

DEC GKS

GKS3D\$ Binding
Reference Manual, Part 2

Order Number: AA-PQP8A-TE

June 1992

This manual describes the GKS3D\$ native interface functions provided for DEC GKS™.

Revision/Update Information: This is a new manual.

**Digital Equipment Corporation
Maynard, Massachusetts**

First Printing, 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 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, DEC FORTRAN, 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.

ZK5679

This manual is available on CDROM.

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

Preface	ix
11 Inquiry Functions	
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
11.2 Function Descriptions	11-4
INQUIRE CHOICE DEVICE STATE	11-5
INQUIRE CHOICE DEVICE STATE 3	11-7
INQUIRE CLIPPING	11-9
INQUIRE CLIPPING 3	11-10
INQUIRE COLOUR FACILITIES	11-11
INQUIRE COLOUR MODEL	11-12
INQUIRE COLOUR MODEL FACILITIES	11-13
INQUIRE COLOUR REPRESENTATION	11-14
INQUIRE CURRENT HLHSR IDENTIFIER VALUE	11-15
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	11-16
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3	11-19
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	11-22
INQUIRE CURRENT PICK IDENTIFIER VALUE	11-23
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	11-24
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3	11-