
GEDGE	4	GDP edge bundle
<b>GDP Graphics Primitives:</b>		
GDP_DISP	–100	Disjoint Polyline
GDP_CCP	–101	Circle: Center and Point on Circumference
GDP_C3P	–102	Circle: Three Points on Circumference
GDP_CCR	–103	Circle: Center and Radius
GDP_C2PR	–104	Circle: Two Points on Circumference, and Radius
GDP_AC2P	–106	Arc: Center and Two Points on Arc
GDP_A3P	–107	Arc: Three Points on Circumference
GDP_ACVR	–108	Arc: Center, Two Vectors, and a Radius
GDP_A2PR	–109	Arc: Two Points on Arc and Radius
GDP_ACPA	–110	Arc: Center, Starting Point, and Angle
GDP_ECA	–111	Ellipse: Center, and Two Axis Vectors
GDP_EFP	–113	Ellipse: Focal Points and Point on Circumference
GDP_EACA	–114	Elliptic Arc: Center, Two Axis Vectors, and Two Vectors
GDP_EAFP	–116	Elliptic Arc: Focal Points and Two Points on Circumference
GDP_R2P	–125	Rectangle: Two Corners
GDP_FAS	–332	Fill Area Set
GDP_FCCP	–333	Filled Circle: Center and Point on Circumference
GDP_FC3P	–334	Filled Circle: Three Points on Circumference
GDP_FCCR	–335	Filled Circle: Center and Radius
GDP_FCPR	–336	Filled Circle: Two Points on Circumference, and Radius
GDP_FACP	–338	Fill Arc: Center and Two Points on Arc
GDP_FA3P	–339	Filled Arc: Three points on Circumference
GDP_FACV	–340	Filled Arc: Center, Two Vectors, and a Radius
GDP_FAPR	–341	Filled Arc: Two Points on Arc, and Radius

(continued on next page)

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>GDP Graphics Primitives:</b>		
GDP_FACA	–342	Filled Arc: Center, Starting Point, and Angle
GDP_FECA	–343	Filled Ellipse: Center and Two Axis Vectors
GDP_FEFP	–345	Filled Ellipse: Focal Points and Point on Circumference
GDP_FEACA	–346	Filled Elliptic Arc: Center, Two Axis Vectors, and Two Vectors
GDP_FEAF	–348	Filled Elliptic Arc: Focal Points and Two Points on Circumference
GDP_FR2P	–349	Filled Rectangle: Two Corners
GDP_GIA	–400	Packed Cell Array
<b>GKS Level Types:</b>		
GLMA	–3	Minimal output, no input
GLMB	–2	Minimal output, request input
GLMC	–1	Minimal output, full input
GL0A	0	All primitives and attributes, no input
GL0B	1	All primitives and attributes, request input
GL0C	2	All primitives and attributes, full input
GL1A	3	Basic segmentation with full output, no input
GL1B	4	Basic segmentation with full output, request input
GL1C	5	Basic segmentation with full output, full input
GL2A	6	Workstation-independent and segment storage, no input
GL2B	7	Workstation-independent and segment storage, request input
GL2C	8	Workstation-independent and segment storage, full input
<b>GKS Operating States:</b>		
GGKCL	0	GKS closed
GGKOP	1	GKS open
GWSOP	2	At least one workstation open
GWSAC	3	At least one workstation active
GSGOP	4	At least one segment open

(continued on next page)

## Constants

### B.1 C Constants

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>HLHSR Identifiers:</b>		
GHLHSR_ID_NONE	0	No HLHSR processing
GHLHSR_ID_PAINTER	1	Painters algorithm
<b>HLHSR Modes:</b>		
GHLHSR_MODE_NONE	0	No HLHSR processing
GHLHSR_MODE_PAINTER	1	Painters algorithm
<b>Highlighting Flags:</b>		
GNORMAL	0	Primitives are not highlighted
GHIGHLIGHTED	1	Primitives are highlighted
<b>Highlighting Methods:</b>		
GHL_MDEF	0	Highlighting with workstation-dependent default method
GHL_MCOM	1	Highlight with complement mode
GHL_MCOL	2	Highlight with color
GHL_MLIN	3	Highlight with extent line box
GHL_MFIL	4	Highlight with extent fill area
GHL_MDUA	5	Highlight with extent line box and fill area
<b>Implicit Regeneration States:</b>		
GSUPPRESSED	0	Implicit regeneration suppressed
GALLOWED	1	Implicit regeneration allowed
<b>Input Classes:</b>		
GNCLASS	0	No input class
GLOCATOR	1	Locator input class
GSTROKE	2	Stroke input class
GVALUATOR	3	Valuator input class
GCHOICE	4	Choice input class
GPICK	5	Pick input class
GSTRING	6	String input class
GVIEWPORT	7	Viewport input class

(continued on next page)

**Table B–1 (Cont.) C Constants**

<b>Constant</b>	<b>Value</b>	<b>Description</b>
<b>Input Mode Types:</b>		
GREQUEST	0	Request mode
GSAMPLE	1	Sample mode
GEVENT	2	Event mode
<b>Input Priority States:</b>		
GHIGHER	0	Relative input priority higher
GLOWER	1	Relative input priority lower
<b>Invalid Index Flags:</b>		
GABSENT	0	Invalid indexes absent
GPRESENT	1	Invalid indexes present
<b>Last Event Flags:</b>		
GLAST	0	Last event
GNOTLAST	1	More events
<b>Line Cap Styles:</b>		
GLC_BUT	2	Line cap type butted
GLC_RND	3	Line cap type rounded
GLC_SQR	4	Line cap type square
<b>Line Join Styles:</b>		
GLJ_MTR	2	Line join type mitre
GLJ_RND	3	Line join type round
GLJ_BVL	4	Line join type bevel
<b>Line Types (Standard):</b>		
GLN_SOLID	1	Solid line
GLN_DASHED	2	Dashed line
GLN_DOTTED	3	Dotted line
GLN_DASHDOT	4	Dashed-dotted line

(continued on next page)

## Constants

### B.1 C Constants

**Table B-1 (Cont.) C Constants**

Constant	Value	Description
<b>Line Types (DEC GKS Implementation Specific):</b>		
GLN_DASH_2_DOT	-1	Double-dashed dotted line
GLN_DASH_3_DOT	-2	Triple-dashed dotted line
GLN_LONG_DASH	-3	Long-dashed line
GLN_LONG_SHORT_DASH	-4	Long-short-dashed line
GLN_SPACED_DASH	-5	Spaced-dashed line
GLN_SPACED_DOT	-6	Spaced-dotted line
GLN_DOUBLE_DOT	-7	Double-dotted line
GLN_TRIPLE_DOT	-8	Triple-dotted line
<b>Marker Types (Standard):</b>		
GMK_POINT	1	Marker type dot
GMK_PLUS	2	Marker type plus
GMK_STAR	3	Marker type asterisk
GMK_O	4	Marker type circle
GMK_X	5	Marker type diagonal cross
<b>Marker Types (DEC GKS Implementation Specific):</b>		
GMK_SOLID_CIRCLE	-1	Marker type solid circle
GMK_TRIANGLE_UP	-2	Marker type hollow up triangle
GMK_SOLID_TRI_UP	-3	Marker type solid up triangle
GMK_TRIANGLE_DOWN	-4	Marker type hollow down triangle
GMK_SOLID_TRI_DOWN	-5	Marker type solid down triangle
GMK_SQUARE	-6	Marker type hollow square
GMK_SOLID_SQUARE	-7	Marker type solid square
GMK_BOWTIE	-8	Marker type hollow bow tie
GMK_SOLID_BOWTIE	-9	Marker type solid bow tie
GMK_HOURLASS	-10	Marker type hollow hourglass
GMK_SOLID_HGLASS	-11	Marker type solid hourglass
GMK_DIAMOND	-12	Marker type hollow diamond
GMK_SOLID_DIAMOND	-13	Marker type solid diamond
<b>Memory Size:</b>		
GDEFAULT_MEM_SIZE	-1	Default memory size
<b>New Frame Action Necessary States:</b>		
GNO	0	No new frame action on update
GYES	1	New frame action on update

(continued on next page)



**Table B–1 (Cont.) C Constants**

<b>Constant</b>	<b>Value</b>	<b>Description</b>
<b>Pick Status Types:</b>		
GP_OK	0	Input obtained
GP_NOPICK	1	Input is NOPICK
GP_NONE	2	No input obtained
<b>Projection Types:</b>		
GPARALLEL	0	Parallel projection
GPERSPECTIVE	1	Perspective projection
<b>Regeneration Flag States:</b>		
GPFORM	0	Implicit regeneration performed
GPOSTPONE	1	Implicit regeneration postponed
<b>Request Status Types:</b>		
GOK	0	Input obtained
GNONE	1	No input obtained
<b>Returned Type Values:</b>		
GSET	0	Returned value is set
GREALIZED	1	Returned value is realized
<b>Simultaneous Events Flags:</b>		
GNOMORE	0	No more simultaneously generated events
GMORE	1	More simultaneously generated events
<b>Text Horizontal Alignment Types:</b>		
GAH_NORMAL	0	Normal horizontal alignment
GAH_LEFT	1	Left horizontal alignment
GAH_CENTRE	2	Center horizontal alignment
GAH_RIGHT	3	Right horizontal alignment
<b>Text Path Types:</b>		
GTP_RIGHT	0	Text string reads from left to right.
GTP_LEFT	1	Text string reads from right to left.
GTP_UP	2	Text string reads from bottom to top.
GTP_DOWN	3	Text string reads from top to bottom.

(continued on next page)

## Constants

### B.1 C Constants

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>Text Precision Types:</b>		
GP_STRING	0	String precision. DEC GKS evaluates character height and width attributes only.
GP_CHAR	1	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
GP_STROKE	2	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
<b>Text Vertical Alignment Types:</b>		
GAV_NORMAL	0	Normal vertical alignment
GAV_TOP	1	Top vertical alignment
GAV_CAP	2	Cap vertical alignment
GAV_HALF	3	Half vertical alignment
GAV_BASE	4	Base vertical alignment
GAV_BOTTOM	5	Bottom vertical alignment
<b>Update States:</b>		
GNOTPENDING	0	Not pending
GPENDING	1	Pending
<b>Viewport Priority States:</b>		
GHIGHER	0	Relative input priority higher
GLOWER	1	Relative input priority lower
<b>Visibility Flags:</b>		
GVISIBLE	0	Set to visible
GINVISIBLE	1	Set to invisible
<b>Workstation Category Types:</b>		
GOUTPUT	0	Output
GINPUT	1	Input
GOUTIN	2	Output/Input
GWISS	3	Workstation-independent segment storage
GMO	4	Metafile output
GMI	5	Metafile input

(continued on next page)

**Table B-1 (Cont.) C Constants**

Constant	Value	Description
<b>Workstation Class Types:</b>		
GVECTOR	0	Vector
GRASTER	1	Raster
GOTHER	2	Other device
<b>Workstation Color Availability States:</b>		
GCOLOUR	0	Color
GMONOCHROME	1	Monochrome
<b>Workstation States:</b>		
GINACTIVE	0	Inactive
GACTIVE	1	Active
<b>Workstation Types:</b>		
GWS_DEF	0	Default workstation type
GWS_DEFAULT	0	Default workstation type
GWS_MO	2	DEC GKS output metafile
GWS_GKS3_OUTPUT	2	DEC GKS output metafile
GWS_MI	3	DEC GKS input metafile
GWS_GKS3_INPUT	3	DEC GKS input metafile
GWS_WISS	5	Workstation-independent segment storage
GWS_CGMO	7	CGM output metafile
GWS_CGM_OUTPUT	7	CGM output metafile
GWS_VTO	10	Digital VT125 (output only)
GWS_VT_OUTPUT	10	Digital VT125 (output only)
GWS_V125C	11	Digital VT125 (color)
GWS_V125	11	Digital VT125 (color)
GWS_V125B	12	Digital VT125 (monochrome)
GWS_V125BW	12	Digital VT125 (monochrome)
GWS_V240C	13	Digital VT240 (color)
GWS_V240	13	Digital VT240 (color)
GWS_V240B	14	Digital VT240 (monochrome)
GWS_V240BW	14	Digital VT240 (monochrome)
GWS_LCG01	15	Digital LCG01 printer
GWS_LCP01	15	Digital LCP01 printer
GWS_V330	16	Digital VT330
GWS_VT330	16	Digital VT330
GWS_V340	17	Digital VT340 (color)

(continued on next page)

## Constants

### B.1 C Constants

**Table B-1 (Cont.) C Constants**

Constant	Value	Description
<b>Workstation Types:</b>		
GWS_VT340	17	Digital VT340 (color)
GWS_LA34	31	LA34 printer (graphics)
GWS_LA100	31	LA100 printer
GWS_LA50	32	LA50 printer
GWS_LA210	34	LA210 printer
GWS_LA75	35	LA75 printer
GWS_L2200	37	DEClaser 2200 printer
GWS_LASER_2200	37	DEClaser 2200 printer
GWS_LN03P	38	LN03 PLUS printer
GWS_LN03_PLUS	38	LN03 PLUS printer
GWS_L2100	39	DECLaser 2100 printer
GWS_LASER_2100	39	DECLaser 2100 printer
GWS_VSII	41	VAXstation II (monochrome)
GWS_VSGPX	41	VAXstation II/GPX
GWS_V2000	41	VAXstation 2000
GWS_V3200	41	VAXstation 3200
GWS_V3500	41	VAXstation 3500
GWS_VSVWS	41	VAXstation workstation software
GWS_VAXSTATION_VWS	41	VAXstation workstation software
GWS_LVPA	51	Digital LVP16
GWS_LVP16A	51	Digital LVP16
GWS_HP747	51	Hewlett-Packard HP7475
GWS_HP7475	51	Hewlett-Packard HP7475
GWS_LVPB	52	Digital LVP16 color plotter
GWS_LVP16B	52	Digital LVP16 color plotter
GWS_HP755	53	Hewlett-Packard HP7550 pen plotter
GWS_HP7550	53	Hewlett-Packard HP7550 pen plotter
GWS_HP758	54	Hewlett-Packard HP7580 pen plotter
GWS_HP7580	54	Hewlett-Packard HP7580 pen plotter
GWS_LGMPS	55	LaserGraphics MPS-2000 film recorder
GWS_LG_MPS2000	55	LaserGraphics MPS-2000 film recorder
GWS_H7585	56	Hewlett-Packard HP7585 pen plotter
GWS_HP7585	56	Hewlett-Packard HP7585 pen plotter
GWS_PTSC	61	PostScript
GWS_POSTSCRIPT	61	PostScript
GWS_PTSCC	62	PostScript (color)
GWS_COLOR_POSTSCRIPT	62	PostScript (color)

(continued on next page)

**Table B-1 (Cont.) C Constants**

Constant	Value	Description
<b>Workstation Types:</b>		
GWS_EPSF	65	Encapsulated PostScript
GWS_EPSFC	66	Color encapsulated PostScript
GWS_COLOR_EPSF	66	Encapsulated PostScript
GWS_TEKO	70	Tektronix 4014
GWS_TEK4014_OUTPUT	70	Tektronix 4014
GWS_T4014	72	Tektronix 4014
GWS_TEK4014	72	Tektronix 4014
GWS_T107O	80	Tektronix 4107
GWS_TEK4107_OUTPUT	80	Tektronix 4107
GWS_T4107	82	Tektronix 4107
GWS_TEK4107	82	Tektronix 4107
GWS_T207O	83	Tektronix 4207
GWS_TEK4207	83	Tektronix 4207
GWS_T4207	84	Tektronix 4207
GWS_TEK4207	84	Tektronix 4207
GWS_T128O	85	Tektronix 4128
GWS_TEK4128_OUTPUT	85	Tektronix 4128
GWS_T4128	86	Tektronix 4128
GWS_TEK4128	86	Tektronix 4128
GWS_T129O	87	Tektronix 4129
GWS_TEK4129_OUTPUT	87	Tektronix 4129
GWS_T4129	88	Tektronix 4129
GWS_TEK4129	88	Tektronix 4129
GWS_VS5O	87	VAXstation 500
GWS_VS500	88	VAXstation 500
GWS_LJ250	91	LJ250 ink jet printer
GWS_LJ25X	92	LJ250 180 DPI
GWS_LJ250_180DPI	92	LJ250 180 DPI
GWS_LA324	93	LA324 printer
GWS_DECWO	210	DECwindows Toolkit (output only)
GWS_DECWINDOWS_OUTPUT	210	DECwindows Toolkit (output only)
GWS_DECW	211	DECwindows Toolkit
GWS_DECWINDOWS	211	DECwindows Toolkit
GWS_DECWD	212	DECwindows drawable
GWS_DECWINDOWS_DRAWABLE	212	DECwindows drawable
GWS_DECWW	213	DECwindows widget
GWS_DECWINDOWS_WIDGET	213	DECwindows widget

(continued on next page)

## Constants

### B.1 C Constants

**Table B–1 (Cont.) C Constants**

Constant	Value	Description
<b>Workstation Types:</b>		
GWS_MOTIFO	230	Motif® (output only)
GWS_MOTIF_OUTPUT	230	Motif (output only)
GWS_MOTIF	231	Motif
GWS_MOTIFD	232	Motif drawable
GWS_MOTIF_DRAWABLE	232	Motif drawable
GWS_MOTIFW	233	Motif widget
GWS_MOTIF_WIDGET	233	Motif widget
GWS_MOTIF_PEX_OUTPUT	240	OSF/Motif® PEX (output only)
GWS_MOTIF_PEX	241	OSF/Motif PEX output and input
GWS_MOTIF_PEX_DRAWABLE	242	OSF/Motif PEX drawable
GWS_MOTIF_PEX_WIDGET	243	OSF/Motif PEX widget
GWS_DDIF	250	DDIF
<b>Writing Modes:</b>		
GWT_MCMT	2	Complement writing mode
GWT_MERS	3	Erase writing mode
GWT_MOVY	4	Overlay writing mode

## B.2 C Function Identifiers

Table B–2 lists the C function names, their values, and their defined identifiers.

**Table B–2 C Function Identifiers**

Function Name	Value	Identifier
<b>Control Function Identifiers:</b>		
ACTIVATE WORKSTATION	4	fn_activatews
CLEAR WORKSTATION	6	fn_clearws
CLOSE GKS	1	fn_closegks
CLOSE WORKSTATION	3	fn_closews
DEACTIVATE WORKSTATION	5	fn_deactivatews
ESCAPE	11	fn_escape
MESSAGE	10	fn_message
OPEN GKS	0	fn_opengks
OPEN WORKSTATION	2	fn_openws

(continued on next page)

## Constants B.2 C Function Identifiers

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Control Function Identifiers:</b>		
REDRAW ALL SEGMENTS ON WORKSTATION	7	fn_redrawsegws
SET DEFERRAL STATE	9	fn_setdeferst
UPDATE WORKSTATION	8	fn_updatews
<b>Output Function Identifiers:</b>		
CELL ARRAY	16	fn_cellarray
CELL ARRAY 3	111	fn_cellarray3
FILL AREA	15	fn_fillarea
FILL AREA 3	108	fn_fillarea3
FILL AREA SET	110	fn_fillareaset
FILL AREA SET 3	109	fn_fillareaset3
GENERALIZED DRAWING PRIMITIVE	17	fn_gdp
GENERALIZED DRAWING PRIMITIVE 3	112	fn_gdp3
POLYLINE	12	fn_polyline
POLYLINE 3	105	fn_polyline3
POLYMARKER	13	fn_polymarker
POLYMARKER 3	106	fn_polymarker3
TEXT	14	fn_text
TEXT 3	107	fn_text3
<b>Attribute Function Identifiers:</b>		
SET ASPECT SOURCE FLAGS	41	fn_setasf
SET ASPECT SOURCE FLAGS 3	119	fn_setasf3
SET CHARACTER EXPANSION FACTOR	28	fn_setcharexpan
SET CHARACTER HEIGHT	31	fn_setcharheight
SET CHARACTER SPACING	29	fn_setcharspace
SET CHARACTER UP VECTOR	32	fn_setcharup
SET COLOUR MODEL	144	fn_setcolourmodel
SET COLOUR REPRESENTATION	48	fn_setcolourrep
SET EDGE COLOUR INDEX	118	fn_setedgecolourind
SET EDGE FLAG	115	fn_setedgeflag
SET EDGE INDEX	114	fn_setedgeindex
SET EDGE REPRESENTATION	120	fn_setedgerep
SET EDGETYPE	116	fn_setedgetype
SET EDGEWIDTH SCALE FACTOR	117	fn_setedgewidthscfac
SET FILL AREA COLOUR INDEX	38	fn_setfillcolourind

(continued on next page)

## Constants

### B.2 C Function Identifiers

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Attribute Function Identifiers:</b>		
SET FILL AREA INDEX	35	fn_setfillind
SET FILL AREA INTERIOR STYLE	36	fn_setfillintstyle
SET FILL AREA REPRESENTATION	46	fn_setfillrep
SET FILL AREA STYLE INDEX	37	fn_setfillstyleind
SET HLHSR IDENTIFIER	126	fn_sethlhsrid
SET HLHSR MODE	127	fn_sethlhsrmode
SET LINETYPE	19	fn_setlinetype
SET LINEWIDTH SCALE FACTOR	20	fn_setlinewidth
SET MARKER SIZE SCALE FACTOR	24	fn_setmarkersize
SET MARKER TYPE	23	fn_setmarkertype
SET PATTERN REFERENCE POINT	40	fn_setpatrefpt
SET PATTERN REFERENCE POINT AND VECTORS	113	fn_setpatrefptvec
SET PATTERN REPRESENTATION	47	fn_setpatrep
SET PATTERN SIZE	39	fn_setpatsize
SET PICK IDENTIFIER	42	fn_setpickid
SET POLYLINE COLOUR INDEX	21	fn_setlinecolourind
SET POLYLINE INDEX	18	fn_setlineind
SET POLYLINE REPRESENTATION	43	fn_setlinerep
SET POLYMARKER COLOUR INDEX	25	fn_setmarkercolourind
SET POLYMARKER INDEX	22	fn_setmarkerind
SET POLYMARKER REPRESENTATION	44	fn_setmarkerrep
SET TEXT ALIGNMENT	34	fn_settextalign
SET TEXT COLOUR INDEX	30	fn_settextcolourind
SET TEXT FONT AND PRECISION	27	fn_settextfontprec
SET TEXT INDEX	26	fn_settextind
SET TEXT PATH	33	fn_settextpath
SET TEXT REPRESENTATION	45	fn_settextrep
<b>Transformation Function Identifiers:</b>		
ACCUMULATE TRANSFORMATION MATRIX	–119	fn_accumtran
ACCUMULATE TRANSFORMATION MATRIX 3	–118	fn_accumtran3
EVALUATE TRANSFORMATION MATRIX	–117	fn_evaltran
EVALUATE TRANSFORMATION MATRIX 3	–116	fn_evaltran3
EVALUATE VIEW MAPPING MATRIX 3	–115	fn_evalviewmaptran3
EVALUATE VIEW ORIENTATION MATRIX 3	–114	fn_evalvieworienttran3
SELECT NORMALIZATION TRANSFORMATION	52	fn_setntran
SET CLIPPING INDICATOR	53	fn_setclip

(continued on next page)



**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Transformation Function Identifiers:</b>		
SET VIEW INDEX	123	fn_setviewindex
SET VIEW REPRESENTATION 3	124	fn_setviewrep3
SET VIEW TRANSFORMATION INPUT PRIORITY	125	fn_setviewxformpr
SET VIEWPORT	50	fn_setviewport
SET VIEWPORT 3	122	fn_setviewport3
SET VIEWPORT INPUT PRIORITY	51	fn_setviewportinputpri
SET WINDOW	49	fn_setwindow
SET WINDOW 3	121	fn_setwindow3
SET WORKSTATION VIEWPORT	55	fn_setwsviewport
SET WORKSTATION VIEWPORT 3	129	fn_setwsviewport3
SET WORKSTATION WINDOW	54	fn_setwswindow
SET WORKSTATION WINDOW 3	128	fn_setwswindow3
<b>Segment Function Identifiers:</b>		
ASSOCIATE SEGMENT WITH WORKSTATION	61	fn_assocsegws
CLOSE SEGMENT	57	fn_closeseg
COPY SEGMENT TO WORKSTATION	62	fn_copysegws
CREATE SEGMENT	56	fn_createseg
DELETE SEGMENT	59	fn_delseg
DELETE SEGMENT FROM WORKSTATION	60	fn_delsegws
INSERT SEGMENT	63	fn_insertseg
INSERT SEGMENT 3	130	fn_insertseg3
RENAME SEGMENT	58	fn_renameseg
SET DETECTABILITY	68	fn_setdet
SET HIGHLIGHTING	66	fn_sethighlight
SET SEGMENT PRIORITY	67	fn_setsegpri
SET SEGMENT TRANSFORMATION	64	fn_setsegtran
SET SEGMENT TRANSFORMATION 3	131	fn_setsegtran3
SET VISIBILITY	65	fn_setvis
<b>Input Function Identifiers:</b>		
AWAIT EVENT	93	fn_awaitevent
FLUSH DEVICE EVENTS	94	fn_flushevents
GET CHOICE	98	fn_getchoice
GET LOCATOR	95	fn_getloc
GET LOCATOR 3	142	fn_getloc3

(continued on next page)

## Constants

### B.2 C Function Identifiers

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Input Function Identifiers:</b>		
GET PICK	99	fn_getpick
GET STRING	100	fn_getstring
GET STROKE	96	fn_getstroke
GET STROKE 3	143	fn_getstroke3
GET VALUATOR	97	fn_getval
INITIALIZE CHOICE	72	fn_initchoice
INITIALIZE CHOICE 3	135	fn_initchoice3
INITIALIZE LOCATOR	69	fn_initloc
INITIALIZE LOCATOR 3	132	fn_initloc3
INITIALIZE PICK	73	fn_initpick
INITIALIZE PICK 3	136	fn_initpick3
INITIALIZE STRING	74	fn_initstring
INITIALIZE STRING 3	137	fn_initstring3
INITIALIZE STROKE	70	fn_initstroke
INITIALIZE STROKE 3	133	fn_initstroke3
INITIALIZE VALUATOR	71	fn_initval
INITIALIZE VALUATOR 3	134	fn_initval3
REQUEST CHOICE	84	fn_reqchoice
REQUEST LOCATOR	81	fn_reqloc
REQUEST LOCATOR 3	138	fn_reqloc3
REQUEST PICK	85	fn_reqpick
REQUEST STRING	86	fn_reqstring
REQUEST STROKE	82	fn_reqstroke
REQUEST STROKE 3	139	fn_reqstroke3
REQUEST VALUATOR	83	fn_reqval
SAMPLE CHOICE	90	fn_samplechoice
SAMPLE LOCATOR	87	fn_sampleloc
SAMPLE LOCATOR 3	140	fn_sampleloc3
SAMPLE PICK	91	fn_samplepick
SAMPLE STRING	92	fn_samplestring
SAMPLE STROKE	88	fn_samplestroke
SAMPLE STROKE 3	141	fn_samplestroke3
SAMPLE VALUATOR	89	fn_sampleval
SET CHOICE MODE	78	fn_setchoicemode
SET LOCATOR MODE	75	fn_setlocmode
SET PICK MODE	79	fn_setpickmode

(continued on next page)

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Input Function Identifiers:</b>		
SET STRING MODE	80	fn_setstringmode
SET STROKE MODE	76	fn_setstrokemode
SET VALUATOR MODE	77	fn_setvalmode
<b>Metafile Function Identifiers:</b>		
GET ITEM TYPE FROM GKSM	102	fn_gettypegksm
INTERPRET ITEM	104	fn_interpret
READ ITEM FROM GKSM	103	fn_readgksm
WRITE ITEM TO GKSM	101	fn_writegksm
<b>Inquiry Function Identifiers:</b>		
INQUIRE ASPECT SOURCE FLAGS	–123	fn_inqasf
INQUIRE ASPECT SOURCE FLAGS 3	–152	fn_inqasf3
INQUIRE CHARACTER BASE VECTOR	–124	fn_inqcharbase
INQUIRE CHARACTER EXPANSION FACTOR	–125	fn_inqcharexpan
INQUIRE CHARACTER HEIGHT	–126	fn_inqcharheight
INQUIRE CHARACTER SPACING	–127	fn_inqcharspace
INQUIRE CHARACTER UP VECTOR	–128	fn_inqcharup
INQUIRE CHARACTER WIDTH	–129	fn_inqcharwidth
INQUIRE CHOICE DEVICE STATE	–56	fn_inqchoicest
INQUIRE CHOICE DEVICE STATE 3	–55	fn_inqchoicest3
INQUIRE CLIPPING	–19	fn_inqclip
INQUIRE CLIPPING 3	–18	fn_inqclip3
INQUIRE COLOUR FACILITIES	–80	fn_inqcolourfacil
INQUIRE COLOUR MODEL	–42	fn_inqcolourmodel
INQUIRE COLOUR MODEL FACILITIES	–82	fn_inqcolourmodelfacil
INQUIRE COLOUR REPRESENTATION	–41	fn_inqcolourrep
INQUIRE CURRENT HLHSR IDENTIFIER VALUE	–11	inqhlhsrid
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	–13	fn_inqindivattr
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3	–12	fn_inqindivattr3
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	–14	fn_inqcurntrannum
INQUIRE CURRENT PICK IDENTIFIER VALUE	–10	fn_inqcurpickid
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	–9	fn_inqprimattr
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3	–8	fn_inqprimattr3
INQUIRE DEFAULT CHOICE DEVICE DATA	–102	fn_inqdefchoice
INQUIRE DEFAULT CHOICE DEVICE DATA 3	–101	fn_inqdefchoice3

(continued on next page)

## Constants

### B.2 C Function Identifiers

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Inquiry Function Identifiers:</b>		
INQUIRE DEFAULT DEFERRAL STATE VALUES	–67	fn_inqdefdeferst
INQUIRE DEFAULT LOCATOR DEVICE DATA	–96	fn_inqdefloc
INQUIRE DEFAULT LOCATOR DEVICE DATA 3	–95	fn_inqdefloc3
INQUIRE DEFAULT PICK DEVICE DATA	–104	fn_inqdefpick
INQUIRE DEFAULT PICK DEVICE DATA 3	–103	fn_inqdefpick3
INQUIRE DEFAULT STRING DEVICE DATA	–106	fn_inqdefstring
INQUIRE DEFAULT STRING DEVICE DATA 3	–105	fn_inqdefstring3
INQUIRE DEFAULT STROKE DEVICE DATA	–98	fn_inqdefstroke
INQUIRE DEFAULT STROKE DEVICE DATA 3	–97	fn_inqdefstroke3
INQUIRE DEFAULT VALUATOR DEVICE DATA	–100	fn_inqdefval
INQUIRE DEFAULT VALUATOR DEVICE DATA 3	–99	fn_inqdefval3
INQUIRE DISPLAY SPACE SIZE	–64	fn_inqdisplaysize
INQUIRE DISPLAY SPACE SIZE 3	–63	fn_inqdisplaysize3
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	–93	fn_inqmodsegattr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	–66	fn_inqmodwsattr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3	–65	fn_inqmodwsattr3
INQUIRE EDGE COLOUR INDEX	–151	fn_inqedgecolourind
INQUIRE EDGE FACILITIES	–76	fn_inqedgefacil
INQUIRE EDGE FLAG	–153	fn_inqedgeflag
INQUIRE EDGE REPRESENTATION	–37	fn_inqedgerep
INQUIRE EDGETYPE	–154	fn_inqedgetype
INQUIRE EDGEWIDTH SCALE FACTOR	–150	fn_inqedgewidth
INQUIRE FILL AREA COLOUR INDEX	–130	fn_inqfillcolourind
INQUIRE FILL AREA FACILITIES	–74	fn_inqfillfacil
INQUIRE FILL AREA INDEX	–131	fn_inqfillind
INQUIRE FILL AREA INTERIOR STYLE	–132	fn_inqfillintstyle
INQUIRE FILL AREA REPRESENTATION	–35	fn_inqfillrep
INQUIRE FILL AREA STYLE INDEX	–133	fn_inqfillstyleind
INQUIRE GENERALIZED DRAWING PRIMITIVE	–89	fn_inqgdp
INQUIRE GENERALIZED DRAWING PRIMITIVE 3	–88	fn_inqgdp3
INQUIRE HLHSR FACILITIES	–85	fn_inqhlhsrfac
INQUIRE HLHSR MODE	–45	fn_inqhlhsrmode
INQUIRE INPUT QUEUE OVERFLOW	–113	fn_inqinputoverflow
INQUIRE LEVEL OF GKS	–2	fn_inqlevelgks
INQUIRE LINETYPE	–136	fn_inqlinetype

(continued on next page)

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Inquiry Function Identifiers:</b>		
INQUIRE LINE WIDTH SCALE FACTOR	–137	fn_inqlinewidth
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	–87	fn_inqavailgdp
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3	–86	fn_inqavailgdp3
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	–3	fn_inqavailwstypes
INQUIRE LIST OF COLOUR INDICES	–40	fn_inqcolourindices
INQUIRE LIST OF EDGE INDICES	–36	fn_inqedgeindices
INQUIRE LIST OF FILL AREA INDICES	–34	fn_inqfillindices
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	–15	fn_inqntrannum
INQUIRE LIST OF PATTERN INDICES	–38	fn_inqpatindices
INQUIRE LIST OF POLYLINE INDICES	–26	fn_inqlineindices
INQUIRE LIST OF POLYMARKER INDICES	–28	fn_inqmarkerindices
INQUIRE LIST OF TEXT INDICES	–30	fn_inqtextindices
INQUIRE LIST OF VIEW INDICES	–43	fn_inqviewindices
INQUIRE LOCATOR DEVICE STATE	–50	fn_inqlocst
INQUIRE LOCATOR DEVICE STATE 3	–49	fn_inqlocst3
INQUIRE MARKER SIZE SCALE FACTOR	–140	fn_inqmarkersize
INQUIRE MARKER TYPE	–141	fn_inqmarkertype
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	–91	fn_inqmaxwsstables
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3	–90	fn_inqmaxwsstables3
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	–5	fn_inqmaxntrannum
INQUIRE MORE SIMULTANEOUS EVENTS	–22	fn_inqmoreevents
INQUIRE NAME OF OPEN SEGMENT	–20	fn_inqnameopenseg
INQUIRE NORMALIZATION TRANSFORMATION	–17	fn_inqntran
INQUIRE NORMALIZATION TRANSFORMATION 3	–16	fn_inqntran3
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	–94	fn_inqnumavailinput
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	–92	fn_inqnumsegpri
INQUIRE OPERATING STATE VALUE	–1	fn_inqopst
INQUIRE PATTERN FACILITIES	–78	fn_inqpatfacil
INQUIRE PATTERN HEIGHT VECTOR	–142	fn_inqpatheight
INQUIRE PATTERN REFERENCE POINT	–143	fn_inqpatrefpt
INQUIRE PATTERN REFERENCE POINT AND VECTORS	–155	fn_inqpatrefptvector
INQUIRE PATTERN REPRESENTATION	–39	fn_inqpatrep

(continued on next page)

## Constants

### B.2 C Function Identifiers

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Inquiry Function Identifiers:</b>		
INQUIRE PATTERN WIDTH VECTOR	–144	fn_inqpatwidth
INQUIRE PICK DEVICE STATE	–58	fn_inqpickst
INQUIRE PICK DEVICE STATE 3	–57	fn_inqpickst3
INQUIRE PIXEL	–112	fn_inqpixel
INQUIRE PIXEL ARRAY	–111	fn_inqpixelarray
INQUIRE PIXEL ARRAY DIMENSIONS	–110	fn_inqpixelarraydim
INQUIRE POLYLINE COLOUR INDEX	–134	fn_inqlinecolourind
INQUIRE POLYLINE FACILITIES	–68	fn_inqlinefacil
INQUIRE POLYLINE INDEX	–135	fn_inqlineind
INQUIRE POLYLINE REPRESENTATION	–27	fn_inqlinerep
INQUIRE POLYMARKER COLOUR INDEX	–138	fn_inqmarkercolourind
INQUIRE POLYMARKER FACILITIES	–70	fn_inqmarkerfacil
INQUIRE POLYMARKER INDEX	–139	fn_inqmarkerind
INQUIRE POLYMARKER REPRESENTATION	–29	fn_inqmarkerrep
INQUIRE PREDEFINED COLOUR REPRESENTATION	–81	fn_inqpredcolourep
INQUIRE PREDEFINED EDGE REPRESENTATION	–77	fn_inqprededgerep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	–75	fn_inqpredfillrep
INQUIRE PREDEFINED PATTERN REPRESENTATION	–79	fn_inqpredpatrep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	–69	fn_inqpredlinerep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	–71	fn_inqpredmarkerrep
INQUIRE PREDEFINED TEXT REPRESENTATION	–73	fn_inqpredtextrep
INQUIRE PREDEFINED VIEW REPRESENTATION	–84	fn_inqpredviewrep
INQUIRE SEGMENT ATTRIBUTES	–109	fn_inqsegattr
INQUIRE SEGMENT ATTRIBUTES 3	–108	fn_inqsegattr3
INQUIRE SET OF ACTIVE WORKSTATIONS	–7	fn_inqactivews
INQUIRE SET OF ASSOCIATED WORKSTATIONS	–107	fn_inqassocws
INQUIRE SET OF OPEN WORKSTATIONS	–6	fn_inqopenws
INQUIRE SET OF SEGMENT NAMES IN USE	–21	fn_inqsegnames
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	–48	fn_inqsegnamesws
INQUIRE STRING DEVICE STATE	–60	fn_inqstringst
INQUIRE STRING DEVICE STATE 3	–59	fn_inqstringst3
INQUIRE STROKE DEVICE STATE	–52	fn_inqstrokest
INQUIRE STROKE DEVICE STATE 3	–51	fn_inqstrokest3
INQUIRE TEXT ALIGNMENT	–145	fn_inqtextalign
INQUIRE TEXT COLOUR INDEX	–146	fn_inqtextcolourind
INQUIRE TEXT EXTENT	–33	fn_inqtextextent
INQUIRE TEXT EXTENT 3	–32	fn_inqtextextent3

(continued on next page)

**Table B–2 (Cont.) C Function Identifiers**

Function Name	Value	Identifier
<b>Inquiry Function Identifiers:</b>		
INQUIRE TEXT FACILITIES	–72	fn_inqtextfacil
INQUIRE TEXT FONT AND PRECISION	–147	fn_inqtextfontprec
INQUIRE TEXT INDEX	–148	fn_inqtextind
INQUIRE TEXT PATH	–149	fn_inqtextpath
INQUIRE TEXT REPRESENTATION	–31	fn_inqtextrep
INQUIRE VALUATOR DEVICE STATE	–54	fn_inqvalst
INQUIRE VALUATOR DEVICE STATE 3	–53	fn_inqvalst3
INQUIRE VIEW FACILITIES	–83	fn_inqviewfac
INQUIRE VIEW REPRESENTATION 3	–44	fn_inqviewrep3
INQUIRE WORKSTATION CATEGORY	–61	fn_inqwscategory
INQUIRE WORKSTATION CLASSIFICATION	–62	fn_inqwsclass
INQUIRE WORKSTATION CONNECTION AND TYPE	–23	fn_inqwsconntype
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	–25	fn_inqwsdeferupdatest
INQUIRE WORKSTATION MAXIMUM NUMBERS	–4	fn_inqwsmaxnum
INQUIRE WORKSTATION STATE	–24	fn_inqwsst
INQUIRE WORKSTATION TRANSFORMATION	–47	fn_inqwstran
INQUIRE WORKSTATION TRANSFORMATION 3	–46	fn_inqwstran3
<b>Error Handling Function Identifiers:</b>		
EMERGENCY CLOSE GKS	–120	fn_emergencyclosegks
ERROR HANDLING	–121	fn_errorhand
ERROR LOGGING	–122	fn_errorlog

### **B.3 Error Constants**

Table B–3 lists the C error constant names, their values, and their descriptions. These are used for error handling.

## Constants

### B.3 Error Constants

**Table B–3 C Error Constants**

Constant	Value	Description
<b>Operating State Errors:</b>		
NO_ERROR	0	Successful completion of routine ****
ENOTGKCL	1	GKS not in proper state: GKS shall be in the state GKCL in routine ****
ENOTGKOP	2	GKS not in proper state: GKS shall be in the state GKOP in routine ****
ENOTWSAC	3	GKS not in proper state: GKS shall be in the state WSAC in routine ****
ENOTSGOP	4	GKS not in proper state: GKS shall be in the state SGOP in routine ****
ENOTACOP	5	GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP in routine ****
ENOTOPAC	6	GKS not in proper state: GKS shall be either in the state WSOP or in the state WSAC in routine ****
ENOTWSOP	7	GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine ****
ENOTGWWS	8	GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC or SGOP in routine ****
<b>Workstation Errors:</b>		
EWSIDINV	20	Specified workstation identifier is invalid in routine ****
ECNIDINV	21	Specified connection identifier is invalid in routine ****
EWSTPINV	22	Specified workstation type is invalid in routine ****
ENOWSTYP	23	Specified workstation type does not exist in routine ****
EWSISOPN	24	Specified workstation is open in routine ****
EWSNOTOP	25	Specified workstation is not open in routine ****
EWSCNTOP	26	Specified workstation cannot be opened in routine ****
EWISSNOP	27	Workstation Independent Segment Storage is not open in routine ****
EWISSOPN	28	Workstation Independent Segment Storage is already open in routine ****
EWSISACT	29	Specified workstation is active in routine ****
EWSNTACT	30	Specified workstation is not active in routine ****
EWSCATMO	31	Specified workstation is of category MO in routine ****
EWSNOTMO	32	Specified workstation is not of category MO in routine ****
EWSCATMI	33	Specified workstation is of category MI in routine ****
EWSNOTMI	34	Specified workstation is not of category MI in routine ****
EWSCATIN	35	Specified workstation is of category INPUT in routine ****
EWSIWISS	36	Specified workstation is Workstation Independent Segment Storage in routine ****
EWSNOTOI	37	Specified workstation is not of category OUTIN in routine ****

(continued on next page)



**Table B–3 (Cont.) C Error Constants**

<b>Constant</b>	<b>Value</b>	<b>Description</b>
<b>Workstation Errors:</b>		
EWSNOTIO	38	Specified workstation is neither of category INPUT nor of category OUTIN in routine ****
EWSNOTOO	39	Specified workstation is neither of category OUTPUT nor of category OUTIN in routine ****
EWSNOPXL	40	Specified workstation has no pixel store readback capability in routine ****
EWSNOGDP	41	Specified workstation type is not able to generate the specified generalized drawing primitive in routine ****
EWSMXOPN	42	Maximum number of simultaneously open workstations would be exceeded in routine ****
EWSMXXACT	43	Maximum number of simultaneously active workstations would be exceeded in routine ****
<b>Transformation Function Errors:</b>		
EBADXFRM	50	Transformation number is invalid in routine ****
EBADRCTD	51	Rectangle definition is invalid in routine ****
EBDVIEWP	52	Viewport is not within the Normalized Device Coordinate unit square in routine ****
EBDWINDW	53	Workstation window is not within the Normalized Device Coordinate unit square in routine ****
EVIEWDSP	54	Workstation viewport is not within the display space in routine ****
<b>Attribute Function Errors:</b>		
EBADLINX	60	Polyline index is invalid in routine ****
ENOLINEX	61	A representation for the specified polyline index has not been defined on this workstation in routine ****
ENOPLINX	62	A representation for the specified polyline index has not been predefined on this workstation in routine ****
ELINEEQZ	63	Specified line type is equal to zero in routine ****
ENOLINTP	64	Specified line type is not supported on this workstation in routine ****
ELNWDLTZ	65	Line width scale factor is less than zero in routine ****
EBADMXX	66	Polymarker index is invalid in routine ****
ENOMARKX	67	A representation for the specified polymarker index has not been defined on this workstation in routine ****
ENOPMXX	68	A representation for the specified polymarker index has not been predefined on this workstation in routine ****
EMARKEQZ	69	Specified marker type is equal to zero in routine ****
ENOMRKTTP	70	Specified marker type is not supported on this workstation in routine ****

(continued on next page)

## Constants

### B.3 Error Constants

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Attribute Function Errors:</b>		
EMKSZLTZ	71	Marker size scale factor is less than zero in routine ****
EBADTXTX	72	Text index is invalid in routine ****
ENOTEXTX	73	A representation for the specified text index has not been defined on this workstation in routine ****
ENOPTXTX	74	A representation for the specified text index has not been predefined on this workstation in routine ****
ETXTFEQZ	75	Text font is equal to zero in routine ****
ENOTXTFP	76	Requested text font is not supported for the specified precision on this workstation in routine ****
ECEXFLEZ	77	Character expansion factor is less than or equal to zero in routine ****
ECHHTLEZ	78	Character height is less than or equal to zero in routine ****
ECHRUPVZ	79	Length of character up vector is zero in routine ****
EBADFILX	80	Fill area index is invalid in routine ****
ENOFILLX	81	A representation for the specified fill area index has not been defined on this workstation in routine ****
ENOPFILX	82	A representation for the specified fill area index has not been predefined on this workstation in routine ****
ENOFSTYL	83	Specified fill area interior style is not supported on this workstation in routine ****
ESTYLEQZ	84	Style (pattern or hatch) index is equal to zero in routine ****
EBADPATN	85	Specified pattern index is invalid in routine ****
ENOHATCH	86	Specified hatch style is not supported on this workstation in routine ****
EPATSZLZ	87	Pattern size value is not positive in routine ****
ENOPATNX	88	A representation for the specified pattern index has not been defined on this workstation in routine ****
ENOPPTNX	89	A representation for the specified pattern index has not been predefined on this workstation in routine ****
ENOPSTYL	90	Interior style PATTERN is not supported on this workstation in routine ****
ECADIMEN	91	Dimensions of color array are invalid in routine ****
ECINDXLZ	92	Color index is less than zero in routine ****
EBADCOLX	93	Color index is invalid in routine ****
ENOCOLRX	94	A representation for the specified color index has not been defined on this workstation in routine ****

(continued on next page)

**Table B-3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Attribute Function Errors:</b>		
ENOPCLRX	95	A representation for the specified color index has not been predefined on this workstation routine ****
ECOLRNGE	96	Color index is outside range of the current color model in routine ****
EBADPICK	97	Pick identifier is invalid in routine ****
ECOLMDNA	98	Specified color model is not available on the workstation in routine ****
<b>Output Function Errors:</b>		
ENPOINTS	100	Number of points is invalid in routine ****
ECHRCODE	101	Invalid code in string in routine ****
EBDGDPID	102	Generalized drawing primitive identifier is invalid in routine ****
EGDPDATA	103	Content of generalized drawing primitive data record is invalid in routine ****
ECANTGDP	104	At least one active workstation is not able to generate the specified generalized drawing primitive in routine ****
ECNTGDPC	105	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformations and clipping rectangle in routine ****
<b>Segment Function Errors:</b>		
EBADNAME	120	Specified segment name is invalid in routine ****
ENAMUSED	121	Specified segment name is already in use in routine ****
EWHATSEG	122	Specified segment does not exist in routine ****
EWORKSEG	123	Specified segment does not exist on specified workstation in routine ****
EWISSSEG	124	Specified segment does not exist on Workstation Independent Segment Storage in routine ****
ESEGOPEN	125	Specified segment is open in routine ****
ESEGPRIR	126	Segment priority is outside the range [0,1] in routine ****

(continued on next page)

## Constants

### B.3 Error Constants

**Table B-3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Input Function Errors:</b>		
ENOINDEV	140	Specified input device is not present on workstation in routine ****
EREQUEST	141	Input device is not in REQUEST mode in routine ****
ENSAMPLE	142	Input device is not in SAMPLE mode in routine ****
ENOEVSMIP	143	EVENT and SAMPLE input mode are not available at this level of GKS in routine ****
ENOPETWS	144	Specified prompt and echo type is not supported on this workstation in routine ****
EEBOUNDS	145	Echo area is outside display space in routine ****
EBADDATA	146	Contents of input data record are invalid in routine ****
EINQOVFL	147	Input queue has overflowed in routine ****
ENOQOVFL	148	Input queue has not overflowed since GKS was opened or the last invocation of INQUIRE INPUT QUEUE OVERFLOW in routine ****
EASWSCLO	149	Input queue has overflowed, but associated workstation has been closed in routine ****
ENOCURIV	150	No input value of the correct class is in the current event report in routine ****
EINVTOUT	151	Timeout is invalid in routine ****
EBDINITV	152	Initial value is invalid in routine ****
ESTROKSZ	153	Number of points in the initial stroke is greater than the buffer size in routine ****
ESTRINSZ	154	Length of the initial string is greater than the buffer size in routine ****
<b>Metafile Function Errors:</b>		
ERESERVE	160	Item type is not allowed for user items in routine ****
EBDLNGTH	161	Item length is invalid in routine ****
EMNOITEM	162	No item is left in GKS Metafile input in routine ****
EMITMINV	163	Metafile item is invalid in routine ****
ENOTGKSI	164	Item type is not a valid GKS item in routine ****
EBADCNTS	165	Content of item data record is invalid for the specified item type in routine ****
EEBDMXDR	166	Maximum item data record length is invalid in routine ****
EINTERPT	167	User item cannot be interpreted in routine ****
ENOFUNCT	168	Specified function is not supported at this level of GKS in routine ****

(continued on next page)

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Escape Function Errors:</b>		
ENOESCFN	180	Specified escape function is not supported in routine ****
ESCIDINV	181	Specified escape function identifier is invalid in routine ****
EESCDATA	182	Contents of escape data record are invalid in routine ****
<b>Miscellaneous Errors:</b>		
EBDERRFL	200	Specified error file is invalid in routine ****
<b>System Errors:</b>		
EMEMSPAC	300	Storage overflow has occurred in GKS in routine ****
ESEGPSAC	301	Storage overflow has occurred in segment storage in routine ****
EIO_READ	302	Input/Output error has occurred while reading in routine ****
EIOWRITE	303	Input/Output error has occurred while writing in routine ****
EIOSENDD	304	Input/Output error has occurred while sending data to a workstation in routine ****
EIORECDA	305	Input/Output error has occurred while receiving data from a workstation in routine ****
EIOLIBMA	306	Input/Output error has occurred during program library management in routine ****
EIORDWDT	307	Input/Output error has occurred while reading workstation description table in routine ****
EMATHERR	308	Arithmetic error has occurred in routine ****
<b>Three-Dimensional Errors:</b>		
EVUVVPNC	400	View up vector and view plane normal are collinear in routine ****
EVPNULL	401	View plane normal is a null vector in routine ****
EVUVNULL	402	View up vector is a null vector in routine ****
EBADPVPL	403	Projection viewport limits are not within NPC range in routine ****
EPRPFBFP	404	Projection reference point is between the front and back clipping planes in routine ****
EPRPONVP	405	Projection reference point is on the view plane in routine ****
EBOXDINV	406	Box definition is invalid in routine ****
EBADVPRT	407	Viewport is not within NDC unit cube in routine ****
EVIINVAL	408	Specified view index is invalid in routine ****

(continued on next page)

## Constants

### B.3 Error Constants

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Three-Dimensional Errors:</b>		
EVRUNDEF	409	A representation for the specified view index has not been defined on this workstation in routine ****
EVRNPDEF	410	A representation for the specified view index has not been predefined on this workstation in routine ****
ENPCWBAD	411	Workstation window limits are not within the NPC cube in routine ****
EBPIFOFP	412	Back clipping plane is in front of front clipping plane in routine ****
EVWCLBAD	413	View clipping limits not within NPC range in routine ****
EEIINVAL	420	Edge index is invalid in routine ****
EEIUNDEF	421	A representation for the specified edge index has not been defined on this workstation in routine ****
EERNPDEF	422	A representation for the specified edge index has not been predefined on this workstation in routine ****
EEDGETPZ	423	Edge type is equal to zero in routine ****
EETNOTWS	424	Specified edge type is not supported on this workstation in routine ****
EEWSFLTZ	425	Edge width scale factor is less than zero in routine ****
EPRVCOLL	426	Pattern reference vectors are collinear in routine ****
EHLHSRNS	427	Specified HLHSR mode not supported on workstation in routine ****
EHLHSRII	428	Specified HLHSR identifier is invalid in routine ****
EHLHSRMI	429	Specified HLHSR mode is invalid in routine ****
ETPVCOLL	430	The text direction vectors are collinear in routine ****
ELOPLINV	431	List of point lists is invalid in routine ****
EGDPDRAW	432	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformation and clipping volume in routine ****

#### **C Language-Dependent Errors:**

EBUFSPAC	2200	Buffer overflow on input or inquiry function in routine ****
----------	------	--

#### **Implementation-Specific Errors:**

GE_COLMAP_NOT_CREATE	–2	Requested color map could not be created as specified in routine ****
GE_WSDF_INVAL	–3	Invalid data in workstation description file in routine ****
GE_WST_MASK_INVAL	–4	Invalid bit mask in workstation type in routine ****
GE_CH_BAD_STR	–5	Bad string addresses found writing choice data record in routine ****
GE_EA_TOO_NAR	–6	Echo area is too narrow for data in routine ****

(continued on next page)

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_TOO_MANY_CH	–7	Maximum number of representable choices exceeded in routine ****
GE_EA_TOO_SHORT	–8	Echo area is too short for data in routine ****
GE_BF_INTREP_NOTS	–9	Binary format and integer number representation not supported in this implementation of GKS in routine ****
GE_ASF_INVAL	–10	Invalid value specified for ASF in routine ****
GE_FAIS_INVAL	–11	Invalid value specified for fill area interior style in routine ****
GE_TXT_ALIGNH_INVAL	–12	Invalid value specified for horizontal component of text alignment in routine ****
GE_TXT_ALIGNV_INVAL	–13	Invalid value specified for vertical component of text alignment in routine ****
GE_TXT_PREC_INVAL	–14	Invalid value specified for text precision in routine ****
GE_TXT_PATH_INVAL	–15	Invalid value specified for text path in routine ****
GE_ECHO_SW_INVAL	–16	Echo switch is invalid in routine ****
GE_DEV_VAL_TYPE_INVAL	–17	Inquired device values not set or realized in routine ****
GE_INTERPRET_ERR	–18	The following error occurred when GKS was interpreting an item ****
GE_ERR_CODE_INVAL	–19	Invalid error status parameter specified in routine ****
GE_GKS_IN_ERR_ST	–20	GKS not in proper state: GKS in the ERROR state in routine ****
GE_FCN_NOTS_AT_LEV	–21	Function is not supported in this level of GKS in routine ****
GE_SEG_XFORM_INVAL	–22	Invalid segment transformation in routine ****
GE_CLIP_FL_INVAL	–23	Invalid value specified for clipping flag in routine ****
GE_VP_PRIOR_FL_INVAL	–24	Invalid value specified for viewport priority flag in routine ****
GE_UPDWS_FL_INVAL	–25	Invalid value specified for update workstation flag in routine ****
GE_DEFMODE_INVAL	–26	Invalid value specified for deferral mode in routine ****
GE_REGENMODE_INVAL	–27	Invalid value specified for regeneration mode in routine ****
GE_EXP_FACT_INVAL	–28	Invalid value specified for expansion factor in routine ****
GE_PET_REC_SIZE_INVAL	–29	Invalid data record size for specified prompt and echo type in routine ****
GE_WSH_ACT_ERR	–30	Cannot load workstation handler: error during image activation in routine ****
GE_DFT_INVAL	–31	Cannot load graphics handler: invalid DFT in routine ****
GE_FONT_FILE_INVAL	–32	Font file for stroke precision text not found or unusable in routine ****
GE_ARR_DESC_INVAL	–33	Array descriptor is not acceptable in routine ****

(continued on next page)

## Constants

### B.3 Error Constants

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_STRLLEN_INVALID	–34	String length less than or equal to 0 in routine ****
GE_DH_ERR	–35	Kernel has detected an unexpected error from a device handler in routine ****
GE_DH_NOACT	–36	Cannot load device handler: error during image activation in routine ****
GE_DH_EVF_ERR	–37	Error in device handler during event flag allocation in routine ****
GE_DH_NOALLOC	–38	Error in device handler, cannot allocate device in routine ****
GE_DESC_INVALID	–39	Descriptor is not acceptable in routine ****
GE_DEVPTR_INVALID	–40	Illegal device pointer in routine ****
GE_WDT_INVALID	–41	Driver handler WDT is invalid in routine ****
GE_LOGN_LWST_INVALID	–42	Logical name for the list of workstation types, GKS\$LIST_TYPES, could not be translated in routine ****
GE_NO_VWS	–43	VAX Workstation Software is not present, workstation type is invalid in routine ****
GE_VT340_COLM_ERR	–44	Error trying to save or restore VT340 color map in routine ****
GE_FA_DECOMP_ERR	–45	Unable to decompose fill area or fill area set in routine ****
GE_NODEF_CONNID	–46	No default connection identifier for specified workstation type in routine ****
GE_IE_BADMEM	–90	Internal GKS error: Bad memory address freed in routine ****
GE_IE_FCNPTR	–91	Internal GKS error: Invalid function pointer parameter in error handler in routine ****
GE_IE_VIRTMEM	–92	Internal GKS error: Insufficient virtual memory in routine ****
GE_IE_PET_NOTS	–93	Internal GKS error: Prompt and echo type not supported in routine ****
GE_IE_CORRMEM	–94	Internal GKS error: Corrupted segment memory in routine ****
GE_IE_NEG_MEMSIZE	–95	Internal GKS error: Negative size passed to allocate memory in routine ****
GE_IE_POLYGON_INVALID	–96	Internal GKS error: Illegal number of points to device handler for rectangular polygon in routine ****
GE_IE_LOGN_BUF_TOO_SMALL	–97	Internal GKS error: Insufficient buffer size for translated logical name in routine ****
GE_IE_LOGN_NOT_UNIQUE	–98	Internal GKS error: Too many translations of logical name in routine ****
GE_IE_FA_NUMPT_RED	–99	Internal GKS error: Unable to reduce number of points in fill area to requested limit in routine ****

(continued on next page)



**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_IE_DH_INPUT	–100	Internal GKS error: Device handler received unexpected input access in routine ****
GE_NEG_EDGE_IND	–150	Edge index is less than zero in routine ****
GE_NEG_WIDTH_SCALE_FACT	–151	Edge width scale factor is less than zero ****
GE_EGE_REP_NOT_DEF	–154	A representation for the specified edge index has not been predefined on this workstation in routine ****
GE_NEG_DISP_SPEED	–155	Display speed is less than zero in routine ****
GE_LOUDNESS_INVAL	–156	Loudness is outside range [0,1] in routine ****
GE_NEG_DURATION	–157	Duration is less than zero in routine ****
GE_GDP_INFO_INVAL	–158	GDP primitive is not defined by the supplied data in routine ****
GE_ARC_TYPE_INVAL	–159	Arc type is invalid in routine ****
GE_ESC_ODR_ARR_TOO_SM	–160	Insufficient space in escape output data record arrays in routine ****
GE_BBOX_TOO_SM	–161	Specified bounding box is too small in routine ****
GE_EDGE_IND_INVAL	–162	Edge index is invalid in routine ****
GE_EDGE_TYPE_NOTS	–163	Specified edge type is not supported on this workstation in routine ****
GE_HL_INVAL	–300	Invalid value specified for highlighting in routine ****
GE_VIS_INVAL	–301	Invalid value specified for visibility in routine ****
GE_DETECT_INVAL	–302	Invalid value specified for detectability in routine ****
GE_IDEV_ACT_ERR	–303	Input device cannot be activated due to conflict with another input device that is currently active in routine ****
GE_ECHO_SET_ERR	–304	Cannot set input device echo on due to conflict with other input devices active in the same echo area in routine ****
GE_IDEV_LOGN_ERR	–305	Error parsing GKS\$[wstype]_INPUT_DEVICES logical name due to syntax error in routine ****
GE_HPGL_THRH_INVAL	–306	The definition of GKS\$HPGL_THRESHOLD is invalid (contains nonnumeric values in routine) ****
GE_PCM_VAL_ERR	–450	Valuator device could not be used—dials are not available on the PCM device in routine ****
GE_PCM_CH_ERR	–451	Choice device could not be used—buttons are not available on the PCM device in routine ****
GE_PCM_INIT_ERR	–452	PCM initialization failed—hardware dials and buttons not available in routine ****
GE_PCM_CONFIG_ERR	–453	PCM configuration failed—valuator or choice device not initialized in routine ****
GE_GFX_OPEN_UID_ERR	–1517	Error trying to open UID database
GE_GFX_FETCH_UID_ERR	–1518	Error trying to fetch from UID database
GE_ILLEGAL_CALL	–2000	Illegal call to routine **** while GKS is in an error state

(continued on next page)

## Constants

### B.3 Error Constants

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_ASF_INVALID	–2001	Invalid value specified for ASF in routine ****
GE_CLIP_INVALID	–2002	Invalid clipping indicator in routine ****
GE_NT_PRIOR_INVALID	–2003	Invalid normalization transformation priority in routine ****
GE_VIEW_IND_INVAL	–2004	Invalid view index specified in routine ****
GE_METAF_INVAL	–2005	Invalid metafile format found in routine ****
GE_NULL_TXT_DIR	–2006	Null text direction vector found in routine ****
GE_TXT_PATH_INVAL	–2007	Invalid text path specified in routine ****
GE_TXT_ALIGN_INVAL	–2008	Invalid text character alignment specified in routine ****
GE_FAIS_INVAL	–2009	Invalid fill area interior style specified in routine ****
GE_EDGE_FL_INVAL	–2010	Invalid edge flag specified in routine ****
GE_ASF_VAL_INVALID	–2011	Invalid aspect source flag specified in routine ****
GE_NULL_PATR_REF_VECT	–2012	Null pattern reference vector found in routine ****
GE_INQ_STVAL_INVAL	–2013	Start value for inquiry is negative or too large in routine ****
GE_CLR_CTRL_FL_INVAL	–2014	Invalid clear control flag specified in routine ****
GE_UPD_CTRL_FL_INVAL	–2015	Invalid update control flag specified in routine ****
GE_DEFMODE_INVALID	–2016	Invalid deferral mode specified in routine ****
GE_REGENMODE_INVALID	–2017	Invalid implicit regeneration mode specified in routine ****
GE_V_XFORM_IP_PRIOR_INVAL	–2018	Invalid view transformation priority in routine ****
GE_TXT_PREC_INVALID	–2019	Invalid text precision specified in routine ****
GE_COOR_SW_INVAL	–2020	Invalid coordinate switch argument in routine ****
GE_PROJ_TYPE_INVAL	–2021	Invalid projection type argument in routine ****
GE_FR_BK_DIST_TOO_SM	–2022	Front plane is too near back plane for this viewport in routine ****
GE_ZLIM_INVAL	–2023	Equal window z limits not allowed with unequal viewport z limits in routine ****
GE_IDEV_MODE_INVAL	–2024	Invalid input device mode specified in routine ****
GE_IDEV_ECHO_INVALID	–2025	Invalid input device echo state specified in routine ****
GE_LN_FA_CRTL_FL_INVAL	–2026	Invalid polyline/fill area control flag in data record in routine ****
GE_ATTR_CRTL_FL_INVAL	–2027	Invalid attribute control flag in data record in routine ****
GE_PET_REC_SIZE_INVAL	–2028	Invalid data record size for specified prompt and echo type in routine ****
GE_NT_INVAL	–2029	Invalid normalization transformation number specified in routine ****
GE_NDC_OUTSIDE	–2030	NDC position(s) outside NDC unit cube in routine ****
GE_IDEV_INVAL	–2031	Invalid input device class specified in routine ****

(continued on next page)

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_VIS_INVALID	–2032	Invalid visibility flag specified in routine ****
GE_HL_INVALID	–2033	Invalid highlighting flag specified in routine ****
GE_DETECT_INVALID	–2034	Invalid detectability flag specified in routine ****
GE_MATRIX_INVAL	–2036	Invalid matrix specified, matrix has a zero fourth row in routine ****
GE_VIEW_MAT_INVAL	–2037	Warning, the specified view matrix is not invertible in routine ****
GE_3DITM_2DMETAFO	–2038	Warning, a 3D item cannot be written into GKSM (2D metafile output) in routine ****
GE_3DNT_2DMETAFO	–2039	Warning, the normalization transformation is 3D and cannot be used with GKSM (2D metafile output) in routine ****
GE_3DST_2DMETAFO	–2040	Warning, the segment transformation is 3D and cannot be used with GKSM (2D metafile output) in routine ****
GE_3DINQ_2DMETAFO	–2041	Warning, 3D inquiries are invalid with GKSM (2D metafile output) in routine ****
GE_LOGN_BUF_TOO_SM	–2500	Insufficient buffer size for translated logical name in routine ****
GE_LOGN_NOT_UNIQUE	–2501	Too many translations of logical name in routine ****
GE_ENVOPT_TRANS_ERR	–2502	System error during logical name translation in routine ****
GE_ERRF_CL_ERR	–2503	Error closing the error logging file in routine ****
GE_WSH_ACTIV_ERR	–2504	Image activation error during load of workstation handler in routine ****
GE_WFT_INVALID	–2505	Invalid WFT detected during load of workstation handler in routine ****
GE_STR_DESC_INVAL	–2506	Invalid string descriptor found in routine ****
GE_STR_DESC_BUF_TOO_SM	–2507	Insufficient buffer space to convert string descriptor in routine ****
GE_DH_ACT_ERR	–2508	Image activation error during load of device handler in routine ****
GE_DFT_INVAL	–2509	Invalid DFT detected during load of device handler in routine ****
GE_LOGN_SYSERR	–2510	System error during logical name creation in routine ****
GE_ARR_DESC_INVAL	–2511	Invalid array descriptor in routine ****
GE_INT_TRANS_STR_INVAL	–2512	Invalid representation of integer in translated string in routine ****
GE_WSTL_TRANS_ERR	–2513	Error translating workstation type list in routine ****
GE_WDT_INVAL	–2514	Invalid device handler WDT detected in routine ****
GE_BNDL_TBL_SIZE_ERR	–2515	Device handler WDT has at least one bundle table of zero size in routine ****
GE_FCN_NAME_DESC_INVAL	–2517	Invalid function name descriptor found in routine ****

(continued on next page)

## Constants

### B.3 Error Constants

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Implementation-Specific Errors:</b>		
GE_ERRF_CR_FAIL	–2518	Create operation failed on GKS error file in routine ****
GE_ERRF_OP_FAIL	–2519	Open operation failed on GKS error file in routine ****
GE_ERRF_CL_FAIL	–2520	Close operation failed on GKS error file in routine ****
GE_ERRF_DEL_FAIL	–2521	Delete operation failed on empty GKS error file in routine ****
GE_FONT_FILE_INVALID	–2522	Font file not found or unusable in routine ****
GE_STRLEN_INVALID	–2523	String length less than or equal to 0 in routine ****
GE_FCN_ID_INVAL	–2525	Invalid function identifier found in routine ****
GE_TIME_CONV_ERR	–2528	Internal error converting time to binary form in routine ****
GE_SET_TIMER_SYSERR	–2529	System error setting interval timer in routine ****
GE_EV_FL_WAIT_SYSERR	–2530	System error waiting for event flag in routine ****
GE_END_TIMER_SYSERRR	–2531	System error canceling interval timer in routine ****
GE_EV_FL_SET_SYSERR	–2532	System error setting event flag in routine ****
GE_EV_FL_GET_SYSERR	–2533	System error getting event flag in routine ****
GE_EV_FL_FREE_SYSERR	–2534	System error freeing event flag in routine ****
GE_SET_TIMER_EVH_SYSERR	–2535	System error setting timer event handler in routine ****
GE_RESET_TIMER_EVH_SYSERR	–2536	System error resetting timer event handler in routine ****
GE_IPM_ERR	–2537	Nonfatal input manager error detected in routine ****
GE_BUF_TOO_SM	–2538	Insufficient buffer space available in routine ****
GE_DH_ERR	–2539	Device handler error detected in routine ****

(continued on next page)

**Table B–3 (Cont.) C Error Constants**

Constant	Value	Description
<b>Fatal Errors:</b>		
GE_FE_INTERNAL_DATA	–3000	Internal data error detected in routine ****
GE_FE_LOOPING_CDN	–3001	Error looping condition forced emergency close in routine ****
GE_FE_CDN	–3002	Fatal error condition forced emergency close in routine ****
GE_FE_BYPASS_FBND	–3003	Illegal call to native interface, bypassing FORTRAN binding, detected in routine ****
GE_FE_BYPASS_CBND	–3004	Illegal call to native interface, bypassing C binding, detected in routine ****
GE_FE_MEMZONE_ALLOC	–3500	Memory zone allocation error, insufficient virtual memory in routine ****
GE_FE_MEMZONE_DEALLOC	–3501	Memory zone deallocation error in routine ****
GE_FE_MEM_ALLOC	–3502	Memory allocation error, insufficient virtual memory in routine ****
GE_FE_MEM_DEALLOC	–3503	Memory deallocation error in routine ****
GE_FE_MEM_REALLOC	–3504	Memory reallocation error in routine ****
GE_FE_FLARR_ALLOC	–3505	Float array allocation error, insufficient virtual memory in routine ****
GE_FE_FLARR_DEALLOC	–3506	Float array deallocation error in routine ****
GE_FE_WSH	–3508	Workstation handler error in routine ****
GE_FE_IPM	–3509	Fatal input manager error detected in routine ****
GE_FE_PETAR_TOO_SM	–3510	Internal prompt and echo type array too small in routine ****



**Program Example**

**Insert tabbed divider here. Then discard this sheet.**





---

## Program Example

This appendix provides information on binding-specific code examples contained in this manual.

### C.1 Code Examples

Table C-1 lists the code examples available throughout the binding. The code examples are listed alphabetically by function.

**Table C-1 Binding-Specific Code Examples**

Function	Example
ACCUMULATE TRANSFORMATION MATRIX	Example 7-1
ACTIVATE WORKSTATION	Example 4-1
ASSOCIATE SEGMENT WITH WORKSTATION	Example 8-1
AWAIT EVENT	Example 9-1
CELL ARRAY	Example 5-1
CLEAR WORKSTATION	Example 4-1
CLOSE GKS	Example 4-1
CLOSE SEGMENT	Example 8-1
CLOSE WORKSTATION	Example 4-1
COPY SEGMENT TO WORKSTATION	Example 8-1
CREATE SEGMENT	Example 8-1
DEACTIVATE WORKSTATION	Example 4-1
EMERGENCY CLOSE GKS	Example 12-1
ERROR HANDLING	Example 12-1
ERROR LOGGING	Example 12-1
ESCAPE	Example 4-2
EVALUATE TRANSFORMATION MATRIX	Example 7-2
FILL AREA	Example 6-1
GENERALIZED DRAWING PRIMITIVE	Example 5-2
GET LOCATOR	Example 9-1
INITIALIZE LOCATOR	Example 9-1
INITIALIZE PICK	Example 9-2
INITIALIZE STRING	Example 9-3
INITIALIZE VALUATOR	Example 9-4

(continued on next page)

## Program Example

### C.1 Code Examples

**Table C-1 (Cont.) Binding-Specific Code Examples**

<b>Function</b>	<b>Example</b>
INQUIRE DISPLAY SPACE SIZE	Example 7-4
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	Example 7-4
INQUIRE LOCATOR DEVICE STATE	Example 9-1
INQUIRE PICK DEVICE STATE	Example 9-2
INQUIRE STRING DEVICE STATE	Example 9-3
INQUIRE VALUATOR DEVICE STATE	Example 9-4
INQUIRE WORKSTATION CONNECTION AND TYPE	Example 4-2
INSERT SEGMENT	Example 8-2
MESSAGE	Example 9-1
OPEN GKS	Example 4-1
OPEN WORKSTATION	Example 4-1
POLYLINE	Example 6-3
POLYMARKER	Example 6-4
REDRAW ALL SEGMENTS ON WORKSTATION	Example 8-1
REQUEST STRING	Example 9-3
SAMPLE PICK	Example 9-2
SAMPLE VALUATOR	Example 9-4
SELECT NORMALIZATION TRANSFORMATION	Example 7-3
SET ASPECT SOURCE FLAGS	Example 6-2
SET CHARACTER HEIGHT	Example 6-4
SET CLIPPING INDICATOR	Example 7-3
SET COLOUR REPRESENTATION	Example 6-1
SET DEFERRAL STATE	Example 5-1
SET DETECTABILITY	Example 9-2
SET FILL AREA COLOUR INDEX	Example 6-1
SET FILL AREA INDEX	Example 6-2
SET FILL AREA INTERIOR STYLE	Example 6-1
SET FILL AREA REPRESENTATION	Example 6-2
SET HIGHLIGHTING	Example 8-3
SET LINETYPE	Example 6-3
SET LOCATOR MODE	Example 9-1
SET MARKER TYPE	Example 6-4
SET PICK IDENTIFIER	Example 9-2
SET PICK MODE	Example 9-2
SET POLYMARKER COLOUR INDEX	Example 6-4
SET SEGMENT TRANSFORMATION	Example 7-1
SET TEXT ALIGNMENT	Example 6-4

(continued on next page)

Table C-1 (Cont.) Binding-Specific Code Examples

Function	Example
SET TEXT PATH	Example 6-4
SET VALUATOR MODE	Example 9-4
SET VIEWPORT	Example 7-3
SET WINDOW	Example 7-3
SET WORKSTATION VIEWPORT	Example 7-4
TEXT	Example 6-4
UPDATE WORKSTATION	Example 4-1

## C.2 C Program

Example C-1 provides an example program written in C using the C binding. The program does the following:

1. Sets a trimetric parallel projection.
2. Creates three segments: a text string, a cube drawn by CELL ARRAY 3, and a cube drawn by POLYLINE 3.
3. If the workstation is of category OUTPUT (output only), a segment is rotated once, and the program exits. Otherwise, the segment is rotated until the operator's action on the input choice device is other than EXIT.

## Program Example C.2 C Program

### Example C-1 C Program Example

```
/* Header files */
#include <stdio.h>

#ifdef VMS
# include <gks.h>
#else
# include <GKS/gks.h>
#endif

/* Constant definition */

# define FALSE      0
# define TRUE       1

# define XFORM      1    /* choices */
# define QUIT       2

# define WSID       1    /* Workstation identifier */
# define TITLE_SEG  1    /* Title segment name */
# define CLIP_SEG   2    /* NDC clip bounds with polyline's segment name */
# define CUBE_SEG   3    /* Cube with cell array's segment name */

main()
{
/* Data declaration */

Gfile      *error_file;
Gint       xform_segment = TRUE;
Gint       status;
Gint       dev_num      = 1;
Gwstype    ws_type      = GWS_DEF;
Gconn      ws_conid     = GWC_DEF;
Gwscat     ws_category;

/* Viewing transformations */

Glimit3    norm_viewport;

static Gpoint3 vrp = {0.0, 0.0, 0.0}; /* View reference point */
static Gpoint3 vpn = {0.9, 0.8, 1.0}; /* View plane normal */
static Gpoint3 vuv = {0.0, 1.0, 0.0}; /* View up vector */
Gfloat        orientation[4][4]; /* View orientation matrix */
static Glimit  view_window = {-1.0, 1.0, -1.0, 1.0};
static Glimit3 proj_viewport = {0.0, 1.0, 0.0, 1.0, 0.0, 1.0};
static Gpoint3 prp = {0.0, 0.0, 10.0}; /* Projection reference point */
Gfloat        vpd = 1.0; /* View plane distance */
Gfloat        fpd = 2.0; /* Front plane distance */
Gfloat        bpd = 0.0; /* Back plane distance */
Gfloat        mapping[4][4]; /* View mapping matrix */
Glimit3      clip_limits; /* View clipping limits */
Gint         view_index = 1;

/*
 * Open graphics environment: open GKS, open and activate workstation.
 * Direct errors to the file CUBE.ERRORS.
 */
}
```

(continued on next page)

**Example C-1 (Cont.) C Program Example**

```

error_file = fopen ("CUBE.ERRORS", "w");
status = gopengks (error_file, 0);
if (status != NO_ERROR)
  { fprintf (error_file, "Error opening GKS.\n");
    gemerencyclosegks ( );
    exit (0);
  }
status = gopenws (WSID, &ws_conid, &ws_type);
if (status != NO_ERROR)
  { fprintf (error_file, "Error opening workstation.\n");
    gemerencyclosegks ( );
    exit (0);
  }
gactivatews (WSID);
gsetdeferst (WSID, GASAP, GALLOWED);
/* Inquire workstation category, so you know if workstation can do input. */
ginqwscategory (&ws_type, &ws_category, &status);
if (status != NO_ERROR)
  { fprintf (error_file, "Error inquiring workstation category.\n");
    gemerencyclosegks ( );
    exit (0);
  }
/* Turn on double buffering for DECwindows workstations. */
if ((ws_type = GKS3D$K_DECWINDOWS_OUTPUT) || (ws_type = GKS3D$K_DECWINDOWS))
  {
/* Escape to set double buffering declarations */
    Gescin   in_data_rec;
    Gint     in_data_rec_size;
    Gescout  out_data_rec;
    Gint     out_data_rec_size;
    Gint     esc_ints[3];

    in_data_rec.esc_idatarec.number_integer = 3;
    in_data_rec.esc_idatarec.number_float   = 0;
    in_data_rec.esc_idatarec.number_strings = 0;
    in_data_rec.esc_idatarec.list_integers  = esc_ints;
    esc_ints[0] = WSID;
    esc_ints[1] = 1;          /* Double buffering ON */
    esc_ints[2] = 1;          /* Redraw double buffer pixmap */
    in_data_rec_size = 16;
    gescape (ESC_ESDB, &in_data_rec, in_data_rec_size,
             &out_data_rec, &out_data_rec_size);
  }
/* If workstation can do input, initialize choice input device. */
if (ws_category == GOUTIN)
  {
/* Inquiry workstation display size declarations. */
    Gdspsize   display_size;
/* Initialize choice device declarations. */
  }

```

(continued on next page)

## Program Example C.2 C Program

### Example C-1 (Cont.) C Program Example

```
Gchoice    choice_init;
Glimit     choice_area;
Gchoicerec choice_data;
Gchar      strings[2][10];
Gchar      *choice_strings[2];
Gint       choice_str_len[2];
Gint       pet = 1;           /* Prompt and echo type */
Gint       k;

/* Determine display size. */

gingqdisplaysize (&ws_type, &display_size, &status);
if (status != NO_ERROR)
  { fprintf (error_file, "Error inquiring workstation display size.\n");
    gemergencyclosegks ( );
    exit (0);
  }

/* Position and size the choice input menu. */

choice_area.xmin = 0.1 * display_size.device.x;
choice_area.xmax = 0.3 * display_size.device.x;
choice_area.ymin = 0.7 * display_size.device.y;
choice_area.ymax = 0.9 * display_size.device.y;

/* Define menu choices. */

choice_init.status = GC_NOCHOICE;
choice_init.choice = QUIT;
strcpy(&strings[0][0], "Rotate");
strcpy(&strings[1][0], "Quit");
for (k = 0; k < QUIT; k++)
  {
    choice_strings[k] = &strings[k][0];
    choice_str_len[k] = strlen(choice_strings[k]);
  }
choice_data.choicepet1_datarec.number = QUIT;
choice_data.choicepet1_datarec.lengths = choice_str_len;
choice_data.choicepet1_datarec.strings = choice_strings;
choice_data.choicepet1_datarec.data = "Example";

ginitchoice (WSID, dev_num, &choice_init, pet, &choice_area, &choice_data);
}

/* Set default NDC transformation. */

gselntran (0);
gsetclip (GNOCLIP);

/* Set up viewing transformation for a parallel projection. */

gevalvieworienttran3 (&vrp, &vpn, &vuv, GNDC, orientation, &status);
gevalviewmaptran3 (&view_window, &proj_viewport, GPARALLEL, &prp,
                  vpd, fpd, bpd, mapping, &status);
clip_limits.xmin = clip_limits.ymin = clip_limits.zmin = 0.0;
clip_limits.xmax = clip_limits.ymax = clip_limits.zmax = 1.0;
gsetviewrep3 (WSID, view_index, orientation, mapping, &clip_limits,
             GNOCLIP, GNOCLIP, GNOCLIP);
gsetviewindex (view_index);
```

(continued on next page)

**Example C-1 (Cont.) C Program Example**

```
/*
 * Create WDSS (workstation-dependent segment storage).
 *
 * NOTE: Tests as "if (*_SEG )" are always TRUE. They are inserted
 *       only for giving examples of variable declaration used by
 *       applied output functions.
 */

if (TITLE_SEG)
{
/* Title segment declarations */

Gtxfp    font_prec;
Gchar    title_string[10];
static  Gpoint3  text_pos   = {0.1, 0.1, 1.0};
static  Gpoint3  text_dir_1 = {1.0, 0.0, 0.0};
static  Gpoint3  text_dir_2 = {0.0, 1.0,-1.0};

gcreateseg (TITLE_SEG);
font_prec.font = -15;
font_prec.prec = GP_STROKE;
gsettextfontprec (&font_prec);
gsettextcolourind (6);
gsetcharheight (0.1);
strcpy (title_string, "DEC GKS-3D");
gtext3 (&text_pos, &text_dir_1, &text_dir_2, title_string);
gcloseseg ( );
}

/* Change NDC transformation and turn clipping ON. */

norm_viewport.xmin = norm_viewport.ymin = norm_viewport.zmin = 0.2;
norm_viewport.xmax = norm_viewport.ymax = norm_viewport.zmax = 0.8;
gsetviewport3 (1, &norm_viewport);
gselntran (1);
gsetclip (GCLIP);

if (CLIP_SEG)
{
/* Data declaration for segment 2. */
```

(continued on next page)

## Program Example C.2 C Program

### Example C-1 (Cont.) C Program Example

```
Gint    *colour_array;
Gint    face;
static Gint    tmp_colour_array[3][3] = {1, 2, 3, 2, 3, 4, 3, 4, 5};
static Gidim    parallelogram_dim = {3, 3};
static Grect3    parallelogram[6] = {
    {0.3,0.3,0.3,    /* Back side*/
     0.7,0.3,0.3,
     0.3,0.7,0.3},
    {0.3,0.3,0.7,    /* Bottom side */
     0.7,0.3,0.7,
     0.3,0.3,0.3},
    {0.3,0.3,0.7,    /* Left side */
     0.3,0.3,0.3,
     0.3,0.7,0.7},
    {0.7,0.3,0.7,    /* Right side */
     0.7,0.3,0.3,
     0.7,0.7,0.7},
    {0.3,0.7,0.7,    /* Top side */
     0.7,0.7,0.7,
     0.3,0.7,0.3},
    {0.3,0.3,0.7,    /* Front side */
     0.7,0.3,0.7,
     0.3,0.7,0.7} };

gcreateseg (CLIP_SEG);
colour_array = (Gint *)tmp_colour_array;
for (face = 0; face < 6; face++)
    gcellarray3 (&parallelogram[face], &parallelogram_dim, colour_array);
gcloseseg ( );
}

/* Re-use the default NDC transformation. */
gselntran (0);
gsetclip (GNOCLIP);
if (CUBE_SEG)
{
    static Gpoint3
        front[5] = { {0.2,0.2,0.8},
                    {0.8,0.2,0.8},
                    {0.8,0.8,0.8},
                    {0.2,0.8,0.8},
                    {0.2,0.2,0.8} },
        right[5] = { {0.8,0.2,0.8},
                    {0.8,0.2,0.2},
                    {0.8,0.8,0.2},
                    {0.8,0.8,0.8},
                    {0.8,0.2,0.8} },
        top[5] = { {0.8,0.8,0.8},
                  {0.8,0.8,0.2},
                  {0.2,0.8,0.2},
                  {0.2,0.8,0.8},
                  {0.8,0.8,0.8} },
        left[5] = { {0.2,0.2,0.2},
                   {0.2,0.8,0.2},
                   {0.2,0.8,0.8},
                   {0.2,0.8,0.8},
                   {0.2,0.2,0.8} },
        bottom[5]= { {0.2,0.2,0.8},
                    {0.8,0.2,0.8},
                    {0.8,0.2,0.2},
```

(continued on next page)



Example C-1 (Cont.) C Program Example

```

                                {0.2,0.2,0.2},
                                {0.2,0.2,0.8} },
    back[5] = { {0.2,0.2,0.2},
                {0.8,0.2,0.2},
                {0.8,0.8,0.2},
                {0.2,0.8,0.2},
                {0.2,0.2,0.2} };
    gcreateseg (CUBE_SEG);
    gpolyline3 (5, bottom);
    gpolyline3 (5, back);
    gpolyline3 (5, right);
    gpolyline3 (5, left);
    gpolyline3 (5, top);
    gpolyline3 (5, front);
    gcloseseg ( );
}

/*
 * Perform segment rotation loop once.  If workstation can perform
 * input, prompt user to rotate segment again or exit.
 */

while (xform_segment)
{
/* Transformation segment declarations */
#   define    PI          3.1415926
#   define    DEG_TO_RAD  PI/180.0
#   define    ANGLE_STEP  30
#   define    MAX_TURNS   2

    Gint      angle;
    Gint      num_turns;
    Gfloat    scale_factor;
    Gpoint3   fixed_pt;
    Gpoint3   shift;
    Gangle3   rotate;
    Gscale3   scale;
    Gfloat    seg_xform[3][4];
    Gqchoice  choice_val;

/* Rotate segment using the transformation utility functions. */
    fixed_pt.x = fixed_pt.y = fixed_pt.z = 0.5;
    shift.x = shift.y = shift.z = 0.0;
    rotate.x_angle = rotate.y_angle = rotate.z_angle = 0.0;
    scale.x_scale = scale.y_scale = scale.z_scale = 0.6;
    for (num_turns = 0; num_turns < MAX_TURNS; num_turns++)
    {
        scale_factor = (num_turns & 1) ? (-0.1) : (0.1);
        for (angle = ANGLE_STEP; angle <= 360; angle += ANGLE_STEP)
        {
            rotate.z_angle = angle * DEG_TO_RAD;
            scale.x_scale = scale.y_scale = scale.z_scale += scale_factor;
            gevaltran3 (&fixed_pt, &shift, &rotate, &scale, GNDC, seg_xform);
            gsetsegtran3 (CLIP_SEG, seg_xform);
        }
    }
}

```

(continued on next page)

## Program Example C.2 C Program

### Example C-1 (Cont.) C Program Example

```
/*
 * If workstation can perform input, ask user "rotate or exit?";
 * otherwise, end the program.
 */

    if (ws_category == GOUTIN)
    {
        greqchoice (WSID, dev_num, &choice_val);
        xform_segment = (choice_val.choice == XFORM &&
                        choice_val.status == GC_OK);
    }
    else
        xform_segment = FALSE;
} /* end while (xform_segment) */

/* Close graphics environment: deactivate and close workstation, close GKS. */
gdeactivatews (WSID);
gclosews (WSID);
gclosegks ( );
}
```

**Function Identifiers**

**Insert tabbed divider here. Then discard this sheet.**



# D

## Function Identifiers

Table D–1 lists the C binding function names in the first column and their corresponding calls in the second column. The function names are arranged alphabetically.

**Table D–1 C Binding Function Identifiers**

<b>Function Name</b>	<b>C Call</b>
ACCUMULATE TRANSFORMATION MATRIX 3	gaccumtran3
ACCUMULATE TRANSFORMATION MATRIX	gaccumtran
ACTIVATE WORKSTATION	gactivatews
ASSOCIATE SEGMENT WITH WORKSTATION	gassocsegws
AWAIT EVENT	gawaitevent
CELL ARRAY 3	gcellarray3
CELL ARRAY	gcellarray
CLEAR WORKSTATION	gclearws
CLOSE GKS	gclosegks
CLOSE SEGMENT	gcloseseg
CLOSE WORKSTATION	gclosews
COPY SEGMENT TO WORKSTATION	gcopysegws
CREATE SEGMENT	gcreateseg
DEACTIVATE WORKSTATION	gdeactivatews
DELETE SEGMENT FROM WORKSTATION	gdelsegws
DELETE SEGMENT	gdelseg
EMERGENCY CLOSE GKS	gemergencyclosegks
ERROR HANDLING	gerrorhand
ERROR LOGGING	gerrorlog
ESCAPE	gescape
EVALUATE TRANSFORMATION MATRIX 3	gevaltran3
EVALUATE TRANSFORMATION MATRIX	gevaltran
EVALUATE VIEW MAPPING MATRIX 3	gevalviewmaptran3
EVALUATE VIEW ORIENTATION MATRIX 3	gevalvieworienttran3
FILL AREA 3	gfillarea3
FILL AREA	gfillarea
FILL AREA SET 3	gfillareaset3

(continued on next page)

## Function Identifiers

**Table D–1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
FILL AREA SET	gfillareaset
FLUSH DEVICE EVENTS	gflushevents
GENERALIZED DRAWING PRIMITIVE 3	ggdp3
GENERALIZED DRAWING PRIMITIVE	ggdp
GET CHOICE	ggetchoice
GET ITEM TYPE FROM GKSM	ggettypegksm
GET LOCATOR 3	ggetloc3
GET LOCATOR	ggetloc
GET PICK	ggetpick
GET STRING	ggetstring
GET STROKE 3	ggetstroke3
GET STROKE	ggetstroke
GET VALUATOR	ggetval
INITIALIZE CHOICE 3	ginitchoice3
INITIALIZE CHOICE	ginitchoice
INITIALIZE LOCATOR 3	ginitloc3
INITIALIZE LOCATOR	ginitloc
INITIALIZE PICK 3	ginitpick3
INITIALIZE PICK	ginitpick
INITIALIZE STRING 3	ginitstring3
INITIALIZE STRING	ginitstring
INITIALIZE STROKE 3	ginitstroke3
INITIALIZE STROKE	ginitstroke
INITIALIZE VALUATOR 3	ginitval3
INITIALIZE VALUATOR	ginitval
INQUIRE CHARACTER BASE VECTOR	ginqcharbase
INQUIRE CHARACTER EXPANSION FACTOR	ginqcharexpan
INQUIRE CHARACTER HEIGHT	ginqcharheight
INQUIRE CHARACTER SPACING	ginqcharspace
INQUIRE CHARACTER UP VECTOR	ginqcharup
INQUIRE CHARACTER WIDTH	ginqcharwidth
INQUIRE CHOICE DEVICE STATE 3	ginqchoicest3
INQUIRE CHOICE DEVICE STATE	ginqchoicest
INQUIRE CLIPPING	ginqclip
INQUIRE CLIPPING 3	ginqclip3
INQUIRE COLOUR FACILITIES	ginqcolourfacil
INQUIRE COLOUR MODEL FACILITIES	ginqcolourmodelfacil
INQUIRE COLOUR MODEL	ginqcolourmodel

(continued on next page)

**Table D-1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
INQUIRE COLOUR REPRESENTATION	ginqcolourep
INQUIRE CURRENT HLHSR IDENTIFIER VALUE	ginqhlhsrid
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3	ginqindivattr3
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	ginqindivattr
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	ginqcurntrannum
INQUIRE CURRENT PICK IDENTIFIER VALUE	ginqcurpickid
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3	ginqprimattr3
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	ginqprimattr
INQUIRE DEFAULT CHOICE DEVICE DATA 3	ginqdefchoice3
INQUIRE DEFAULT CHOICE DEVICE DATA	ginqdefchoice
INQUIRE DEFAULT DEFERRAL STATE VALUES	ginqdefdeferst
INQUIRE DEFAULT LOCATOR DEVICE DATA 3	ginqdefloc3
INQUIRE DEFAULT LOCATOR DEVICE DATA	ginqdefloc
INQUIRE DEFAULT PICK DEVICE DATA 3	ginqdefpick3
INQUIRE DEFAULT PICK DEVICE DATA	ginqdefpick
INQUIRE DEFAULT STRING DEVICE DATA 3	ginqdefstring3
INQUIRE DEFAULT STRING DEVICE DATA	ginqdefstring
INQUIRE DEFAULT STROKE DEVICE DATA 3	ginqdefstroke3
INQUIRE DEFAULT STROKE DEVICE DATA	ginqdefstroke
INQUIRE DEFAULT VALUATOR DEVICE DATA 3	ginqdefval3
INQUIRE DEFAULT VALUATOR DEVICE DATA	ginqdefval
INQUIRE DISPLAY SPACE SIZE 3	ginqdisplaysize3
INQUIRE DISPLAY SPACE SIZE	ginqdisplaysize
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	ginqmodsegattr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3	ginqmodwsattr3
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	ginqmodwsattr
INQUIRE EDGE COLOUR INDEX	ginqedgecolourind
INQUIRE EDGE FACILITIES	ginqedgefacil
INQUIRE EDGE FLAG	ginqedgeflag
INQUIRE EDGE REPRESENTATION	ginqedgerep
INQUIRE EDGETYPE	ginqedgetype
INQUIRE EDGEWIDTH SCALE FACTOR	ginqedgewidth

(continued on next page)

## Function Identifiers

**Table D–1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
INQUIRE FILL AREA COLOUR INDEX	ginqfillcolourind
INQUIRE FILL AREA FACILITIES	ginqfillfacil
INQUIRE FILL AREA INDEX	ginqfillind
INQUIRE FILL AREA INTERIOR STYLE INDEX	ginqfillintstyle
INQUIRE FILL AREA REPRESENTATION	ginqfillrep
INQUIRE FILL AREA STYLE INDEX	gfillstyleind
INQUIRE GENERALIZED DRAWING PRIMITIVE 3	ginqgdp3
INQUIRE GENERALIZED DRAWING PRIMITIVE	ginqgdp
INQUIRE HLHSR FACILITIES	ginqhlhsrfac
INQUIRE HLHSR MODE	ginqhlhsrmode
INQUIRE INPUT QUEUE OVERFLOW	ginqinputoverflow
INQUIRE LEVEL OF GKS	ginqlevelgks
INQUIRE LINE TYPE	ginqlinetype
INQUIRE LINE WIDTH SCALE FACTOR	ginqlinewidth
INQUIRE LIST OF ASPECT SOURCE FLAGS 3	ginqasf3
INQUIRE LIST OF ASPECT SOURCE FLAGS	ginqasf
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3	ginqavailgdp3
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	ginqavailgdp
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	ginqavailwstypes
INQUIRE LIST OF COLOUR INDICES	ginqcolourindices
INQUIRE LIST OF EDGE INDICES	ginqedgeindices
INQUIRE LIST OF FILL AREA INDICES	ginqfillindices
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	ginqntrannum
INQUIRE LIST OF PATTERN INDICES	ginqpatindices
INQUIRE LIST OF POLYLINE INDICES	ginqlineindices
INQUIRE LIST OF POLYMARKER INDICES	ginqmarkerindices
INQUIRE LIST OF TEXT INDICES	ginqtextindices
INQUIRE LIST OF VIEW INDICES	ginqviewindices
INQUIRE LOCATOR DEVICE STATE 3	ginqlocst3
INQUIRE LOCATOR DEVICE STATE	ginqlocst
INQUIRE MARKER SIZE SCALE FACTOR	ginqmarkersize
INQUIRE MARKER TYPE	ginqmarkertype
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3	ginqmaxwssttables3
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	ginqmaxwssttables

(continued on next page)



**Table D-1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	ginqmaxntrannum
INQUIRE MORE SIMULTANEOUS EVENTS	ginqmoreevents
INQUIRE NAME OF OPEN SEGMENT	ginqnameopenseg
INQUIRE NORMALIZATION TRANSFORMATION 3	ginqntran3
INQUIRE NORMALIZATION TRANSFORMATION	ginqntran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	ginqnumavailinput
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	ginqnumsegpri
INQUIRE OPERATING STATE VALUE	ginqopst
INQUIRE PATTERN FACILITIES	ginqpatfacil
INQUIRE PATTERN HEIGHT VECTOR	ginqpatheight
INQUIRE PATTERN REFERENCE POINT	ginqpatrefpt
INQUIRE PATTERN REFERENCE POINT AND VECTOR	ginqpatrefptvector
INQUIRE PATTERN REPRESENTATION	ginqpatrep
INQUIRE PATTERN WIDTH VECTOR	ginqpatwidth
INQUIRE PICK DEVICE STATE 3	ginqpickst3
INQUIRE PICK DEVICE STATE	ginqpickst
INQUIRE PIXEL	ginqpixel
INQUIRE PIXEL ARRAY	ginqpixelarray
INQUIRE PIXEL ARRAY DIMENSIONS	ginqpixelarraydim
INQUIRE POLYLINE COLOUR INDEX	ginqlinecolourind
INQUIRE POLYLINE FACILITIES	ginqlinefacil
INQUIRE POLYLINE INDEX	ginqlineind
INQUIRE POLYLINE REPRESENTATION	ginqlinerep
INQUIRE POLYMARKER COLOUR INDEX	ginqmarkercolourind
INQUIRE POLYMARKER FACILITIES	ginqmarkerfacil
INQUIRE POLYMARKER INDEX	ginqmarkerind
INQUIRE POLYMARKER REPRESENTATION	ginqmarkerrep
INQUIRE PREDEFINED COLOUR REPRESENTATION	ginqpredcolourrep
INQUIRE PREDEFINED EDGE REPRESENTATION	ginqprededgerep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	ginqpredfillrep
INQUIRE PREDEFINED PATTERN REPRESENTATION	ginqpredpatrep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	ginqpredlinerep

(continued on next page)

## Function Identifiers

**Table D–1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	ginqpredmarkerrep
INQUIRE PREDEFINED TEXT REPRESENTATION	ginqpredtextrep
INQUIRE PREDEFINED VIEW REPRESENTATION	ginqpredviewrep
INQUIRE SEGMENT ATTRIBUTES 3	ginqsegattr3
INQUIRE SEGMENT ATTRIBUTES	ginqsegattr
INQUIRE SET OF ACTIVE WORKSTATIONS	ginqactivevws
INQUIRE SET OF ASSOCIATED WORKSTATIONS	ginqassocws
INQUIRE SET OF OPEN WORKSTATIONS	ginqopenws
INQUIRE SET OF SEGMENT NAMES IN USE	ginqsegnames
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	ginqsegnamesws
INQUIRE STRING DEVICE STATE 3	ginqstringst3
INQUIRE STRING DEVICE STATE	ginqstringst
INQUIRE STROKE DEVICE STATE 3	ginqstrokest3
INQUIRE STROKE DEVICE STATE	ginqstrokest
INQUIRE TEXT ALIGNMENT	ginqtextalign
INQUIRE TEXT COLOUR INDEX	gingtextcolourind
INQUIRE TEXT EXTENT 3	gingtextextent3
INQUIRE TEXT EXTENT	gingtextextent
INQUIRE TEXT FACILITIES	gingtextfacil
INQUIRE TEXT FONT AND PRECISION	gingtextfontprec
INQUIRE TEXT INDEX	gingtextind
INQUIRE TEXT PATH	gingtextpath
INQUIRE TEXT REPRESENTATION	gingtextrep
INQUIRE VALUATOR DEVICE STATE 3	gingvalst3
INQUIRE VALUATOR DEVICE STATE	gingvalst
INQUIRE VIEW FACILITIES	gingviewfac
INQUIRE VIEW REPRESENTATION 3	gingviewrep3
INQUIRE WORKSTATION CATEGORY	gingwscategory
INQUIRE WORKSTATION CLASSIFICATION	gingwsclass
INQUIRE WORKSTATION CONNECTION AND TYPE	gingwsconntype
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	gingwsdeferupdatest
INQUIRE WORKSTATION MAXIMUM NUMBERS	gingwsmaxnum
INQUIRE WORKSTATION STATE	gingwsst
INQUIRE WORKSTATION TRANSFORMATION 3	gingwstran3

(continued on next page)

**Table D–1 (Cont.) C Binding Function Identifiers**

Function Name	C Call
INQUIRE WORKSTATION TRANSFORMATION	ginqwstran
INSERT SEGMENT 3	ginsertseg3
INSERT SEGMENT	ginsertseg
INTERPRET ITEM	ginterpret
MESSAGE	gmessage
OPEN GKS	gopengks
OPEN WORKSTATION	gopenws
POLYLINE 3	gpolyline3
POLYLINE	gpolyline
POLYMARKER 3	gpolymarker3
POLYMARKER	gpolymarker
READ ITEM FROM GKSM	greadgksm
REDRAW ALL SEGMENTS ON WORKSTATION	gredrawsegws
RENAME SEGMENT	grenameseg
REQUEST CHOICE	greqchoice
REQUEST LOCATOR 3	greqloc3
REQUEST LOCATOR	greqloc
REQUEST PICK	greqpick
REQUEST STRING	greqstring
REQUEST STROKE 3	greqstroke3
REQUEST STROKE	greqstroke
REQUEST VALUATOR	greqval
SAMPLE CHOICE	gsamplechoice
SAMPLE LOCATOR 3	gsampleloc3
SAMPLE LOCATOR	gsampleloc
SAMPLE PICK	gsamplepick
SAMPLE STRING	gsamplestring
SAMPLE STROKE 3	gsamplestroke3
SAMPLE STROKE	gsamplestroke
SAMPLE VALUATOR	gsampleval
SELECT NORMALIZATION TRANSFORMATION	gselntran
SET ASPECT SOURCE FLAGS 3	gsetasf3
SET ASPECT SOURCE FLAGS	gsetasf
SET CHARACTER EXPANSION FACTOR	gsetcharexpan
SET CHARACTER HEIGHT	gsetcharheight
SET CHARACTER SPACING	gsetcharspace
SET CHARACTER UP VECTOR	gsetcharup
SET CHOICE MODE	gsetchoicemode

(continued on next page)

## Function Identifiers

**Table D–1 (Cont.) C Binding Function Identifiers**

<b>Function Name</b>	<b>C Call</b>
SET CLIPPING INDICATOR	gsetclip
SET COLOUR MODEL	gsetcolourmodel
SET COLOUR REPRESENTATION	gsetcolourrep
SET DEFERRAL STATE	gsetdeferst
SET DETECTABILITY	gsetdet
SET EDGE COLOUR INDEX	gsetedgecolourind
SET EDGE FLAG	gsetedgeflag
SET EDGE INDEX	gsetedgeindex
SET EDGE REPRESENTATION	gsetedgerep
SET EDGETYPE	gsetedgetype
SET EDGEWIDTH SCALE FACTOR	gsetedgewidthscfac
SET FILL AREA COLOUR INDEX	gsetfillcolourind
SET FILL AREA INDEX	gsetfillind
SET FILL AREA INTERIOR STYLE	gsetfillintstyle
SET FILL AREA REPRESENTATION	gsetfillrep
SET FILL AREA STYLE INDEX	gsetfillstyleind
SET HIGHLIGHTING	gsethighlight
SET HLHSR IDENTIFIER	gsethlhsrid
SET HLHSR MODE	gsethlhsrmode
SET LINETYPE	gsetlinetype
SET LINEWIDTH SCALE FACTOR	gsetlinewidth
SET LOCATOR MODE	gsetlocmode
SET MARKER SIZE SCALE FACTOR	gsetmarkersize
SET MARKER TYPE	gsetmarkertype
SET PATTERN REFERENCE POINT AND VECTORS	gsetpatrefptvec
SET PATTERN REFERENCE POINT	gsetpatrefpt
SET PATTERN REPRESENTATION	gsetpatrep
SET PATTERN SIZE	gsetpatsize
SET PICK IDENTIFIER	gsetpickid
SET PICK MODE	gsetpickmode
SET POLYLINE COLOUR INDEX	gsetlinecolourind
SET POLYLINE INDEX	gsetlineind
SET POLYLINE REPRESENTATION	gsetlinerep
SET POLYMARKER COLOUR INDEX	gsetmarkercolourind
SET POLYMARKER INDEX	gsetmarkerind
SET POLYMARKER REPRESENTATION	gsetmarkerrep
SET SEGMENT PRIORITY	gsetsegpri
SET SEGMENT TRANSFORMATION 3	gsetsegtran3

(continued on next page)

Table D–1 (Cont.) C Binding Function Identifiers

Function Name	C Call
SET SEGMENT TRANSFORMATION	gsetsegtran
SET STRING MODE	gsetstringmode
SET STROKE MODE	gsetstrokemode
SET TEXT ALIGNMENT	gsettextalign
SET TEXT COLOUR INDEX	gsettextcolourind
SET TEXT FONT AND PRECISION	gsettextfontprec
SET TEXT INDEX	gsettextind
SET TEXT PATH	gsettextpath
SET TEXT REPRESENTATION	gsettextrep
SET VALUATOR MODE	gsetvalmode
SET VIEW INDEX	gsetviewindex
SET VIEW REPRESENTATION 3	gsetviewrep3
SET VIEW TRANSFORMATION INPUT PRIORITY	gsetviewxformpr
SET VIEWPORT 3	gsetviewport3
SET VIEWPORT	gsetviewport
SET VIEWPORT INPUT PRIORITY	gsetviewportinputpri
SET VISIBILITY	gsetvis
SET WINDOW 3	gsetwindow3
SET WINDOW	gsetwindow
SET WORKSTATION VIEWPORT 3	gsetwsviewport3
SET WORKSTATION VIEWPORT	gsetwsviewport
SET WORKSTATION WINDOW 3	gsetwswindow3
SET WORKSTATION WINDOW	gsetwswindow
TEXT 3	gtext3
TEXT	gtext
UPDATE WORKSTATION	gupdatews
WRITE ITEM TO GKSM	gwritegksm



**Initial Attribute Values**

**Insert tabbed divider here. Then discard this sheet.**





---

## Initial Attribute Values

This appendix lists the initial values of all output attributes and normalization transformation settings according to the following categories:

- Polyline attributes
- Polymarker attributes
- Text attributes
- Fill area attributes
- Fill area set attributes
- Segment attributes
- Normalization transformation settings

### E.1 Initial Polyline Attributes

This section lists the initial values for the polyline attributes.

Attribute	Initial Value	Description
Polyline index	1	Polyline bundle number 1
Line type	GLN_SOLID	Solid line
Line width	1.0	Minimum width
Line color index	1	Workstation dependent value
Line type ASF	GINDIVIDUAL	Use current line type
Line width ASF	GINDIVIDUAL	Use current line width
Line color index ASF	GINDIVIDUAL	Use current line color index

## Initial Attribute Values

### E.2 Initial Polymarker Attributes

### E.2 Initial Polymarker Attributes

This section lists the initial values for the polymarker attributes.

Attribute	Initial Value	Description
Polymarker index	1	Polymarker bundle number 1
Marker type	GMK_STAR	Asterisk for marker
Marker size	1.0	Nominal size
Color index	1	Workstation dependent value
Marker type ASF	GINDIVIDUAL	Use current marker type
Marker size ASF	GINDIVIDUAL	Use current marker size
Marker color index ASF	GINDIVIDUAL	Use current marker color index

### E.3 Initial Text Attributes

This section lists the initial values for the text attributes.

Attribute	Initial Value	Description
Text index	1	Text bundle number 1
Text font and precision	1 GP_STRING	Hardware font 1, string precision
Character expansion factor	1.0	Width-to-height ratio from font file
Character spacing	0.0	Adjacent character bodies
Color index	1	Workstation dependent value
Text font and precision ASF	GINDIVIDUAL	Use current font and precision
Character expansion factor ASF	GINDIVIDUAL	Use current width and height ratio
Character spacing ASF	GINDIVIDUAL	Use current character space
Text color index ASF	GINDIVIDUAL	Use current text color index
Character height	0.01	Capital letters at 0.01 WC units
Character up vector	(0, 1)	Up vector parallel to y-axis in WC units
Text path	GTP_RIGHT	Right angle clockwise from up vector
Text alignment	GAH_NORMAL GAV_NORMAL	Natural alignment with respect to text path

## E.4 Initial Fill Area Attributes

This section lists the initial values for the fill area attributes.

<b>Attribute</b>	<b>Initial Value</b>	<b>Description</b>
Fill area index	1	Fill area bundle number 1
Interior style	GHOLLOW	Boundary of polygonal area
Style index	1	Workstation-dependent pattern or hatch style
Color index	1	Workstation-dependent value
Interior style ASF	GINDIVIDUAL	Use current interior style
Style index ASF	GINDIVIDUAL	Use current pattern or hatch style
Color index ASF	GINDIVIDUAL	Use current fill area color index
Pattern size	1.0,1.0	Unit square in WC units
Pattern reference point	(0.0, 0.0, 0.0)	Pattern starting point in WC units

## E.5 Initial Fill Area Set Attributes

This section lists the initial values for the fill area set attributes.

<b>Attribute</b>	<b>Initial Value</b>	<b>Description</b>
Fill area index	1	Fill area bundle number 1
Interior style	GHOLLOW	Boundary of polygonal area
Fill area style index	1	Workstation-dependent pattern or hatch style
Fill area color index	1	Workstation-dependent value
Fill area interior style ASF	GINDIVIDUAL	Use current interior style
Fill area style index ASF	GINDIVIDUAL	Use current pattern or hatch style
Fill area color index ASF	GINDIVIDUAL	Use current fill area color index
Pattern size	1.0,1.0	Unit square in WC units
Pattern reference point	(0.0, 0.0, 0.0)	Pattern starting point in WC units
Edge index	1	Use edge bundle number 1
Edge flag	GEDGE_OFF	Edge not drawn
Edge type	GLN_SOLID	Solid edge
Edge width scale factor	1.0	Minimum width
Edge color index	1	Workstation-dependent value
Edge flag ASF	GINDIVIDUAL	Use current edge flag

## Initial Attribute Values

### E.5 Initial Fill Area Set Attributes

Attribute	Initial Value	Description
Edge type ASF	GINDIVIDUAL	Use current edge type
Edge width scale factor ASF	GINDIVIDUAL	Use current edge width scale factor
Edge color index ASF	GINDIVIDUAL	Use current edge color index

### E.6 Initial Segment Attributes

This section lists the initial values for the segment attributes.

Attribute	Initial Value	Description
Transformation number	0	The identity transformation presents the segment as stored in NDC space.
Visibility	GVISIBLE	The segment is visible.
Highlighting	GNORMAL	The segment is not highlighted.
Segment priority	0.0	The segment has the lowest priority.
Detectability	GUNDETECTABLE	The segment is undetectable.

The default segment transformation is called the identity transformation. The identity transformation uses a  $3 \times 4$  matrix as follows:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

### E.7 Initial Normalization Transformation Settings

The initial normalization transformation number is the value 0.

The initial viewport input priority is in sequential order from the values 0 to 255, with transformation number 0 the highest and 255 the lowest.

The default normalization window and viewport limits are cubic, begin with a lower left corner point of (0.0, 0.0, 0.0), and extend to the value (1.0, 1.0, 1.0) on the X, Y, and Z axes.

Initially, clipping is enabled (GCLIP) at the normalization viewport limit.

**Differences in GKS**

**Implementations**

**Insert tabbed divider here. Then discard this sheet.**



---

## Differences in GKS Implementations

The GKS standard omits a lot details about how to implement certain functionality. This allows different GKS implementations the freedom to adapt to different environments and requirements.

The allowed differences in GKS implementations can be divided into two groups:

- Global differences
- Workstation-dependent differences

This appendix lists the global differences allowed by the standard, and the corresponding DEC GKS implementation. Workstation-dependent differences are listed in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.

### F.1 Global Differences

The global differences relate to the implementation as a whole. The allowable global differences are listed in Table F–1, along with the DEC GKS-specific values. The differences are grouped into three main categories: functional scope, capacity, and miscellaneous.

**Table F–1 Global Differences**

Description	DEC GKS Value
<b>Functional Scope</b>	
GKS level	DEC GKS is level 2c.
<b>Capacity</b>	
Number of available workstation types	There are 63 workstation types, not including the default type. The workstation types include MO (metafile output), MI (metafile input), and WISS (workstation-independent segment storage).
List of available workstation types	See the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for the list of supported workstation types.
Maximum number of simultaneously open workstations	The maximum is 256.
Maximum number of simultaneously active workstations	The maximum is 256.

(continued on next page)

## Differences in GKS Implementations

### F.1 Global Differences

**Table F–1 (Cont.) Global Differences**

Description	DEC GKS Value
<b>Capacity</b>	
Maximum number of workstations associated with a segment	The maximum is 256.
Maximum normalization transformation number	The maximum is 255.
Number of simultaneously definable segments (per workstation)	The maximum number is 128.
Maximum size of input queue	The maximum size is 256 characters.
Number of fonts available	DEC GKS has 25 device-independent fonts, in addition to device-dependent fonts. See the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for information on fonts.
Number of GDPs	DEC GKS has 31 GDPs. See the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for more information on the supported GDPs.
Number of escapes	DEC GKS has 61 escape functions. See the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for more information on the supported escapes.
<b>Miscellaneous</b>	
Initial setting of ASFs	DEC GKS initially sets all the ASFs to INDIVIDUAL.
EMERGENCY CLOSE GKS behavior	The EMERGENCY CLOSE GKS function closes the segment, if it is open, deactivates any active workstations, closes all open workstations, then closes GKS.
Actions performed on parameters of inquiry functions if the information is unavailable	The information returned in a DEC GKS inquiry function is only valid if the value of the error indicator is 0.
Metafile format used by each workstation type of category MO	See the <i>DEC GKS User's Guide</i> for a detailed description of the metafile format.
Font definitions	See the <i>Device Specifics Reference Manual for DEC GKS and DEC PHIGS</i> for complete information on fonts supported by DEC GKS.



## A

---

### ACCUMULATE TRANSFORMATION MATRIX 3

function, *Part 1*, 7–12

### ACCUMULATE TRANSFORMATION MATRIX

function, *Part 1*, 7–10

example, *Part 1*, 7–37

### Accumulating

segment transformations, *Part 1*, 8–9

### ACTIVATE WORKSTATION function, *Part 1*, 4–9

example, *Part 1*, 4–24

### Activating workstations, *Part 1*, 4–5

### Alignment

text, *Part 1*, 6–51

### Angles

See also Segments

rotation, *Part 1*, 8–8

### ANSI

CGM standard, *Part 1*, 10–1

GKS standard, *Part 1*, 1–1

### Appearance

attributes, *Part 1*, 6–1

### Arguments

characteristics of, *Part 1*, 1–5

descriptions, *Part 1*, 1–5

inquiry error status, *Part 2*, 11–3

inquiry value type argument, *Part 2*, 11–4

### Arrays

color index, *Part 1*, 5–4, 5–6

### ASAP, *Part 1*, 1–7

### ASFs, *Part 1*, 6–4

### Aspect ratio

See also Transformations

### ASSOCIATE SEGMENT WITH WORKSTATION

function, *Part 1*, 8–11

example, *Part 1*, 8–28

### Association

See also Segments

segments, *Part 1*, 8–3

windows and viewports, *Part 1*, 7–5

### Asynchronous input, *Part 1*, 9–14

See also Input

### Attribute functions, *Part 1*, 6–1 to 6–70

introduction to, *Part 1*, 6–1 to 6–5

### Attributes, *Part 1*, 1–3

attribute source flags, *Part 1*, 6–4

bound to primitives, *Part 1*, 6–1

### Attributes (cont'd)

bundled, *Part 1*, 6–3

fill area, *Part 1*, 6–2

fill area set, *Part 1*, 6–3

GDPs, *Part 1*, 6–3

geometric and nongeometric, *Part 1*, 6–1

implicit regenerations, *Part 1*, 6–4

segments, *Part 1*, 8–3

individual, *Part 1*, 6–3

initial values, *Part 2*, E–1 to E–4

input prompt and echo types, *Part 1*, 9–5

list of errors, *Part 2*, A–6 to A–10

metafiles, *Part 1*, 10–2

pick identification, *Part 1*, 8–2

polyline, *Part 1*, 6–2

polymarker, *Part 1*, 6–2

segments, *Part 1*, 8–5

text, *Part 1*, 6–2

### Attribute source flags, *Part 1*, 6–4

### Audit metafiles, *Part 1*, 10–1

### AWAIT EVENT function, *Part 1*, 9–17, 9–22

example, *Part 1*, 9–94

### Axes, *Part 1*, 7–1

See also Coordinates

See also Segments

segment fixed point, *Part 1*, 8–7

## B

---

### Background

color, *Part 1*, 6–5

### Binding

attributes to primitives, *Part 1*, 6–1

### Boundaries, *Part 1*, 7–8

See Windows or Viewports

### Break input, *Part 1*, 9–15

### Buffers

See also Data records

See also Input

input data record, *Part 1*, 9–8

string input, *Part 1*, 9–4

stroke input, *Part 1*, 9–4

### Bundles, *Part 1*, 6–3

See also Attributes

edge, *Part 1*, 6–22

fill area, *Part 1*, 6–27, 6–29

pattern styles, *Part 1*, 6–40

polyline, *Part 1*, 6–44, 6–45

## Bundles (cont'd)

- polymarker, *Part 1*, 6-48, 6-49
- text, *Part 1*, 6-56, 6-58

## C

---

### Calls

- error handler, *Part 2*, 12-1
- function
  - reproducing, *Part 1*, 10-2

### Categories

- See also Workstations
- of functions, *Part 1*, 1-1
- workstations, *Part 1*, 4-2
  - list of, *Part 1*, 4-2

### C binding

- list of constants, *Part 2*, B-1 to B-39

### C binding errors, *Part 2*, A-19

### C binding files

- VMS, *Part 1*, 2-1, 3-1

### CELL ARRAY 3 function, *Part 1*, 5-6

### CELL ARRAY function, *Part 1*, 5-4

- example, *Part 1*, 5-24

### Cell arrays, *Part 1*, 5-6

### CGM metafiles

- ANSI standard, *Part 1*, 10-1
- creating, *Part 1*, 10-3 to 10-4

### Change vectors

- input, *Part 1*, 9-5
- segment translation, *Part 1*, 8-7

### Characters

- height, *Part 1*, 6-12
- strings, *Part 1*, 5-20, 5-22

### Choice

- See also Input
- input class, *Part 1*, 9-4
- specifying NOCHOICE input, *Part 1*, 9-15, 9-16

### Classes

- See also Input
- See also Logical input devices
- choice, *Part 1*, 9-4
- locator, *Part 1*, 9-4
- pick, *Part 1*, 9-4
- string, *Part 1*, 9-4
- stroke, *Part 1*, 9-4
- valuator, *Part 1*, 9-4

### Cleanup

- error handling, *Part 2*, 12-1

### Clearing

- See also Workstations
- workstation surface, *Part 1*, 4-10
  - implicit regeneration, *Part 1*, 4-7

### CLEAR WORKSTATION function, *Part 1*, 4-10

- example, *Part 1*, 4-24

### Clipping, *Part 1*, 7-4, 7-7

- See also Transformations
- disable, *Part 1*, 7-4
- enable, *Part 1*, 7-4
- segments, *Part 1*, 8-9
- text precision, *Part 1*, 6-54

### Clipping flag

- initial value, *Part 2*, E-4

### CLOSE GKS function, *Part 1*, 4-11

- example, *Part 1*, 4-24

### CLOSE SEGMENT function, *Part 1*, 8-12

- example, *Part 1*, 8-28

### CLOSE WORKSTATION function, *Part 1*, 4-12

- example, *Part 1*, 4-24

### Closing

- See also GKS
- See also Workstations
- GKS, *Part 1*, 4-5
  - error handling, *Part 2*, 12-1
  - segments, *Part 1*, 4-5
  - workstations, *Part 1*, 4-5

### Color

- See also Attributes
- background, *Part 1*, 6-5
- fill area, *Part 1*, 6-26
- foreground, *Part 1*, 6-5
- indexes
  - arrays, *Part 1*, 5-4
  - 3D arrays, *Part 1*, 5-6
- markers, *Part 1*, 6-47
- model, *Part 1*, 6-16
- polyline, *Part 1*, 6-43
- representation, *Part 1*, 6-17
- text, *Part 1*, 6-53

### Compiling

- ULTRIX programs, *Part 1*, 3-1
- VMS programs, *Part 1*, 2-2

### Completion states, *Part 2*, A-1

### Components

- See also Rotation
- See also Scale
- See also Translation
- segment transformations, *Part 1*, 8-7

### Composition

- See also Transformations
- picture, *Part 1*, 1-3, 7-1

### Conditions

- error, *Part 2*, 12-1, A-1 to A-37

### Configuration files, *Part 1*, 3-7

- customizing
  - system level, *Part 1*, 3-7
  - user level, *Part 1*, 3-7

### Connection identifiers

- metafiles, *Part 1*, 10-1, 10-4
- specifying on ULTRIX, *Part 1*, 3-2
- specifying on VMS, *Part 1*, 2-2

## Constants, *Part 2*, B-1

- action pending states, *Part 2*, B-12
- aspect source flags, *Part 2*, B-1
- attribute control flags, *Part 2*, B-1
- choice input prompt flags, *Part 2*, B-1
- choice status types, *Part 2*, B-1
- clear screen states, *Part 2*, B-1
- clipping flags, *Part 2*, B-2
- color availability flags, *Part 2*, B-2
- color models, *Part 2*, B-2
- coordinate switch, *Part 2*, B-2
- default connection identifier, *Part 2*, B-2
- deferral modes, *Part 2*, B-2
- detectability flags, *Part 2*, B-2
- device coordinate units, *Part 2*, B-2
- display surface states, *Part 2*, B-3
- dynamic modification states, *Part 2*, B-3
- echo states, *Part 2*, B-3
- edge flags, *Part 2*, B-3
- edge types, *Part 2*, B-3
- error
  - attribute function, *Part 2*, B-27 to B-29
  - C language-dependent, *Part 2*, B-32
  - 3D, *Part 2*, B-31 to B-32
  - escape function, *Part 2*, B-31
  - fatal, *Part 2*, B-39
  - implementation-specific, *Part 2*, B-32 to B-38
  - input function, *Part 2*, B-30
  - metafile function, *Part 2*, B-30
  - miscellaneous, *Part 2*, B-31
  - operating state, *Part 2*, B-26
  - output function, *Part 2*, B-29
  - segment function, *Part 2*, B-29
  - system, *Part 2*, B-31
  - transformation function, *Part 2*, B-27
  - workstation, *Part 2*, B-26
- error handling modes, *Part 2*, B-3
- escape function identifiers, *Part 2*, B-4
- fill area control flags, *Part 2*, B-5
- fill area interior styles, *Part 2*, B-5
- GDP bundle types, *Part 2*, B-6
- GDP graphics primitives, *Part 2*, B-6
- GKS level types, *Part 2*, B-7
- GKS operating states, *Part 2*, B-7
- highlighting flags, *Part 2*, B-8
- highlighting methods, *Part 2*, B-8
- HLHSR identifiers, *Part 2*, B-8
- HLHSR modes, *Part 2*, B-8
- horizontal alignment types, *Part 2*, B-11
- implicit regeneration states, *Part 2*, B-8
- input classes, *Part 2*, B-8
- input mode types, *Part 2*, B-9
- input priority states, *Part 2*, B-9
- invalid index flags, *Part 2*, B-9
- last event flag, *Part 2*, B-9
- line cap styles, *Part 2*, B-9
- line join styles, *Part 2*, B-9

## Constants (cont'd)

- line types (implementation-specific), *Part 2*, B-10
- line types (standard), *Part 2*, B-9
- list of, *Part 2*, B-1 to B-39
- marker types (implementation-specific), *Part 2*, B-10
- marker types (standard), *Part 2*, B-10
- memory size, *Part 2*, B-10
- new frame action necessary states, *Part 2*, B-10
- pick status types, *Part 2*, B-11
- projection types, *Part 2*, B-11
- prompt flags, *Part 2*, B-1
- regeneration flag states, *Part 2*, B-11
- request status types, *Part 2*, B-11
- requirements, *Part 1*, 2-1, 3-1
- returned type values, *Part 2*, B-11
- simultaneous events flags, *Part 2*, B-11
- text horizontal alignment types, *Part 2*, B-11
- text path types, *Part 2*, B-11
- text precision types, *Part 2*, B-12
- text vertical alignment types, *Part 2*, B-12
- update states, *Part 2*, B-12
- vertical alignment types, *Part 2*, B-12
- viewport priority states, *Part 2*, B-12
- visibility flags, *Part 2*, B-12
- workstation category types, *Part 2*, B-12
- workstation class types, *Part 2*, B-13
- workstation color availability states, *Part 2*, B-13
- workstation states, *Part 2*, B-13
- workstation types, *Part 2*, B-13
- writing modes, *Part 2*, B-16

## Control

- error handling, *Part 2*, 12-1
- workstation surface, *Part 1*, 4-6

## Control functions, *Part 1*, 4-1 to 4-32

- introduction to, *Part 1*, 4-1 to 4-8
- metafiles, *Part 1*, 10-2

## Coordinates

- See also Transformations
- input change vectors, *Part 1*, 9-5
- locator and stroke input, *Part 1*, 9-4
- maximum device, *Part 1*, 7-8
- systems, *Part 1*, 7-1
  - used for output, *Part 1*, 5-2
- viewport input priority, *Part 1*, 7-6, 9-18

## Copying segments, *Part 1*, 8-3

## COPY SEGMENT TO WORKSTATION function, *Part 1*, 8-13

- example, *Part 1*, 8-28

## CREATE SEGMENT function, *Part 1*, 8-14

- example, *Part 1*, 8-28

## Creating

- metafiles, *Part 1*, 10-1
- segments, *Part 1*, 4-5, 8-1

Current  
See also Transformations  
metafile item, *Part 1*, 10–4  
windows and viewports, *Part 1*, 7–8  
*Current event report* entry, *Part 1*, 9–17  
See also Event mode  
See also Input  
Cycling  
disabled input echo, *Part 1*, 9–14  
logical input device control, *Part 1*, 9–14

## D

---

Data  
user defined  
metafiles, *Part 1*, 10–2  
Data records  
See also Escapes  
See also Input  
input, *Part 1*, 9–8  
prompt and echo types, *Part 1*, 9–5 to 9–13  
sizes, *Part 1*, 9–19  
standard, *Part 1*, 9–8  
using inquiry functions, *Part 1*, 9–19  
metafile  
item, *Part 1*, 10–2  
Data structures  
See also GKS  
DEACTIVATE WORKSTATION function, *Part 1*,  
4–13  
example, *Part 1*, 4–24  
Deactivating  
See also Workstations  
workstations, *Part 1*, 4–5  
Defaults  
See also Attributes  
See also Transformations  
colors, *Part 1*, 6–5  
GKS-3D error handler, *Part 2*, 12–4  
identity segment transformation, *Part 1*, 8–7  
normalization window, *Part 1*, 7–3  
unity transformation, *Part 1*, 7–4  
Deferral  
See also Implicit regenerations  
DECwindows, *Part 1*, 1–6  
output, *Part 1*, 4–6, 5–3  
Definition file, *Part 1*, 2–1  
Definition files  
including, *Part 1*, 2–1, 3–1  
Degrees  
See also GDPs  
See also Segments  
translating to radians, *Part 1*, 8–8  
DELETE SEGMENT FROM WORKSTATION  
function, *Part 1*, 8–16

DELETE SEGMENT function, *Part 1*, 8–15  
Deleting segments, *Part 1*, 8–2  
Descriptions  
functions, *Part 1*, 1–4  
Description tables, *Part 1*, 4–1  
GKS, *Part 2*, 11–1  
workstation, *Part 2*, 11–1  
Detecting  
errors, *Part 2*, 12–1  
segments, *Part 1*, 8–5  
Device  
transformations, *Part 1*, 7–7 to 7–8  
Device coordinates, *Part 1*, 7–1  
See also Transformations  
See also Workstations  
Device dependent  
bundled attributes, *Part 1*, 6–3  
Device independent  
attributes, *Part 1*, 6–1  
Device-independent programming  
input, *Part 1*, 9–20  
Device number, *Part 1*, 9–1  
Devices  
See also Workstations  
logical input, *Part 1*, 9–1 to 9–3  
manipulation  
ESCAPE, *Part 1*, 4–14  
maximum coordinate values, *Part 1*, 7–8  
physical input, *Part 1*, 9–1  
Display  
See also Workstations  
surface, *Part 1*, 7–1  
surface control, *Part 1*, 7–1  
CLEAR WORKSTATION, *Part 1*, 4–10  
REDRAW ALL SEGMENTS ON  
WORKSTATION, *Part 1*, 4–20  
SET DEFERRAL STATE, *Part 1*, 4–21  
UPDATE WORKSTATION, *Part 1*, 4–23  
Dynamic modification  
See also Implicit regenerations  
attributes, *Part 1*, 4–7  
workstation transformations, *Part 1*, 4–7

## E

---

Echo  
See also Input  
cycling and disabled echo, *Part 1*, 9–14  
input values, *Part 1*, 9–2, 9–3, 9–14  
prompt and echo types, *Part 1*, 9–5 to 9–13  
Echo area, *Part 1*, 9–2  
Edge  
index, *Part 1*, 6–21  
representation, *Part 1*, 6–22  
type, *Part 1*, 6–24  
width scale factor, *Part 1*, 6–25

- Emergency
  - closure of GKS, *Part 2*, 12-1
- EMERGENCY CLOSE GKS function, *Part 2*, 12-3
  - example, *Part 2*, 12-6
- Ending
  - GKS program, *Part 1*, 4-11
- Entries
  - See also GKS
  - bundle table, *Part 1*, 6-3
- Environment
  - GKS, *Part 1*, 4-1
  - workstation, *Part 1*, 4-1
- Environment variables, *Part 1*, 3-3
  - default file, *Part 1*, 3-4
  - defining
    - at *cs*h, *Part 1*, 3-3
    - at *sh*, *Part 1*, 3-3
    - in file, *Part 1*, 3-3
  - GKS*sasf*, *Part 1*, 3-5
  - GKS*sconid*, *Part 1*, 3-2, 3-5
  - .GKS*defaults*, *Part 1*, 3-4
  - GKS*defmode*, *Part 1*, 3-5
  - GKS*serrfile*, *Part 1*, 3-5, 3-6
  - GKS*error*, *Part 1*, 3-5
  - GKS*sirg*, *Part 1*, 3-5
  - GKS*metafile\_type*, *Part 1*, 3-5
  - GKS*ndc\_clip*, *Part 1*, 3-6
  - GKS*stroke\_font1*, *Part 1*, 3-6
  - GKS*swstype*, *Part 1*, 3-6
  - search order, *Part 1*, 3-4
  - stderr*, *Part 1*, 3-6
  - system defaults file, *Part 1*, 3-2
  - types, *Part 1*, 3-5
    - general, *Part 1*, 3-5
  - user defaults file, *Part 1*, 3-2
- Error codes
  - defined, *Part 1*, 2-4, 3-6
  - ULTRIX, *Part 1*, 3-6
  - VMS, *Part 1*, 2-4
- Error files
  - default, *Part 1*, 2-4
  - defined, *Part 1*, 3-6
  - ULTRIX, *Part 1*, 3-6
  - VMS, *Part 1*, 2-4
- Error handling, *Part 1*, 2-4 to 2-5, 3-6
  - GKS, *Part 1*, 1-4
- ERROR HANDLING function, *Part 2*, 12-4
- Error-handling functions
  - gemergencyclosegks*, *Part 2*, 12-3
  - gerrorhand*, *Part 2*, 12-4
  - gerrorlog*, *Part 2*, 12-5
  - introduction to, *Part 2*, 12-1 to 12-2
- ERROR LOGGING function, *Part 2*, 12-5
  - example, *Part 2*, 12-6
- Errors
  - constants
    - attribute function, *Part 2*, B-27 to B-29
- Errors
  - constants (cont'd)
    - C language-dependent, *Part 2*, B-32
    - 3D, *Part 2*, B-31 to B-32
    - escape function, *Part 2*, B-31
    - fatal, *Part 2*, B-39
    - implementation-specific, *Part 2*, B-32 to B-38
    - input function, *Part 2*, B-30
    - metafile function, *Part 2*, B-30
    - miscellaneous, *Part 2*, B-31
    - operating state, *Part 2*, B-26
    - output function, *Part 2*, B-29
    - segment function, *Part 2*, B-29
    - system, *Part 2*, B-31
    - transformation function, *Part 2*, B-27
    - workstation, *Part 2*, B-26 to B-27
  - file, *Part 2*, 12-2
  - inquiry error status argument, *Part 2*, 11-3
  - logging, *Part 1*, 4-4; *Part 2*, 12-5
  - messages, *Part 2*, A-1 to A-37
    - attributes, *Part 2*, A-6 to A-10
    - C binding, *Part 2*, A-19
    - escapes, *Part 2*, A-15
    - fatal, *Part 2*, A-36 to A-37
    - implementation-specific, *Part 2*, A-19 to A-35
    - input, *Part 2*, A-12 to A-14
    - metafiles, *Part 2*, A-14 to A-15
    - miscellaneous, *Part 2*, A-15 to A-16
    - operating state, *Part 2*, A-1 to A-2
    - output, *Part 2*, A-10 to A-11
    - segments, *Part 2*, A-11 to A-12
    - system, *Part 2*, A-16 to A-19
    - transformations, *Part 2*, A-5 to A-6
    - workstation, *Part 2*, A-3 to A-5
  - states, *Part 2*, 12-1
- Error status files
  - list of, *Part 1*, 2-1, 3-1
- ESCAPE function, *Part 1*, 4-14
  - example, *Part 1*, 4-26
- Escapes
  - list of errors, *Part 2*, A-15
- EVALUATE TRANSFORMATION MATRIX 3
  - function, *Part 1*, 7-16
- EVALUATE TRANSFORMATION MATRIX
  - function, *Part 1*, 7-14
  - example, *Part 1*, 7-41
- EVALUATE VIEW MAPPING MATRIX 3 function, *Part 1*, 7-18
- EVALUATE VIEW ORIENTATION MATRIX 3
  - function, *Part 1*, 7-21
- Event functions, *Part 1*, 9-16
- Event input queue, *Part 1*, 9-16
  - overflow, *Part 1*, 9-17
- Event mode, *Part 1*, 9-16 to 9-18
  - See also Input
  - cycling devices, *Part 1*, 9-14

## Examples

- list of functions, *Part 2*, C-1
- table of, *Part 2*, C-1

## Executing

- ULTRIX programs, *Part 1*, 3-1
- VMS programs, *Part 1*, 2-2

## Expansion

- See also Scale
- See also Segments
- segments, *Part 1*, 8-7
- text, *Part 1*, 6-11

## Extent rectangle

- See also Attributes
- See also Segments
- See also Text
- segments
  - highlighting, *Part 1*, 8-6

## F

---

### Fatal errors, *Part 2*, 12-1

- list of, *Part 2*, A-36 to A-37

### File

- definition, *Part 1*, 2-1

### Files

- error, *Part 2*, 12-2
- error status
  - list of, *Part 1*, 2-1, 3-1

### File specifications

- metafiles, *Part 1*, 10-1

### FILL AREA 3 function, *Part 1*, 5-9

### FILL AREA function, *Part 1*, 5-8

- example, *Part 1*, 6-60

### Fill areas

- See also Attributes
  - attributes
    - SET FILL AREA COLOUR INDEX, *Part 1*, 6-26, 6-28
    - SET FILL AREA INDEX, *Part 1*, 6-27
    - SET FILL AREA STYLE INDEX, *Part 1*, 6-31
    - SET PATTERN REFERENCE POINT, *Part 1*, 6-38
    - SET PATTERN SIZE, *Part 1*, 6-41
  - bundles, *Part 1*, 6-27
  - 2D, *Part 1*, 5-8
  - 3D, *Part 1*, 5-9
  - initial attributes, *Part 2*, E-3
  - interior styles, *Part 1*, 6-28
  - representation, *Part 1*, 6-29
  - style indexes, *Part 1*, 6-31
- ### Fill area set, *Part 1*, 5-10, 5-11
- ### FILL AREA SET 3 function, *Part 1*, 5-11
- ### FILL AREA SET function, *Part 1*, 5-10
- ### Fill area sets
- initial attributes, *Part 2*, E-3 to E-4

## Fixed points

- See also Rotation
- See also Scale
- See also Segments
- segment transformations, *Part 1*, 8-7

## Flags

- See also Attributes
- ASF, *Part 1*, 6-7, 6-9
- aspect source, *Part 1*, 6-4
- edge flag, *Part 1*, 6-20

## Flush

- event queue, *Part 1*, 9-17
- FLUSH DEVICE EVENTS, *Part 1*, 9-17, 9-18
- FLUSH DEVICE EVENTS function, *Part 1*, 9-24

## Fonts

- establishing, *Part 1*, 6-54

## Foreground color, *Part 1*, 6-5

## Format

- function descriptions, *Part 1*, 1-4
- metafiles, *Part 1*, 10-2

## Function

- constants, *Part 1*, 1-6
- data structures, *Part 1*, 1-6
- description, *Part 1*, 1-6
- header, *Part 1*, 1-4
- identifiers, *Part 2*, B-16 to B-25
  - attribute, *Part 2*, B-17
  - control, *Part 2*, B-16
  - error handling, *Part 2*, B-25
  - input, *Part 2*, B-19
  - inquiry, *Part 2*, B-21 to B-25
  - metafile, *Part 2*, B-21
  - output, *Part 2*, B-17
  - segment, *Part 2*, B-19
  - transformation, *Part 2*, B-18
- operating states, *Part 1*, 1-5
- presentation, *Part 1*, 1-4 to 1-7
- Program Examples sections, *Part 1*, 1-6
- See Also sections, *Part 1*, 1-6
- syntax, *Part 1*, 1-5

## Functional standards

- See also GKS

## Functions

- See also GKS
- attribute, *Part 1*, 6-6
- control, *Part 1*, 4-8
- DEC GKS categories, *Part 1*, 1-1
- error-handling, *Part 2*, 12-1 to 12-2
- input, *Part 1*, 9-21
- inquiry, *Part 2*, 11-5 to 11-203
- metafile, *Part 1*, 10-5
- output, *Part 1*, 5-3
- segment, *Part 1*, 8-10
- transformation, *Part 1*, 7-9

## G

- gaccumtran*, *Part 1*, 7–10  
*gaccumtran3*, *Part 1*, 7–12  
*gactivatews*, *Part 1*, 4–9  
*gassocsegws*, *Part 1*, 8–11  
*gawaitevent*, *Part 1*, 9–22  
*gcellarray*, *Part 1*, 5–4  
*gcellarray3*, *Part 1*, 5–6  
*gclearws*, *Part 1*, 4–10  
*gclosegks*, *Part 1*, 4–11  
*gcloseseg*, *Part 1*, 8–12  
*gclosews*, *Part 1*, 4–12  
*gcopysegws*, *Part 1*, 8–13  
*gcreateseg*, *Part 1*, 8–14  
*gdeactivatews*, *Part 1*, 4–13  
*gdelseg*, *Part 1*, 8–15  
*gdelsegws*, *Part 1*, 8–16  
GDPs, *Part 1*, 5–12, 5–14  
    attributes, *Part 1*, 6–3  
*gemergencyclosegks*, *Part 2*, 12–3  
GENERALIZED DRAWING PRIMITIVE 3  
    function, *Part 1*, 5–14  
GENERALIZED DRAWING PRIMITIVE function,  
    *Part 1*, 5–12  
    example, *Part 1*, 5–26  
Generalized drawing primitives  
    See GDPs  
Generation  
    See also Output  
    output, *Part 1*, 5–1  
        attributes, *Part 1*, 6–1  
        pictures, *Part 1*, 7–1  
Geometric attributes, *Part 1*, 6–1  
*gerrorhand*, *Part 2*, 12–4  
*gerrorlog*, *Part 2*, 12–5  
*gescape*, *Part 1*, 4–14  
GET CHOICE function, *Part 1*, 9–25  
GET functions, *Part 1*, 9–16  
GET ITEM TYPE FROM GKSM function, *Part 1*,  
    10–6  
GET ITEM TYPE FROM METAFILE  
    See GET ITEM TYPE FROM GKSM function  
GET LOCATOR 3 function, *Part 1*, 9–27  
GET LOCATOR function, *Part 1*, 9–26  
    example, *Part 1*, 9–94  
GET PICK function, *Part 1*, 9–28  
GET STRING function, *Part 1*, 9–29  
GET STROKE 3 function, *Part 1*, 9–32  
GET STROKE function, *Part 1*, 9–30  
GET VALUATOR function, *Part 1*, 9–34  
*gevaltran*, *Part 1*, 7–14  
*gevaltran3*, *Part 1*, 7–16  
*gevalviewmaptran3*, *Part 1*, 7–18  
*gevalvieworienttran3*, *Part 1*, 7–21  
*gfillarea*, *Part 1*, 5–8  
*gfillarea3*, *Part 1*, 5–9  
*gfillareaset*, *Part 1*, 5–10  
*gfillareaset3*, *Part 1*, 5–11  
*gflushevents*, *Part 1*, 9–24  
*ggdp*, *Part 1*, 5–12  
*ggdp3*, *Part 1*, 5–14  
*ggetchoice*, *Part 1*, 9–25  
*ggetloc*, *Part 1*, 9–26  
*ggetloc3*, *Part 1*, 9–27  
*ggetpick*, *Part 1*, 9–28  
*ggetstring*, *Part 1*, 9–29  
*ggetstroke*, *Part 1*, 9–30  
*ggetstroke3*, *Part 1*, 9–32  
*ggettypegksm*, *Part 1*, 10–6  
*ggetval*, *Part 1*, 9–34  
*ginitchoice*, *Part 1*, 9–35  
*ginitchoice3*, *Part 1*, 9–37  
*ginitloc*, *Part 1*, 9–39  
*ginitloc3*, *Part 1*, 9–43  
*ginitpick*, *Part 1*, 9–47  
*ginitpick3*, *Part 1*, 9–49  
*ginitstring*, *Part 1*, 9–51  
*ginitstring3*, *Part 1*, 9–53  
*ginitstroke*, *Part 1*, 9–55  
*ginitstroke3*, *Part 1*, 9–58  
*ginitval*, *Part 1*, 9–61  
*ginitval3*, *Part 1*, 9–63  
*ginqactivews*, *Part 2*, 11–159  
*ginqasf*, *Part 2*, 11–6  
*ginqasf3*, *Part 2*, 11–7  
*ginqassocws*, *Part 2*, 11–160  
*ginqavailgdp*, *Part 2*, 11–91  
*ginqavailgdp3*, *Part 2*, 11–92  
*ginqavailwstypes*, *Part 2*, 11–93  
*ginqcharbase*, *Part 2*, 11–8  
*ginqcharexpan*, *Part 2*, 11–9  
*ginqcharheight*, *Part 2*, 11–10  
*ginqcharspace*, *Part 2*, 11–11  
*ginqcharup*, *Part 2*, 11–12  
*ginqcharwidth*, *Part 2*, 11–13  
*ginqchoicest*, *Part 2*, 11–14  
*ginqchoicest3*, *Part 2*, 11–17  
*ginqclip*, *Part 2*, 11–20  
*ginqclip3*, *Part 2*, 11–21  
*ginqcolourfacil*, *Part 2*, 11–22  
*ginqcolourindices*, *Part 2*, 11–94  
*ginqcolourmodel*, *Part 2*, 11–23  
*ginqcolourmodelfacil*, *Part 2*, 11–24  
*ginqcolourrep*, *Part 2*, 11–25  
*ginqcurntrannum*, *Part 2*, 11–32  
*ginqcurpickid*, *Part 2*, 11–33  
*ginqdefchoice*, *Part 2*, 11–38  
*ginqdefchoice3*, *Part 2*, 11–40  
*ginqdefdeferst*, *Part 2*, 11–42

ginqdefloc, *Part 2*, 11–43  
 ginqdefloc3, *Part 2*, 11–45  
 ginqdefpick, *Part 2*, 11–47  
 ginqdefpick3, *Part 2*, 11–49  
 ginqdefstring, *Part 2*, 11–51  
 ginqdefstring3, *Part 2*, 11–53  
 ginqdefstroke, *Part 2*, 11–55  
 ginqdefstroke3, *Part 2*, 11–57  
 ginqdefval, *Part 2*, 11–59  
 ginqdefval3, *Part 2*, 11–61  
 ginqdisplaysize, *Part 2*, 11–63  
 ginqdisplaysize3, *Part 2*, 11–64  
 ginqedgecolourind, *Part 2*, 11–71  
 ginqedgefacil, *Part 2*, 11–72  
 ginqedgeflag, *Part 2*, 11–73  
 ginqedgeindices, *Part 2*, 11–95  
 ginqedgerep, *Part 2*, 11–74  
 ginqedgetype, *Part 2*, 11–75  
 ginqedgewidth, *Part 2*, 11–76  
 ginqfillcolourind, *Part 2*, 11–77  
 ginqfillfacil, *Part 2*, 11–78  
 ginqfillind, *Part 2*, 11–79  
 ginqfillindices, *Part 2*, 11–96  
 ginqfillintstyle, *Part 2*, 11–80  
 ginqfillrep, *Part 2*, 11–81  
 ginqfillstyleind, *Part 2*, 11–82  
 ginqgdp, *Part 2*, 11–83  
 ginqgdp3, *Part 2*, 11–84  
 ginqhlhsrfac, *Part 2*, 11–85  
 ginqhlhsrid, *Part 2*, 11–26  
 ginqhlhsrmode, *Part 2*, 11–86  
 ginqindivattr, *Part 2*, 11–27  
 ginqindivattr3, *Part 2*, 11–29  
 ginqinputoverflow, *Part 2*, 11–87  
 ginqlevelgks, *Part 2*, 11–88  
 ginqlinecolourind, *Part 2*, 11–139  
 ginqlinefacil, *Part 2*, 11–140  
 ginqlineind, *Part 2*, 11–141  
 ginqlineindices, *Part 2*, 11–99  
 ginqlinerep, *Part 2*, 11–142  
 ginqlinetype, *Part 2*, 11–89  
 ginqlinewidth, *Part 2*, 11–90  
 ginqlocst, *Part 2*, 11–103  
 ginqlocst3, *Part 2*, 11–108  
 ginqmarkercolourind, *Part 2*, 11–143  
 ginqmarkerfacil, *Part 2*, 11–144  
 ginqmarkerind, *Part 2*, 11–145  
 ginqmarkerindices, *Part 2*, 11–100  
 ginqmarkerrep, *Part 2*, 11–146  
 ginqmarkersize, *Part 2*, 11–113  
 ginqmarkertype, *Part 2*, 11–114  
 ginqmaxntrannum, *Part 2*, 11–117  
 ginqmaxwssttables, *Part 2*, 11–115  
 ginqmaxwssttables3, *Part 2*, 11–116  
 ginqmodsegattr, *Part 2*, 11–66  
 ginqmodwsattr, *Part 2*, 11–67  
 ginqmodwsattr3, *Part 2*, 11–69  
 ginqmoreevents, *Part 2*, 11–118  
 ginqnameopenseg, *Part 2*, 11–119  
 ginqntran, *Part 2*, 11–120  
 ginqntran3, *Part 2*, 11–121  
 ginqntrannum, *Part 2*, 11–97  
 ginqnumavailinput, *Part 2*, 11–122  
 ginqnumsegpri, *Part 2*, 11–123  
 ginqopenws, *Part 2*, 11–161  
 ginqopst, *Part 2*, 11–124  
 ginqpatfacil, *Part 2*, 11–125  
 ginqpatheight, *Part 2*, 11–126  
 ginqpatindices, *Part 2*, 11–98  
 ginqpatrefpt, *Part 2*, 11–127  
 ginqpatrefptvector, *Part 2*, 11–128  
 ginqpatrep, *Part 2*, 11–129  
 ginqpatwidth, *Part 2*, 11–130  
 ginqpickst, *Part 2*, 11–131  
 ginqpickst3, *Part 2*, 11–133  
 ginqpixel, *Part 2*, 11–135  
 ginqpixelarray, *Part 2*, 11–136  
 ginqpixelarraydim, *Part 2*, 11–138  
 ginqpredcolourrep, *Part 2*, 11–147  
 ginqprededgerep, *Part 2*, 11–148  
 ginqpredfillrep, *Part 2*, 11–149  
 ginqpredlinerep, *Part 2*, 11–151  
 ginqpredmarkerrep, *Part 2*, 11–152  
 ginqpredpatrep, *Part 2*, 11–150  
 ginqpredtextrep, *Part 2*, 11–153  
 ginqpredviewrep, *Part 2*, 11–154  
 ginqprimattr, *Part 2*, 11–34  
 ginqprimattr3, *Part 2*, 11–36  
 ginqsegattr, *Part 2*, 11–155  
 ginqsegattr3, *Part 2*, 11–157  
 ginqsegnames, *Part 2*, 11–162  
 ginqsegnamesws, *Part 2*, 11–163  
 ginqstringst, *Part 2*, 11–164  
 ginqstringst3, *Part 2*, 11–166  
 ginqstrokest, *Part 2*, 11–168  
 ginqstrokest3, *Part 2*, 11–171  
 ginqtextalign, *Part 2*, 11–175  
 ginqtextcolourind, *Part 2*, 11–176  
 ginqtexttextent, *Part 2*, 11–177  
 ginqtexttextent3, *Part 2*, 11–178  
 ginqtextfacil, *Part 2*, 11–179  
 ginqtextfontprec, *Part 2*, 11–181  
 ginqtextind, *Part 2*, 11–182  
 ginqtextindices, *Part 2*, 11–101  
 ginqtextpath, *Part 2*, 11–183  
 ginqtextrep, *Part 2*, 11–184  
 ginqvalst, *Part 2*, 11–186  
 ginqvalst3, *Part 2*, 11–188  
 ginqviewfac, *Part 2*, 11–190  
 ginqviewindices, *Part 2*, 11–102  
 ginqviewrep3, *Part 2*, 11–191  
 ginqwscategory, *Part 2*, 11–193



- ginqwsclass, *Part 2*, 11–194
- ginqwsconntype, *Part 2*, 11–195
- ginqwsdeferupdatest, *Part 2*, 11–196
- ginqwsmaxnum, *Part 2*, 11–198
- ginqwsst, *Part 2*, 11–199
- ginqwstran, *Part 2*, 11–200
- ginqwstran3, *Part 2*, 11–202
- ginsertseg, *Part 1*, 8–17
- ginsertseg3, *Part 1*, 8–19
- ginterpret, *Part 1*, 10–7
- GKS
  - ANSI and ISO standards, *Part 1*, 1–1
  - categories of functions, *Part 1*, 1–1
  - closing, *Part 1*, 4–5
  - description table, *Part 1*, 4–1
  - environment, *Part 1*, 4–1
  - error handling, *Part 1*, 1–4; *Part 2*, 12–1
  - input
    - levels of, *Part 1*, 1–4
  - introduction to, *Part 1*, 1–1 to 1–4
  - kernel, *Part 1*, 4–1
  - levels, *Part 1*, 1–4
  - metafile standard, *Part 1*, 10–1
  - opening, *Part 1*, 4–4
  - operating state
    - errors, *Part 2*, A–1 to A–2
  - output
    - levels of, *Part 1*, 1–4
- GKS\$ASF, *Part 1*, 2–3
- GKS\$CONID, *Part 1*, 2–3
- GKS\$DEF\_MODE, *Part 1*, 2–3
- GKS\$ERRFILE, *Part 1*, 2–4
- GKS\$ERROR, *Part 1*, 2–4
- GKS\$IRG, *Part 1*, 2–4
- GKS\$METAFILE\_TYPE, *Part 1*, 2–4
- GKS\$NDC\_CLIP, *Part 1*, 2–4
- GKS\$STROKE\_FONT1, *Part 1*, 2–4
- GKS\$WSTYPE, *Part 1*, 2–4
- GKS3 metafiles
  - creating, *Part 1*, 10–1 to 10–3
- GKSasf, *Part 1*, 3–5
- gksconfig.c, *Part 1*, 3–7
- GKSconid, *Part 1*, 3–5
- GKSdefmode, *Part 1*, 3–5
- GKS environment functions, *Part 1*, 4–11, 4–18
- GKSerrfile, *Part 1*, 3–5
- GKSerror, *Part 1*, 3–5
- GKSirg, *Part 1*, 3–5
- GKSmetafile\_type, *Part 1*, 3–5
- GKSM metafiles, *Part 1*, 10–1
  - creating, *Part 1*, 10–1 to 10–3
- GKSndc\_clip, *Part 1*, 3–6
- GKSstroke\_font1, *Part 1*, 3–6
- GKSswstype, *Part 1*, 3–6
- gks\_decw\_config.c, *Part 1*, 3–7
- gmessage, *Part 1*, 4–17
- gopengks, *Part 1*, 4–18
- gopenws, *Part 1*, 4–19
- gpolyline, *Part 1*, 5–16
- gpolyline3, *Part 1*, 5–17
- gpolymarker, *Part 1*, 5–18
- gpolymarker3, *Part 1*, 5–19
- Graphics handlers, *Part 1*, 4–1
  - See also Devices
  - See also Workstations
  - input, *Part 1*, 9–5
    - nominal sizes, *Part 1*, 6–1
- greadgksm, *Part 1*, 10–8
- gredrawsegws, *Part 1*, 4–20
- grenameseg, *Part 1*, 8–21
- greqchoice, *Part 1*, 9–65
- greqloc, *Part 1*, 9–66
- greqloc3, *Part 1*, 9–68
- greqpick, *Part 1*, 9–70
- greqstring, *Part 1*, 9–71
- greqstroke, *Part 1*, 9–73
- greqstroke3, *Part 1*, 9–75
- greqval, *Part 1*, 9–77
- gsamplechoice, *Part 1*, 9–78
- gsampleloc, *Part 1*, 9–79
- gsampleloc3, *Part 1*, 9–80
- gsamplepick, *Part 1*, 9–81
- gsamplestring, *Part 1*, 9–82
- gsamplestroke, *Part 1*, 9–83
- gsamplestroke3, *Part 1*, 9–85
- gsampleval, *Part 1*, 9–87
- gselntran, *Part 1*, 7–23
- gsetasf, *Part 1*, 6–7
- gsetasf3, *Part 1*, 6–9
- gsetcharexpan, *Part 1*, 6–11
- gsetcharheight, *Part 1*, 6–12
- gsetcharspace, *Part 1*, 6–13
- gsetcharup, *Part 1*, 6–14
- gsetchoicemode, *Part 1*, 9–88
- gsetclip, *Part 1*, 7–24
- gsetcolourmodel, *Part 1*, 6–16
- gsetcolourrep, *Part 1*, 6–17
- gsetdeferst, *Part 1*, 4–21
- gsetdet, *Part 1*, 8–22
- gsetedgecolourind, *Part 1*, 6–19
- gsetedgeflag, *Part 1*, 6–20
- gsetedgeindex, *Part 1*, 6–21
- gsetedgerep, *Part 1*, 6–22
- gsetedgetype, *Part 1*, 6–24
- gsetedgewidthscfac, *Part 1*, 6–25
- gsetfillcolourind, *Part 1*, 6–26
- gsetfillind, *Part 1*, 6–27
- gsetfillintstyle, *Part 1*, 6–28
- gsetfillrep, *Part 1*, 6–29
- gsetfillstyleind, *Part 1*, 6–31
- gsethighlight, *Part 1*, 8–23

gsethlhsrid, *Part 1*, 6–32  
 gsethlhsrmode, *Part 1*, 6–33  
 gsetlinecolourind, *Part 1*, 6–43  
 gsetlineind, *Part 1*, 6–44  
 gsetlinerep, *Part 1*, 6–45  
 gsetlinetype, *Part 1*, 6–34  
 gsetlinewidth, *Part 1*, 6–35  
 gsetlocmode, *Part 1*, 9–89  
 gsetmarkercolourind, *Part 1*, 6–47  
 gsetmarkerind, *Part 1*, 6–48  
 gsetmarkerrep, *Part 1*, 6–49  
 gsetmarkersize, *Part 1*, 6–36  
 gsetmarkertype, *Part 1*, 6–37  
 gsetpatrefpt, *Part 1*, 6–38  
 gsetpatrefptvec, *Part 1*, 6–39  
 gsetpatrep, *Part 1*, 6–40  
 gsetpatsize, *Part 1*, 6–41  
 gsetpickid, *Part 1*, 6–42  
 gsetpickmode, *Part 1*, 9–90  
 gsetsegpri, *Part 1*, 8–24  
 gsetsegtran, *Part 1*, 8–25  
 gsetsegtran3, *Part 1*, 8–26  
 gsetstringmode, *Part 1*, 9–91  
 gsetstrokemode, *Part 1*, 9–92  
 gsettextalign, *Part 1*, 6–51  
 gsettextcolourind, *Part 1*, 6–53  
 gsettextfontprec, *Part 1*, 6–54  
 gsettextind, *Part 1*, 6–56  
 gsettextpath, *Part 1*, 6–57  
 gsettextrep, *Part 1*, 6–58  
 gsetvalmode, *Part 1*, 9–93  
 gsetviewindex, *Part 1*, 7–25  
 gsetviewport, *Part 1*, 7–28  
 gsetviewport3, *Part 1*, 7–29  
 gsetviewportinputpri, *Part 1*, 7–30  
 gsetviewrep3, *Part 1*, 7–26  
 gsetviewxformpr, *Part 1*, 7–27  
 gsetvis, *Part 1*, 8–27  
 gsetwindow, *Part 1*, 7–31  
 gsetwindow3, *Part 1*, 7–32  
 gsetwsviewport, *Part 1*, 7–33  
 gsetwsviewport3, *Part 1*, 7–34  
 gsetwswindow, *Part 1*, 7–35  
 gsetwswindow3, *Part 1*, 7–36  
 gtext, *Part 1*, 5–20  
 gtext3, *Part 1*, 5–22  
 gupdatews, *Part 1*, 4–23  
 gwritegksm, *Part 1*, 10–9

## H

### Handlers

See also Devices  
 See also Workstations  
 See Graphics handlers  
 errors, *Part 2*, 12–1  
 set and realized values, *Part 2*, 11–4

Hardware fonts, *Part 1*, 6–54

See also Fonts

Hatches, *Part 1*, 6–28

See also Fill areas

fill areas, *Part 1*, 5–8, 5–9

style index values, *Part 1*, 6–31

Height

See also Attributes

See also Transformations

of text, *Part 1*, 6–12

Highlighting

segments, *Part 1*, 8–6

Hollow

fill area interior style, *Part 1*, 6–28

fill areas, *Part 1*, 5–8, 5–9

## I

### Identifiers

function, *Part 2*, B–16

attribute, *Part 2*, B–17

control, *Part 2*, B–16

error handling, *Part 2*, B–25

input, *Part 2*, B–19

inquiry, *Part 2*, B–21 to B–25

metafile, *Part 2*, B–21

output, *Part 2*, B–17

segment, *Part 2*, B–19

transformation, *Part 2*, B–18

pick, *Part 1*, 8–2, 9–4

Identity

segment transformation, *Part 1*, 8–8

transformation, *Part 1*, 7–7

Implementation-specific errors

list of, *Part 2*, A–19 to A–35

Implicit regenerations, *Part 1*, 4–7

See also Deferral

attribute changes, *Part 1*, 6–4

segments, *Part 1*, 8–3

workstation transformations, *Part 1*, 7–8

Include

definition files, *Part 1*, 2–1, 3–1

INCLUDE statement

all languages, *Part 1*, 2–1, 3–1

Index

See also Attributes

See also Bundles

color, *Part 1*, 6–16

arrays, *Part 1*, 5–4

3D arrays, *Part 1*, 5–6

edge, *Part 1*, 6–22

edge color index, *Part 1*, 6–19

fill area, *Part 1*, 6–27, 6–29

styles, *Part 1*, 6–31

into bundle tables, *Part 1*, 6–3

pattern styles, *Part 1*, 6–40

polyline, *Part 1*, 6–45

## Index (cont'd)

- polymarker, *Part 1*, 6–49
- text, *Part 1*, 6–56, 6–58

Individual attributes, *Part 1*, 6–3

Initialize

- See also GKS
- See also Workstations
- GKS environment, *Part 1*, 4–18
- input functions
  - INITIALIZE CHOICE, *Part 1*, 9–35
  - INITIALIZE CHOICE 3, *Part 1*, 9–37
  - INITIALIZE LOCATOR, *Part 1*, 9–39
  - INITIALIZE LOCATOR 3, *Part 1*, 9–43
  - INITIALIZE PICK, *Part 1*, 9–47
  - INITIALIZE PICK 3, *Part 1*, 9–49
  - INITIALIZE STRING, *Part 1*, 9–51
  - INITIALIZE STRING 3, *Part 1*, 9–53
  - INITIALIZE STROKE 3, *Part 1*, 9–55, 9–58
  - INITIALIZE VALUATOR, *Part 1*, 9–61
  - INITIALIZE VALUATOR 3, *Part 1*, 9–63
- workstation environment, *Part 1*, 4–19

INITIALIZE CHOICE 3 function, *Part 1*, 9–37

INITIALIZE CHOICE function, *Part 1*, 9–35

INITIALIZE functions, *Part 1*, 9–3

INITIALIZE LOCATOR 3 function, *Part 1*, 9–43

INITIALIZE LOCATOR function, *Part 1*, 9–39

- example, *Part 1*, 9–94

INITIALIZE PICK 3 function, *Part 1*, 9–49

INITIALIZE PICK function, *Part 1*, 9–47

- example, *Part 1*, 9–98

INITIALIZE STRING 3 function, *Part 1*, 9–53

INITIALIZE STRING function, *Part 1*, 9–51

- example, *Part 1*, 9–104

INITIALIZE STROKE 3 function, *Part 1*, 9–58

INITIALIZE STROKE function, *Part 1*, 9–55

INITIALIZE VALUATOR 3 function, *Part 1*, 9–63

INITIALIZE VALUATOR function, *Part 1*, 9–61

- example, *Part 1*, 9–107

Initializing a logical input device, *Part 1*, 9–3

Initializing input, *Part 1*, 9–13

Initial string

- input, *Part 1*, 9–4

Input

- asynchronous, *Part 1*, 9–14
- breaking, *Part 1*, 9–15
- classes, *Part 1*, 9–1, 9–4
- current values, *Part 1*, 9–19
- cycling device control, *Part 1*, 9–14
- data record
  - sizes, *Part 1*, 9–19
  - standard, *Part 1*, 9–8
  - using inquiry functions, *Part 1*, 9–19
- default values, *Part 1*, 9–19
- device-independent programming, *Part 1*, 9–20
- event mode, *Part 1*, 9–16 to 9–18
  - flushing the queue, *Part 1*, 9–17
- event queue, *Part 1*, 9–16

## Input (cont'd)

- event queue overflow, *Part 1*, 9–17
- initializing, *Part 1*, 9–13
- inquiry function use, *Part 1*, 9–19
- list of errors, *Part 2*, A–12 to A–14
- menus, *Part 1*, 9–4
- metafiles, *Part 1*, 10–1, 10–2
- operating modes, *Part 1*, 9–2, 9–3, 9–14 to 9–18
- pick
  - visibility, *Part 1*, 8–10
- pick identification, *Part 1*, 8–2
- request mode, *Part 1*, 9–14 to 9–15
- sample mode, *Part 1*, 9–15 to 9–16
- segment detectability, *Part 1*, 8–5
- segments, *Part 1*, 8–2
- specifying no input, *Part 1*, 9–15
- synchronous, *Part 1*, 9–14
- text, *Part 1*, 9–4
- triggers, *Part 1*, 9–3, 9–15
- viewport priority, *Part 1*, 7–6, 9–18
- workstation categories, *Part 1*, 4–2

Input data records

- sizes, *Part 1*, 9–19

Input functions, *Part 1*, 9–1 to 9–112

- introduction to, *Part 1*, 9–1 to 9–18

Input operating modes, *Part 1*, 9–14

Input priority

- initial value, *Part 2*, E–4

INQUIRE ASPECT SOURCE FLAGS 3 function, *Part 2*, 11–7

INQUIRE ASPECT SOURCE FLAGS function, *Part 2*, 11–6

INQUIRE CHARACTER BASE VECTOR function, *Part 2*, 11–8

INQUIRE CHARACTER EXPANSION FACTOR function, *Part 2*, 11–9

INQUIRE CHARACTER HEIGHT function, *Part 2*, 11–10

INQUIRE CHARACTER SPACING function, *Part 2*, 11–11

INQUIRE CHARACTER UP VECTOR function, *Part 2*, 11–12

INQUIRE CHARACTER WIDTH function, *Part 2*, 11–13

INQUIRE CHOICE DEVICE STATE 3 function, *Part 2*, 11–17

INQUIRE CHOICE DEVICE STATE function, *Part 2*, 11–14

INQUIRE CLIPPING 3 function, *Part 2*, 11–21

INQUIRE CLIPPING function, *Part 2*, 11–20

INQUIRE COLOUR FACILITIES function, *Part 2*, 11–22

INQUIRE COLOUR MODEL FACILITIES function, *Part 2*, 11–24

INQUIRE COLOUR MODEL function, *Part 2*, 11-23

INQUIRE COLOUR REPRESENTATION function, *Part 2*, 11-25

INQUIRE CURRENT HLHSR IDENTIFIER VALUE function, *Part 2*, 11-26

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3 function, *Part 2*, 11-29

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function, *Part 2*, 11-27

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER function, *Part 2*, 11-32

INQUIRE CURRENT PICK IDENTIFIER VALUE function, *Part 2*, 11-33

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3 function, *Part 2*, 11-36

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function, *Part 2*, 11-34

INQUIRE DEFAULT CHOICE DEVICE DATA 3 function, *Part 2*, 11-40

INQUIRE DEFAULT CHOICE DEVICE DATA function, *Part 2*, 11-38

INQUIRE DEFAULT DEFERRAL STATE VALUES function, *Part 2*, 11-42

INQUIRE DEFAULT LOCATOR DEVICE DATA 3 function, *Part 2*, 11-45

INQUIRE DEFAULT LOCATOR DEVICE DATA function, *Part 2*, 11-43

INQUIRE DEFAULT PICK DEVICE DATA 3 function, *Part 2*, 11-49

INQUIRE DEFAULT PICK DEVICE DATA function, *Part 2*, 11-47

INQUIRE DEFAULT STRING DEVICE DATA 3 function, *Part 2*, 11-53

INQUIRE DEFAULT STRING DEVICE DATA function, *Part 2*, 11-51

INQUIRE DEFAULT STROKE DEVICE DATA 3 function, *Part 2*, 11-57

INQUIRE DEFAULT STROKE DEVICE DATA function, *Part 2*, 11-55

INQUIRE DEFAULT VALUATOR DEVICE DATA 3 function, *Part 2*, 11-61

INQUIRE DEFAULT VALUATOR DEVICE DATA function, *Part 2*, 11-59

INQUIRE DISPLAY SPACE SIZE 3 function, *Part 2*, 11-64

INQUIRE DISPLAY SPACE SIZE function, *Part 2*, 11-63  
example, *Part 1*, 7-50

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES function, *Part 2*, 11-66

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3 function, *Part 2*, 11-69

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function, *Part 2*, 11-67  
example, *Part 1*, 7-50

INQUIRE EDGE COLOUR INDEX function, *Part 2*, 11-71

INQUIRE EDGE FACILITIES function, *Part 2*, 11-72

INQUIRE EDGE FLAG function, *Part 2*, 11-73

INQUIRE EDGE REPRESENTATION function, *Part 2*, 11-74

INQUIRE EDGETYPE function, *Part 2*, 11-75

INQUIRE EDGEWIDTH SCALE FACTOR function, *Part 2*, 11-76

INQUIRE FILL AREA COLOUR INDEX function, *Part 2*, 11-77

INQUIRE FILL AREA FACILITIES function, *Part 2*, 11-78

INQUIRE FILL AREA INDEX function, *Part 2*, 11-79

INQUIRE FILL AREA INTERIOR STYLE function, *Part 2*, 11-80

INQUIRE FILL AREA REPRESENTATION function, *Part 2*, 11-81

INQUIRE FILL AREA STYLE INDEX function, *Part 2*, 11-82

INQUIRE GENERALIZED DRAWING PRIMITIVE 3 function, *Part 2*, 11-84

INQUIRE GENERALIZED DRAWING PRIMITIVE function, *Part 2*, 11-83

INQUIRE HLHSR FACILITIES function, *Part 2*, 11-85

INQUIRE HLHSR MODE function, *Part 2*, 11-86

INQUIRE INPUT QUEUE OVERFLOW function, *Part 1*, 9-17; *Part 2*, 11-87

INQUIRE LEVEL OF GKS function, *Part 2*, 11-88

INQUIRE LINETYPE function, *Part 2*, 11-89

INQUIRE LINewidth SCALE FACTOR function, *Part 2*, 11-90

INQUIRE LIST OF ASPECT SOURCE FLAGS  
See INQUIRE ASPECT SOURCE FLAGS function

INQUIRE LIST OF ASPECT SOURCE FLAGS 3  
See INQUIRE ASPECT SOURCE FLAGS 3 function

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 function, *Part 2*, 11-92

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES function, *Part 2*, 11-91

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES function, *Part 2*, 11-93

INQUIRE LIST OF COLOUR INDICES function, *Part 2*, 11–94

INQUIRE LIST OF EDGE INDICES function, *Part 2*, 11–95

INQUIRE LIST OF FILL AREA INDICES function, *Part 2*, 11–96

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS function, *Part 2*, 11–97

INQUIRE LIST OF PATTERN INDICES function, *Part 2*, 11–98

INQUIRE LIST OF POLYLINE INDICES function, *Part 2*, 11–99

INQUIRE LIST OF POLYMARKER INDICES function, *Part 2*, 11–100

INQUIRE LIST OF TEXT INDICES function, *Part 2*, 11–101

INQUIRE LIST OF VIEW INDICES function, *Part 2*, 11–102

INQUIRE LOCATOR DEVICE STATE 3 function, *Part 2*, 11–108

INQUIRE LOCATOR DEVICE STATE function, *Part 2*, 11–103  
example, *Part 1*, 9–94

INQUIRE MARKER SIZE SCALE FACTOR function, *Part 2*, 11–113

INQUIRE MARKER TYPE function, *Part 2*, 11–114

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3 function, *Part 2*, 11–116

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES function, *Part 2*, 11–115

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER function, *Part 2*, 11–117

INQUIRE MORE SIMULTANEOUS EVENTS function, *Part 2*, 11–118

INQUIRE NAME OF OPEN SEGMENT function, *Part 2*, 11–119

INQUIRE NORMALIZATION TRANSFORMATION 3 function, *Part 2*, 11–121

INQUIRE NORMALIZATION TRANSFORMATION function, *Part 2*, 11–120

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES function, *Part 2*, 11–122

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED function, *Part 2*, 11–123

INQUIRE OPERATING STATE VALUE function, *Part 2*, 11–124

INQUIRE PATTERN FACILITIES function, *Part 2*, 11–125

INQUIRE PATTERN HEIGHT VECTOR function, *Part 2*, 11–126

INQUIRE PATTERN REFERENCE POINT AND VECTORS function, *Part 2*, 11–128

INQUIRE PATTERN REFERENCE POINT function, *Part 2*, 11–127

INQUIRE PATTERN REPRESENTATION function, *Part 2*, 11–129

INQUIRE PATTERN WIDTH VECTOR function, *Part 2*, 11–130

INQUIRE PICK DEVICE STATE 3 function, *Part 2*, 11–133

INQUIRE PICK DEVICE STATE function, *Part 2*, 11–131  
example, *Part 1*, 9–98

INQUIRE PIXEL ARRAY DIMENSIONS function, *Part 2*, 11–138

INQUIRE PIXEL ARRAY function, *Part 2*, 11–136

INQUIRE PIXEL function, *Part 2*, 11–135

INQUIRE POLYLINE COLOUR INDEX function, *Part 2*, 11–139

INQUIRE POLYLINE FACILITIES function, *Part 2*, 11–140

INQUIRE POLYLINE INDEX function, *Part 2*, 11–141

INQUIRE POLYLINE REPRESENTATION function, *Part 2*, 11–142

INQUIRE POLYMARKER COLOUR INDEX function, *Part 2*, 11–143

INQUIRE POLYMARKER FACILITIES function, *Part 2*, 11–144

INQUIRE POLYMARKER INDEX function, *Part 2*, 11–145

INQUIRE POLYMARKER REPRESENTATION function, *Part 2*, 11–146

INQUIRE PREDEFINED COLOUR REPRESENTATION function, *Part 2*, 11–147

INQUIRE PREDEFINED EDGE REPRESENTATION function, *Part 2*, 11–148

INQUIRE PREDEFINED FILL AREA REPRESENTATION function, *Part 2*, 11–149

INQUIRE PREDEFINED PATTERN REPRESENTATION function, *Part 2*, 11–150

INQUIRE PREDEFINED POLYLINE REPRESENTATION function, *Part 2*, 11–151

INQUIRE PREDEFINED POLYMARKER REPRESENTATION function, *Part 2*, 11–152

INQUIRE PREDEFINED TEXT REPRESENTATION function, *Part 2*, 11–153

INQUIRE PREDEFINED VIEW REPRESENTATION function, *Part 2*, 11–154

INQUIRE SEGMENT ATTRIBUTES 3 function, *Part 2*, 11-157

INQUIRE SEGMENT ATTRIBUTES function, *Part 2*, 11-155

INQUIRE SET OF ACTIVE WORKSTATIONS function, *Part 2*, 11-159

INQUIRE SET OF ASSOCIATED WORKSTATIONS function, *Part 2*, 11-160

INQUIRE SET OF OPEN WORKSTATIONS function, *Part 2*, 11-161

INQUIRE SET OF SEGMENT NAMES IN USE function, *Part 2*, 11-162

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION function, *Part 2*, 11-163

INQUIRE STRING DEVICE STATE 3 function, *Part 2*, 11-166

INQUIRE STRING DEVICE STATE function, *Part 2*, 11-164  
example, *Part 1*, 9-104

INQUIRE STROKE DEVICE STATE 3 function, *Part 2*, 11-171

INQUIRE STROKE DEVICE STATE function, *Part 2*, 11-168

INQUIRE TEXT ALIGNMENT function, *Part 2*, 11-175

INQUIRE TEXT COLOUR INDEX function, *Part 2*, 11-176

INQUIRE TEXT EXTENT 3 function, *Part 2*, 11-178

INQUIRE TEXT EXTENT function, *Part 2*, 11-177

INQUIRE TEXT FACILITIES function, *Part 2*, 11-179

INQUIRE TEXT FONT AND PRECISION function, *Part 2*, 11-181

INQUIRE TEXT INDEX function, *Part 2*, 11-182

INQUIRE TEXT PATH function, *Part 2*, 11-183

INQUIRE TEXT REPRESENTATION function, *Part 2*, 11-184

INQUIRE VALUATOR DEVICE STATE 3 function, *Part 2*, 11-188

INQUIRE VALUATOR DEVICE STATE function, *Part 2*, 11-186  
example, *Part 1*, 9-107

INQUIRE VIEW FACILITIES function, *Part 2*, 11-190

INQUIRE VIEW REPRESENTATION 3 function, *Part 2*, 11-191

INQUIRE WORKSTATION CATEGORY function, *Part 2*, 11-193

INQUIRE WORKSTATION CLASSIFICATION function, *Part 2*, 11-194

INQUIRE WORKSTATION CONNECTION AND TYPE function, *Part 2*, 11-195  
example, *Part 1*, 4-26

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES function, *Part 2*, 11-196

INQUIRE WORKSTATION MAXIMUM NUMBERS function, *Part 2*, 11-198

INQUIRE WORKSTATION STATE function, *Part 2*, 11-199

INQUIRE WORKSTATION TRANSFORMATION 3 function, *Part 2*, 11-202

INQUIRE WORKSTATION TRANSFORMATION function, *Part 2*, 11-200

Inquiry functions, *Part 2*, 11-5 to 11-203  
input use, *Part 1*, 9-19  
introduction to, *Part 2*, 11-1 to 11-4

Inserting segments, *Part 1*, 8-3

INSERT SEGMENT 3 function, *Part 1*, 8-19

INSERT SEGMENT function, *Part 1*, 8-17  
example, *Part 1*, 8-32

Integer lists  
allocation for, *Part 2*, 11-4

Interface  
prompt and echo types, *Part 1*, 9-5 to 9-13

Interior styles  
See also Attributes  
See also Hatches  
See also Patterns  
of fill areas, *Part 1*, 6-28

Interpret  
metafiles, *Part 1*, 10-1

INTERPRET ITEM function, *Part 1*, 10-7

Items  
metafile header, *Part 1*, 10-2

## K

---

Kernel  
GKS, *Part 1*, 4-1

## L

---

Lengths  
See also Data records  
See also Input  
input data record, *Part 1*, 9-8  
metafile data record, *Part 1*, 10-4

Levels  
of GKS, *Part 1*, 1-4

Lines  
See also Attributes  
See also Output  
generating, *Part 1*, 5-16, 5-17  
types, *Part 1*, 1-3, 6-34  
width, *Part 1*, 6-35

Linking, *Part 1*, 2-2, 3-1  
reducing time, *Part 1*, 3-7  
RISC processors, *Part 1*, 3-2

Lists  
See also GKS  
See also Input

## Lists (cont'd)

- See also Workstations
- GKS state, *Part 2*, 11-1
- segment state, *Part 2*, 11-1
- viewport input priority, *Part 1*, 7-6, 9-18
- workstation state, *Part 2*, 11-1

## Locator

- input class, *Part 1*, 9-4
- viewport input priority, *Part 1*, 7-6, 9-18

## Logging

- errors, *Part 2*, 12-5

## Logical device number

- See Device number

## Logical input devices, *Part 1*, 9-1 to 9-3

- See also Input
- activating, *Part 1*, 9-2, 9-3
- classes, *Part 1*, 9-1
- controlling the appearance of, *Part 1*, 9-2
- deactivating, *Part 1*, 9-2, 9-3
- device number, *Part 1*, 9-1
- initializing, *Part 1*, 9-3
- triggering, *Part 1*, 9-3
- workstation identifier, *Part 1*, 9-1

## Logical names, *Part 1*, 2-3

- defining
  - at DCL level, *Part 1*, 2-3
- general, *Part 1*, 2-3
- GKS\$ASF, *Part 1*, 2-3
- GKS\$CONID, *Part 1*, 2-3
- GKS\$DEF\_MODE, *Part 1*, 2-3
- GKS\$ERRFILE, *Part 1*, 2-4
- GKS\$ERROR, *Part 1*, 2-4
- GKS\$IRG, *Part 1*, 2-4
- GKS\$METAFILE\_TYPE, *Part 1*, 2-4
- GKS\$NDC\_CLIP, *Part 1*, 2-4
- GKS\$STROKE\_FONT1, *Part 1*, 2-4
- GKS\$WSTYPE, *Part 1*, 2-4
- search order, *Part 1*, 2-3
- types, *Part 1*, 2-3

## VMS

- default, *Part 1*, 2-2
- GKS\$CONID, *Part 1*, 2-2
- GKS\$ERRFILE, *Part 1*, 2-4
- GKS\$WSTYPE, *Part 1*, 2-2

# M

---

## Mapping

- See also Transformations
- cell array color indexes, *Part 1*, 5-6
- color indexes, *Part 1*, 5-4
- device transformations, *Part 1*, 7-7

## Marker, *Part 1*, 5-18, 5-19

## Markers

- See also Attributes
- See also Output
- size, *Part 1*, 6-36

## Markers (cont'd)

- types, *Part 1*, 6-37

## Matrix

- See also Rotation
- See also Scale
- See also Translation
- segment transformation, *Part 1*, 8-8

## Matrixes

- view mapping, *Part 1*, 7-7
- view orientation, *Part 1*, 7-7

## Measure

- See also Logical input devices
- cycling input device control, *Part 1*, 9-14

## Menus

- See also Choice
- input, *Part 1*, 9-4

## MESSAGE function, *Part 1*, 4-17

- example, *Part 1*, 9-94

## Messages

- See also Errors
- error, *Part 2*, A-1 to A-37
- sent to workstations, *Part 1*, 4-17

## Metafile functions, *Part 1*, 1-4, 10-5 to 10-9

- introduction to, *Part 1*, 10-1 to 10-5

## Metafiles, *Part 1*, 10-1

- creating, *Part 1*, 10-1
- creating CGM metafiles, *Part 1*, 10-3
- current item, *Part 1*, 10-4
- item header, *Part 1*, 10-2
- list of errors, *Part 2*, A-14 to A-15
- reading, *Part 1*, 10-4 to 10-5
- reproducing pictures, *Part 1*, 10-1
- reserved data numbers, *Part 1*, 10-5
- structure, *Part 1*, 10-2
- user-defined data, *Part 1*, 10-5
- workstation categories, *Part 1*, 4-2

## Mirror images

- cell arrays, *Part 1*, 5-4
- 3D cell arrays, *Part 1*, 5-6

## Mode

- See also Input
- control
  - SET CHOICE MODE function, *Part 1*, 9-88
  - SET LOCATOR MODE function, *Part 1*, 9-89
  - SET PICK MODE function, *Part 1*, 9-90
  - SET STRING MODE function, *Part 1*, 9-91
  - SET STROKE MODE function, *Part 1*, 9-92
  - SET VALUATOR MODE function, *Part 1*, 9-93
- event, *Part 1*, 9-2, 9-3, 9-16
  - AWAIT EVENT function, *Part 1*, 9-22
  - FLUSH DEVICE EVENTS function, *Part 1*, 9-24
  - GET CHOICE function, *Part 1*, 9-25

## Mode

- event (cont'd)
  - GET LOCATOR 3 function, *Part 1*, 9-27
  - GET LOCATOR function, *Part 1*, 9-26
  - GET PICK function, *Part 1*, 9-28
  - GET STRING function, *Part 1*, 9-29
  - GET STROKE 3 function, *Part 1*, 9-32
  - GET STROKE function, *Part 1*, 9-30
  - GET VALUATOR function, *Part 1*, 9-34
- input operating, *Part 1*, 9-2, 9-3, 9-14
- request, *Part 1*, 9-2, 9-3, 9-14
  - REQUEST CHOICE function, *Part 1*, 9-65
  - REQUEST LOCATOR 3 function, *Part 1*, 9-68
  - REQUEST LOCATOR function, *Part 1*, 9-66
  - REQUEST PICK function, *Part 1*, 9-70
  - REQUEST STRING function, *Part 1*, 9-71
  - REQUEST STROKE 3 function, *Part 1*, 9-75
  - REQUEST STROKE function, *Part 1*, 9-73
  - REQUEST VALUATOR function, *Part 1*, 9-77
- sample, *Part 1*, 9-2, 9-3, 9-15
  - SAMPLE CHOICE function, *Part 1*, 9-78
  - SAMPLE LOCATOR 3 function, *Part 1*, 9-80
  - SAMPLE LOCATOR function, *Part 1*, 9-79
  - SAMPLE PICK function, *Part 1*, 9-81
  - SAMPLE STRING function, *Part 1*, 9-82
  - SAMPLE STROKE 3 function, *Part 1*, 9-85
  - SAMPLE STROKE function, *Part 1*, 9-83
  - SAMPLE VALUATOR function, *Part 1*, 9-87

## Models

- color, *Part 1*, 6-16

## Multiple transformations

- See also Segments
- See also Transformations

## N

---

### Names

- segment, *Part 1*, 8-1

### NDC

- See also Transformations
- See Normalized device coordinates
- fixed points, *Part 1*, 8-7

*New frame necessary at update* entry, *Part 1*, 8-4

Nominal sizes, *Part 1*, 6-1

Nongeometric attributes, *Part 1*, 6-1

- See also Attributes

### Normalization

- clipping, *Part 1*, 7-4
- overlapping viewports, *Part 1*, 7-6
- transformations, *Part 1*, 1-3

### Normalization

- transformations (cont'd)
  - maximum number, *Part 1*, 7-5
  - viewports, *Part 1*, 7-4
  - windows, *Part 1*, 7-3

### Normalization transformations

- See also Transformations
- See Transformations

Normalized device coordinates, *Part 1*, 7-1

Normalized projection coordinates, *Part 1*, 7-1, 7-7

### NPC

- See Normalized projection coordinates

### Numbers

- See also Errors
- See also Input
- error, *Part 2*, A-1

## O

---

### OFF

- error state, *Part 2*, 12-1

### ON

- error state, *Part 2*, 12-1

### One-to-one

- See also Mapping

OPEN GKS function, *Part 1*, 4-18

- example, *Part 1*, 4-24

### Opening

- GKS, *Part 1*, 4-4
- GKSM metafile workstations, *Part 1*, 10-1
- segments, *Part 1*, 4-5, 8-1
- workstations, *Part 1*, 4-4

Opening a workstation, *Part 1*, 2-2, 3-2 to 3-3

OPEN WORKSTATION function, *Part 1*, 4-19

- example, *Part 1*, 4-24

### Operating modes

- input, *Part 1*, 9-2, 9-3, 9-14 to 9-18

Operating states, *Part 1*, 4-3

- list of errors, *Part 2*, A-1 to A-2
- using output, *Part 1*, 5-1

### Operating system

- ULTRIX, *Part 1*, 3-1

### Order

- See also Transformations
- viewport input priority, *Part 1*, 7-6

### Origin

- See also Transformations
- world coordinate system, *Part 1*, 7-3

### Output

- See also Attributes
- altering the primitive, *Part 1*, 5-2
- attribute functions
  - See Attribute functions, *Part 1*, 6-1
- attributes, *Part 1*, 1-3, 5-2
- bound attributes, *Part 1*, 6-1
- default windows and viewports, *Part 1*, 5-2



Output (cont'd)

- deferral, *Part 1*, 4–6, 5–3
  - DECwindows, *Part 1*, 1–6
- list of errors, *Part 2*, A–10 to A–11
- list of primitives, *Part 1*, 1–2
- lost during transformations, *Part 1*, 7–8
- metafiles, *Part 1*, 10–1, 10–2
- pick identification, *Part 1*, 8–2
- pictures, *Part 1*, 7–1
- segments, *Part 1*, 8–1
- valid operating states, *Part 1*, 5–1
- workstation categories, *Part 1*, 4–2, 5–1

Output attributes

- See Attributes

Output functions, *Part 1*, 5–1 to 5–27

- introduction to, *Part 1*, 5–1 to 5–3

Overflow

- event input queue, *Part 1*, 9–17

Overlapping

- See also Transformations
- segments, *Part 1*, 8–6
- viewports, *Part 1*, 7–6, 9–18

## P

---

Parallel projection, *Part 1*, 7–7

Pasteboard

- See also Transformations
- normalization viewport, *Part 1*, 7–5

Path

- See also Text
- text, *Part 1*, 6–57

Patterns, *Part 1*, 6–28

- See also Attributes
- fill areas, *Part 1*, 5–8, 5–9
- reference points, *Part 1*, 6–38
- reference point vector, *Part 1*, 6–39
- representation, *Part 1*, 6–40
- specifying size, *Part 1*, 6–41
- style index values, *Part 1*, 6–31

Pending

- See also Implicit regenerations
- bundle changes, *Part 1*, 4–7
- output generation, *Part 1*, 4–6
- segment attribute changes, *Part 1*, 4–7
- workstation transformations, *Part 1*, 4–7

Perspective projection, *Part 1*, 7–7

Physical input devices, *Part 1*, 9–1

pi, *Part 1*, 8–8

Pick

- See also Input
- See also Segments
- identifier, *Part 1*, 8–2, 9–4
- input class, *Part 1*, 9–4
- segment detectability, *Part 1*, 8–5
- specifying NOPICK input, *Part 1*, 9–15, 9–16
- visibility, *Part 1*, 8–10

Pictures

- See also Output
- See also Transformations
- composition, *Part 1*, 1–3, 7–1
- reproducing
  - metafiles, *Part 1*, 10–1

Pipeline

- See also Segments

Plotting

- See also Transformations
- pictures, *Part 1*, 7–1

Pointers

- See also Bundles
- into bundle tables, *Part 1*, 6–3

Points

- See also Transformations
- coordinate, *Part 1*, 7–1
- pattern reference, *Part 1*, 6–38, 6–39
- segments
  - fixed points, *Part 1*, 8–7
- viewport input priority, *Part 1*, 7–6

Polygons

- See also Attributes
- See also Output
- fill areas, *Part 1*, 5–8, 5–9

Polyline

- See also Attributes
- See also Output
- attributes
  - SET LINETYPE, *Part 1*, 6–34
  - SET LINEWIDTH SCALE FACTOR, *Part 1*, 6–35
  - SET POLYLINE COLOUR INDEX, *Part 1*, 6–43, 6–44
- bundles, *Part 1*, 6–44
- line type, *Part 1*, 1–3
- representation, *Part 1*, 6–45
- type, *Part 1*, 6–34

POLYLINE 3 function, *Part 1*, 5–17

POLYLINE function, *Part 1*, 5–16

- example, *Part 1*, 6–65

Polylines

- initial attributes, *Part 2*, E–1

Polymarker

- See also Output
- See also Transformations
- attributes
  - SET MARKER SIZE SCALE FACTOR, *Part 1*, 6–36
  - SET MARKER TYPE, *Part 1*, 6–37
  - SET POLYMARKER COLOUR INDEX, *Part 1*, 6–47
  - SET POLYMARKER INDEX, *Part 1*, 6–48
- bundle table, *Part 1*, 6–48
- representation, *Part 1*, 6–49

POLYMARKER 3 function, *Part 1*, 5–19

POLYMARKER function, *Part 1*, 5–18

- example, *Part 1*, 6–68

Polymarkers

- initial attributes, *Part 2*, E–2

Positioning

- primitives, *Part 1*, 7–5

Precision

- text
  - establishing, *Part 1*, 6–54

Presentation

- See also Transformations
- pictures, *Part 1*, 7–7

Primitives

- See also Attributes
- See also Output
- attributes, *Part 1*, 6–1
- bound attributes, *Part 1*, 6–1
- clipping segments, *Part 1*, 8–9
- highlighting, *Part 1*, 8–6
- input prompt and echo types, *Part 1*, 9–5
- list, *Part 1*, 1–2
- lost during regeneration, *Part 1*, 4–7
- lost during transformations, *Part 1*, 7–8
- output, *Part 1*, 5–1 to 5–3
- pick identification, *Part 1*, 8–2
- reproducing
  - metafiles, *Part 1*, 10–1
  - segment detectability, *Part 1*, 8–5
  - segments, *Part 1*, 8–1
  - transformation, *Part 1*, 7–3

Priority

- See also Input
- segments, *Part 1*, 8–6
- viewport input, *Part 1*, 7–6, 9–18

Programming

- See also GKS
- device-independent input, *Part 1*, 9–20
- error handling, *Part 2*, 12–1

Programs

- execution of, *Part 1*, 2–2, 3–1
- pausing, *Part 1*, 1–6

Projections

- parallel, *Part 1*, 7–7
- perspective, *Part 1*, 7–7

Prompt and echo types, *Part 1*, 9–2, 9–5 to 9–13

- See also Input
- standard data records, *Part 1*, 9–8

Proportionate

- See also Transformations

## Q

---

Queue

- event input, *Part 1*, 9–16

## R

---

Radians

- translating to degrees, *Part 1*, 8–8

Ranges

- See also Transformations
- windows and viewports, *Part 1*, 7–3

Ratio

- See also Transformations

Reading a metafile, *Part 1*, 10–4

READ ITEM FROM GKSM function, *Part 1*, 10–8

READ ITEM FROM METAFILE

- See READ ITEM FROM GKSM function

Realized values, *Part 2*, 11–4

Real numbers

- input, *Part 1*, 9–4

Records

- See also Escapes
- See also GDPs
- See also Input
- input, *Part 1*, 9–8
  - prompt and echo types, *Part 1*, 9–5 to 9–13
  - standard, *Part 1*, 9–8

Rectangles

- See also Attributes
- See also Transformations
- clipping, *Part 1*, 7–4
  - segments, *Part 1*, 8–9

REDRAW ALL SEGMENTS ON WORKSTATION

- function, *Part 1*, 4–20
- example, *Part 1*, 8–28

Regenerations

- segments, *Part 1*, 8–3
- workstation surface, *Part 1*, 4–7
- workstation transformations, *Part 1*, 7–8

Releasing

- DEC GKS buffers, *Part 1*, 4–11

RENAME SEGMENT function, *Part 1*, 8–21

Renaming

- segments, *Part 1*, 8–1

Reports

- current event on input queue, *Part 1*, 9–16

Representation

- See also Attributes
- bundle table entries, *Part 1*, 6–3
- color, *Part 1*, 6–17
- edge, *Part 1*, 6–22
- fill area, *Part 1*, 6–29
- functions, *Part 1*, 6–4
- implicit regenerations, *Part 1*, 6–4
- pattern, *Part 1*, 6–40

## Representation (cont'd)

- polyline, *Part 1*, 6–45
- polymarker, *Part 1*, 6–49
- text, *Part 1*, 6–58

## Reproducing

- metafiles, *Part 1*, 10–1

- REQUEST CHOICE function, *Part 1*, 9–65
- REQUEST functions, *Part 1*, 9–2, 9–3, 9–15
- REQUEST LOCATOR 3 function, *Part 1*, 9–68
- REQUEST LOCATOR function, *Part 1*, 9–66
- Request mode, *Part 1*, 9–14 to 9–15

### See also Input

- breaking, *Part 1*, 9–15

- REQUEST PICK function, *Part 1*, 9–70
- REQUEST STRING function, *Part 1*, 9–71
  - example, *Part 1*, 9–104
- REQUEST STROKE 3 function, *Part 1*, 9–75
- REQUEST STROKE function, *Part 1*, 9–73
- REQUEST VALUATOR function, *Part 1*, 9–77

## Reverse video

- highlighting segments, *Part 1*, 8–6

## Rotation

- fixed points, *Part 1*, 8–7
- segments, *Part 1*, 8–7

- RUN DCL command, *Part 1*, 2–2

# S

---

- SAMPLE CHOICE function, *Part 1*, 9–78
- SAMPLE functions, *Part 1*, 9–15
- SAMPLE LOCATOR 3 function, *Part 1*, 9–80
- SAMPLE LOCATOR function, *Part 1*, 9–79
- Sample mode, *Part 1*, 9–15 to 9–16
- SAMPLE PICK function, *Part 1*, 9–81
  - example, *Part 1*, 9–98
- SAMPLE STRING function, *Part 1*, 9–82
- SAMPLE STROKE 3 function, *Part 1*, 9–85
- SAMPLE STROKE function, *Part 1*, 9–83
- SAMPLE VALUATOR function, *Part 1*, 9–87
  - example, *Part 1*, 9–107

## Scale

### See also Segments

- edge width factor, *Part 1*, 6–25
- fixed points, *Part 1*, 8–7
- segments, *Part 1*, 8–7
- valuator input, *Part 1*, 9–4

- Scale factors, *Part 1*, 6–1

## Scratch pad

### See also Transformations

- normalization window, *Part 1*, 7–5

- Segment functions, *Part 1*, 8–1 to 8–41
  - introduction to, *Part 1*, 8–1 to 8–10

## Segments

- accumulated transformations, *Part 1*, 8–9
- associating, *Part 1*, 8–3
- attributes, *Part 1*, 8–5
  - SET DETECTABILITY function, *Part 1*, 8–22

## Segments

### attributes (cont'd)

- SET HIGHLIGHTING function, *Part 1*, 8–23
- SET SEGMENT PRIORITY function, *Part 1*, 8–24
- SET VISIBILITY function, *Part 1*, 8–27
- clipping, *Part 1*, 8–9
- closing, *Part 1*, 4–5
- copying, *Part 1*, 8–3
- creating, *Part 1*, 4–5, 8–1
- deleting, *Part 1*, 8–2
- detectability, *Part 1*, 8–5
- highlighting, *Part 1*, 8–6
- initial attributes, *Part 2*, E–4
- input, *Part 1*, 8–2
- inserting, *Part 1*, 8–3
- list of errors, *Part 2*, A–11 to A–12
- metafiles, *Part 1*, 10–2
- names, *Part 1*, 8–1
- opening, *Part 1*, 4–5, 8–1
- order of transformation, *Part 1*, 8–9
- overlapping, *Part 1*, 8–6
- priority, *Part 1*, 8–6
- redrawn, *Part 1*, 4–7
- renaming, *Part 1*, 8–1
- rotating, *Part 1*, 8–7
- scaling, *Part 1*, 8–7
- selecting a transformation, *Part 1*, 8–8
- state list, *Part 1*, 8–1
- storage, *Part 1*, 8–2
- surface update, *Part 1*, 8–3
- transformation, *Part 1*, 8–7 to 8–9
  - ACCUMULATE TRANSFORMATION MATRIX 3 function, *Part 1*, 7–12
  - ACCUMULATE TRANSFORMATION MATRIX function, *Part 1*, 7–10
  - EVALUATE TRANSFORMATION MATRIX 3 function, *Part 1*, 7–16
  - EVALUATE TRANSFORMATION MATRIX function, *Part 1*, 7–14
  - EVALUATE VIEW MAPPING MATRIX 3 function, *Part 1*, 7–18
  - EVALUATE VIEW ORIENTATION MATRIX 3 function, *Part 1*, 7–21
  - SET SEGMENT TRANSFORMATION 3 function, *Part 1*, 8–26
  - SET SEGMENT TRANSFORMATION function, *Part 1*, 8–25
- transformation matrix, *Part 1*, 8–8
- translating, *Part 1*, 8–7
- using
  - ASSOCIATE SEGMENT WITH WORKSTATION function, *Part 1*, 8–11
  - CLOSE SEGMENT function, *Part 1*, 8–12
  - COPY SEGMENT TO WORKSTATION, *Part 1*, 8–13

## Segments

- using (cont'd)
  - CREATE SEGMENT function, *Part 1*, 8–14
  - DELETE SEGMENT FROM WORKSTATION function, *Part 1*, 8–16
  - DELETE SEGMENT function, *Part 1*, 8–15
  - INSERT SEGMENT function, *Part 1*, 8–17, 8–19
  - RENAME SEGMENT function, *Part 1*, 8–21
- visibility, *Part 1*, 8–10
- WDSS, *Part 1*, 8–2
- WISS, *Part 1*, 8–3
- SELECT NORMALIZATION TRANSFORMATION function, *Part 1*, 7–23
  - example, *Part 1*, 7–46
- SET ASPECT SOURCE FLAGS 3 function, *Part 1*, 6–9
- SET ASPECT SOURCE FLAGS function, *Part 1*, 6–7
  - example, *Part 1*, 6–62
- SET CHARACTER EXPANSION FACTOR function, *Part 1*, 6–11
- SET CHARACTER HEIGHT function, *Part 1*, 6–12
  - example, *Part 1*, 6–68
- SET CHARACTER SPACING function, *Part 1*, 6–13
- SET CHARACTER UP VECTOR function, *Part 1*, 6–14
- SET CHOICE MODE function, *Part 1*, 9–88
- SET CLIPPING INDICATOR function, *Part 1*, 7–24
  - example, *Part 1*, 7–46
- SET COLOUR MODEL function, *Part 1*, 6–16
- SET COLOUR REPRESENTATION function, *Part 1*, 6–17
  - example, *Part 1*, 6–60
- SET DEFERRAL STATE function, *Part 1*, 4–21
  - example, *Part 1*, 5–24
- SET DETECTABILITY function, *Part 1*, 8–22
  - example, *Part 1*, 9–98
- SET EDGE COLOUR INDEX function, *Part 1*, 6–19
- SET EDGE FLAG function, *Part 1*, 6–20
- SET EDGE INDEX function, *Part 1*, 6–21
- SET EDGE REPRESENTATION function, *Part 1*, 6–22
- SET EDGETYPE function, *Part 1*, 6–24
- SET EDGEWIDTH SCALE FACTOR function, *Part 1*, 6–25
- SET FILL AREA COLOUR INDEX function, *Part 1*, 6–26
  - example, *Part 1*, 6–60
- SET FILL AREA INDEX function, *Part 1*, 6–27
  - example, *Part 1*, 6–62
- SET FILL AREA INTERIOR STYLE function, *Part 1*, 6–28
  - example, *Part 1*, 6–60
- SET FILL AREA REPRESENTATION function, *Part 1*, 6–29
  - example, *Part 1*, 6–62
- SET FILL AREA STYLE INDEX function, *Part 1*, 6–31
- SET HIGHLIGHTING function, *Part 1*, 8–23
  - example, *Part 1*, 8–38
- SET HLHSR IDENTIFIER function, *Part 1*, 6–32
- SET HLHSR MODE function, *Part 1*, 6–33
- SET LINE COLOUR INDEX
  - See SET POLYLINE COLOUR INDEX function
- SET LINE INDEX
  - See SET POLYLINE INDEX function
- SET LINE REPRESENTATION
  - See SET POLYLINE REPRESENTATION function
- SET LINETYPE function, *Part 1*, 6–34
  - example, *Part 1*, 6–65
- SET LINEWIDTH SCALE FACTOR function, *Part 1*, 6–35
- SET LOCATOR MODE function, *Part 1*, 9–89
  - example, *Part 1*, 9–94
- SET MARKER COLOUR INDEX
  - See SET POLYMARKER COLOUR INDEX function
- SET MARKER INDEX
  - See SET POLYMARKER INDEX function
- SET MARKER REPRESENTATION
  - See SET POLYMARKER REPRESENTATION function
- SET MARKER SIZE SCALE FACTOR function, *Part 1*, 6–36
- SET MARKER TYPE function, *Part 1*, 6–37
  - example, *Part 1*, 6–68
- SET MODE functions, *Part 1*, 9–2, 9–3, 9–14, 9–15
- SET PATTERN REFERENCE POINT AND VECTORS function, *Part 1*, 6–39
- SET PATTERN REFERENCE POINT function, *Part 1*, 6–38
- SET PATTERN REPRESENTATION function, *Part 1*, 6–40
- SET PATTERN SIZE function, *Part 1*, 6–41
- SET PICK IDENTIFIER function, *Part 1*, 6–42
  - example, *Part 1*, 9–98
- SET PICK MODE function, *Part 1*, 9–90
  - example, *Part 1*, 9–98
- SET POLYLINE COLOUR INDEX function, *Part 1*, 6–43
- SET POLYLINE INDEX function, *Part 1*, 6–44

SET POLYLINE REPRESENTATION function,  
     *Part 1, 6–45*  
 SET POLYLINE TYPE  
     See SET LINETYPE function  
 SET POLYLINE WIDTH SCALE FACTOR  
     See SET LINEWIDTH SCALE FACTOR  
     function  
 SET POLYMARKER COLOUR INDEX function,  
     *Part 1, 6–47*  
     example, *Part 1, 6–68*  
 SET POLYMARKER INDEX function, *Part 1,*  
     6–48  
 SET POLYMARKER REPRESENTATION  
     function, *Part 1, 6–49*  
 SET POLYMARKER SIZE SCALE FACTOR  
     See SET MARKER SIZE SCALE FACTOR  
     function  
 SET POLYMARKER TYPE  
     See SET MARKER TYPE function  
 SET SEGMENT PRIORITY function, *Part 1, 8–24*  
 SET SEGMENT TRANSFORMATION 3 function,  
     *Part 1, 8–26*  
 SET SEGMENT TRANSFORMATION function,  
     *Part 1, 8–25*  
     example, *Part 1, 7–37*  
 SET STRING MODE function, *Part 1, 9–91*  
 SET STROKE MODE function, *Part 1, 9–92*  
 SET TEXT ALIGNMENT function, *Part 1, 6–51*  
     example, *Part 1, 6–68*  
 SET TEXT COLOUR INDEX function, *Part 1,*  
     6–53  
 SET TEXT EXPANSION FACTOR  
     See SET CHARACTER EXPANSION FACTOR  
     function  
 SET TEXT FONT AND PRECISION function,  
     *Part 1, 6–54*  
 SET TEXT HEIGHT  
     See SET CHARACTER HEIGHT function  
 SET TEXT INDEX function, *Part 1, 6–56*  
 SET TEXT PATH function, *Part 1, 6–57*  
     example, *Part 1, 6–68*  
 SET TEXT REPRESENTATION function, *Part 1,*  
     6–58  
 SET TEXT SPACING  
     See SET CHARACTER SPACING function  
 SET TEXT UP VECTOR  
     See SET CHARACTER UP VECTOR function  
 Settings  
     See also Attributes  
     See also Transformations  
     attribute values, *Part 1, 6–1*  
     pattern sizes, *Part 1, 6–41*  
     segment transformations, *Part 1, 8–8*  
 SET VALUATOR MODE function, *Part 1, 9–93*  
     example, *Part 1, 9–107*  
  
 Set values, *Part 2, 11–4*  
 SET VIEW INDEX function, *Part 1, 7–25*  
 SET VIEWPORT 3 function, *Part 1, 7–29*  
 SET VIEWPORT function, *Part 1, 7–28*  
     example, *Part 1, 7–46*  
 SET VIEWPORT INPUT PRIORITY function,  
     *Part 1, 7–30*  
 SET VIEW REPRESENTATION 3 function, *Part*  
     1, 7–19, 7–26  
 SET VIEW TRANSFORMATION INPUT  
     PRIORITY function, *Part 1, 7–27*  
 SET VISIBILITY function, *Part 1, 8–27*  
 SET WINDOW 3 function, *Part 1, 7–32*  
 SET WINDOW function, *Part 1, 7–31*  
     example, *Part 1, 7–46*  
 SET WORKSTATION VIEWPORT 3 function,  
     *Part 1, 7–34*  
 SET WORKSTATION VIEWPORT function, *Part*  
     1, 7–33  
     example, *Part 1, 7–50*  
 SET WORKSTATION WINDOW 3 function, *Part*  
     1, 7–36  
 SET WORKSTATION WINDOW function, *Part 1,*  
     7–35  
 Shift segments, *Part 1, 8–7*  
 Shrink segments, *Part 1, 8–7*  
 Sizes  
     input data record, *Part 1, 9–19*  
     markers, *Part 1, 6–36*  
     patterns, *Part 1, 6–41*  
     segments, *Part 1, 8–8*  
 Software fonts, *Part 1, 6–54*  
 Solid  
     See also Attributes  
     fill area interior style, *Part 1, 6–28*  
     fill areas, *Part 1, 5–8, 5–9*  
 Spacing  
     text, *Part 1, 6–13*  
 Standards  
     See also ANSI  
     See also GKS  
     metafiles, *Part 1, 10–1*  
 State lists  
     GKS, *Part 1, 4–3, 4–18, 8–1; Part 2, 11–1*  
         attributes, *Part 1, 6–1*  
         segment, *Part 1, 4–3, 8–1*  
         segments, *Part 2, 11–1*  
         surface control entries, *Part 1, 4–8*  
         workstation, *Part 1, 4–3, 4–19; Part 2, 11–1*  
             attributes, *Part 1, 6–3*  
 Statements  
     include, *Part 1, 2–1, 3–1*  
 States  
     error, *Part 2, 12–1*  
     operating, *Part 1, 4–3*  
 Status  
     inquiry error status argument, *Part 2, 11–3*

Storage  
 metafiles, *Part 1*, 1–4, 10–1  
 segments, *Part 1*, 8–2

Strings  
 See also Text  
 input class, *Part 1*, 9–4

Stroke  
 input class, *Part 1*, 9–4  
 viewport input priority, *Part 1*, 9–18  
 viewport priority, *Part 1*, 7–6

Structure  
 metafiles, *Part 1*, 10–2

Styles  
 See also Attributes  
 fill areas, *Part 1*, 6–31

Surface  
 See also Implicit regenerations  
 control, *Part 1*, 4–6  
 foreground and background colors, *Part 1*, 6–5  
 implicit regenerations  
   attribute changes, *Part 1*, 6–4  
 regeneration, *Part 1*, 4–7  
 state list entries, *Part 1*, 4–8  
 update  
   segments, *Part 1*, 8–3

Symbols  
 polymarkers, *Part 1*, 5–18, 5–19

Synchronous input, *Part 1*, 9–14  
 See also Input

Syntax  
 format, *Part 1*, 1–5

System defaults file, *Part 1*, 3–6

System errors  
 list of, *Part 2*, A–16 to A–19

## T

---

Tables  
 See also Attributes  
 See also Bundles  
 attribute bundle, *Part 1*, 6–3  
 color index, *Part 1*, 6–16, 6–17  
 edge bundle index, *Part 1*, 6–22  
 fill area bundle index, *Part 1*, 6–29  
 GKS description, *Part 2*, 11–1  
 pattern style bundle index, *Part 1*, 6–40  
 polyline bundle index, *Part 1*, 6–45  
 polymarker bundle index, *Part 1*, 6–49  
 text bundle index, *Part 1*, 6–58  
 workstation description, *Part 2*, 11–1

Terminate  
 workstation environment, *Part 1*, 4–12

Terminating  
 error handling, *Part 2*, 12–1  
 request input, *Part 1*, 9–14

Text  
 See also Attributes  
 alignment, *Part 1*, 6–51  
 attributes  
   SET CHARACTER EXPANSION FACTOR,  
     *Part 1*, 6–11  
   SET CHARACTER HEIGHT, *Part 1*, 6–12  
   SET CHARACTER SPACING, *Part 1*, 6–13  
   SET CHARACTER UP VECTOR, *Part 1*,  
     6–14  
   SET TEXT ALIGNMENT, *Part 1*, 6–51  
   SET TEXT COLOUR INDEX, *Part 1*, 6–53  
   SET TEXT FONT AND PRECISION, *Part*  
     1, 6–54  
   SET TEXT INDEX, *Part 1*, 6–56  
   SET TEXT PATH, *Part 1*, 6–57  
 bundles, *Part 1*, 6–56  
 character width, *Part 1*, 6–11  
 expansion factor, *Part 1*, 6–11  
 fonts, *Part 1*, 6–54  
 height, *Part 1*, 6–12  
 initial attributes, *Part 2*, E–2  
 input, *Part 1*, 9–4  
 path, *Part 1*, 6–57  
 precision, *Part 1*, 6–54  
 representation, *Part 1*, 6–58  
 spacing, *Part 1*, 6–13  
 up-vector, *Part 1*, 6–14  
 TEXT 3 function, *Part 1*, 5–22  
 TEXT function, *Part 1*, 5–20  
   example, *Part 1*, 6–68

Toggling  
 logical input device control, *Part 1*, 9–14

Transformation functions, *Part 1*, 7–1 to 7–54  
 introduction to, *Part 1*, 7–1 to 7–8

Transformations  
 device, *Part 1*, 7–7  
 identity, *Part 1*, 7–7  
 identity (segment), *Part 1*, 8–8  
 implicit regenerations, *Part 1*, 7–8  
 input change vectors, *Part 1*, 9–5  
 list of errors, *Part 2*, A–5 to A–6  
 metafiles, *Part 1*, 10–2  
 normalization, *Part 1*, 1–3, 7–3 to 7–6  
   clipping, *Part 1*, 7–4  
   initial attributes, *Part 2*, E–4  
   maximum number, *Part 1*, 7–5  
   overlapping viewports, *Part 1*, 7–6  
 SELECT NORMALIZATION  
   TRANSFORMATION function,  
     *Part 1*, 7–23  
 SET CLIPPING INDICATOR function,  
   *Part 1*, 7–24  
 SET VIEWPORT 3 function, *Part 1*, 7–29  
 SET VIEWPORT function, *Part 1*, 7–28  
 SET VIEWPORT INPUT PRIORITY  
   function, *Part 1*, 7–30  
 SET WINDOW 3 function, *Part 1*, 7–32

## Transformations

- normalization (cont'd)
  - SET WINDOW function, *Part 1*, 7–31
  - text height, *Part 1*, 6–12
- normalization viewports, *Part 1*, 7–4
- normalization windows, *Part 1*, 7–3
- overlapping viewports, *Part 1*, 9–18
- segments, *Part 1*, 8–7 to 8–9
  - accumulating, *Part 1*, 8–9
  - fixed points, *Part 1*, 8–7
  - matrix, *Part 1*, 8–8
- unity, *Part 1*, 7–4
- used for output, *Part 1*, 5–2
- view, *Part 1*, 7–7
- viewport input priority, *Part 1*, 9–18
- workstation, *Part 1*, 1–3
  - SET WORKSTATION VIEWPORT 3 function, *Part 1*, 7–34
  - SET WORKSTATION VIEWPORT function, *Part 1*, 7–33
  - SET WORKSTATION WINDOW 3 function, *Part 1*, 7–36
  - SET WORKSTATION WINDOW function, *Part 1*, 7–35

## Translations

- segments, *Part 1*, 8–7
- viewport input priority, *Part 1*, 7–6

## Transporting

- metafiles, *Part 1*, 10–1

## Transposing

- pictures, *Part 1*, 7–4

## Triggers

- input, *Part 1*, 9–3, 9–15

## Types

- edge, *Part 1*, 6–24
- inquiry value type argument, *Part 2*, 11–4
- line, *Part 1*, 6–34
- marker, *Part 1*, 6–37
- prompt and echo, *Part 1*, 9–5 to 9–13
- workstation
  - metafile, *Part 1*, 10–1
- workstations, *Part 1*, 4–2

## U

---

### ULTRIX linking

- RISC, *Part 1*, 3–2

### ULTRIX operating system, *Part 1*, 3–1 to 3–7

### Unity transformation, *Part 1*, 7–4

### Update

- See also Implicit regenerations
- attribute changes, *Part 1*, 6–4
- regenerating the surface, *Part 1*, 4–7
- releasing deferred output, *Part 1*, 4–6
- surface
  - segments, *Part 1*, 8–3
- the workstation surface, *Part 1*, 4–6

### UPDATE WORKSTATION function, *Part 1*, 4–23

- example, *Part 1*, 4–24

### Up-vector

- text, *Part 1*, 6–14

### User defaults file, *Part 1*, 3–6

### User defined

- error handler, *Part 2*, 12–1

## V

---

### Valuator

- input class, *Part 1*, 9–4

### Values

- attribute, *Part 1*, 6–1
- initial attribute, *Part 2*, E–1 to E–4
- maximum device coordinates, *Part 1*, 7–8
- of constants, *Part 2*, B–1 to B–39

### Vectors

- See also GDPs
- See also Segments
- pattern reference point, *Part 1*, 6–39
- text up-vector, *Part 1*, 6–14
- translation point, *Part 1*, 8–7

### View

- reference plane, *Part 1*, 7–21
- transformations, *Part 1*, 7–7
- volume, *Part 1*, 7–19

### View mapping matrix, *Part 1*, 7–7

### View orientation matrix, *Part 1*, 7–7

### View plane, *Part 1*, 7–7

### Viewports, *Part 1*, 7–28, 7–29, 7–30

- See also Transformations

- input priority, *Part 1*, 7–6, 9–18

- normalization, *Part 1*, 7–4

- initial value, *Part 2*, E–4

- overlapping, *Part 1*, 7–6, 9–18

- workstation, *Part 1*, 7–7

### View reference coordinates, *Part 1*, 7–1

- VRC, *Part 1*, 7–7

### View table, *Part 1*, 7–7

### Visibility segments, *Part 1*, 8–10

### Visual interface

- See also Input

- input prompt and echo types, *Part 1*, 9–5 to 9–13

### VMS logical names

- GKS\$CONID, *Part 1*, 2–2

- GKS\$ERRFILE, *Part 1*, 2–4

- GKS\$WSTYPE, *Part 1*, 2–2

### VRC

- See View reference coordinates

## W

WDSS, *Part 1*, 8–2

See also Segments

Width

See also Attributes

See also Transformations

character, *Part 1*, 6–11

line, *Part 1*, 6–35

Windows, *Part 1*, 7–31, 7–32

See also Transformations

normalization

initial value, *Part 2*, E–4

workstation, *Part 1*, 7–7

WISS, *Part 1*, 4–2, 8–3

Workstation attributes, *Part 1*, 6–16, 6–17, 6–22,

6–29, 6–40, 6–45, 6–49, 6–58

Workstation identifier, *Part 1*, 9–1

Workstation-independent primitive attributes,

*Part 1*, 6–7, 6–9, 6–11, 6–12, 6–13, 6–14, 6–19,

6–20, 6–21, 6–24, 6–25, 6–26, 6–27, 6–28,

6–31, 6–34, 6–35, 6–36, 6–37, 6–38, 6–39,

6–41, 6–42, 6–43, 6–44, 6–47, 6–48, 6–51,

6–53, 6–54, 6–56, 6–57

Workstations

activating, *Part 1*, 4–5, 4–9

attributes, *Part 1*, 6–1

clearing the surface, *Part 1*, 4–10

closing, *Part 1*, 4–5

deactivating, *Part 1*, 4–5, 4–13

deferral state, *Part 1*, 4–21

definition of, *Part 1*, 4–2

description tables, *Part 1*, 4–1

device coordinates, *Part 1*, 7–1

device manipulation

ESCAPE, *Part 1*, 4–14

device number, *Part 1*, 9–1

environment, *Part 1*, 4–1

environment functions

ACTIVATE WORKSTATION, *Part 1*, 4–9

CLOSE WORKSTATION, *Part 1*, 4–12

DEACTIVATE WORKSTATION, *Part 1*,  
4–13

OPEN WORKSTATION, *Part 1*, 4–19

foreground and background colors, *Part 1*, 6–5

identifiers

input, *Part 1*, 9–1

implicit regeneration, *Part 1*, 4–21

implicit regenerations

transformations, *Part 1*, 7–8

list of errors, *Part 2*, A–3 to A–5

maximum device coordinates, *Part 1*, 7–8

nominal sizes, *Part 1*, 6–1

opening, *Part 1*, 4–4

segments, *Part 1*, 4–20

state list

attributes, *Part 1*, 6–3

Workstations

state list (cont'd)

color model, *Part 1*, 6–16

edge bundle table, *Part 1*, 6–22

fill area bundle table, *Part 1*, 6–29

pattern style bundle table, *Part 1*, 6–40

polyline bundle table, *Part 1*, 6–45

polymarker bundle table, *Part 1*, 6–49

text bundle table, *Part 1*, 6–58

stored segments, *Part 1*, 8–1

surface, *Part 1*, 7–1

surface control, *Part 1*, 4–6

surface regeneration, *Part 1*, 4–7

transformations, *Part 1*, 1–3, 7–7 to 7–8

types, *Part 1*, 4–2

metafile, *Part 1*, 10–1

update

segments, *Part 1*, 8–3

updating, *Part 1*, 4–23

Workstation type

default, *Part 1*, 2–2, 3–3

defined, *Part 1*, 2–2, 3–3

specifying on ULTRIX, *Part 1*, 3–3

specifying on VMS, *Part 1*, 2–2

World coordinates, *Part 1*, 7–1

See also Transformations

fixed points, *Part 1*, 8–7

origin, *Part 1*, 7–3

WRITE ITEM TO GKSM function, *Part 1*, 10–9

Writing to metafiles, *Part 1*, 10–2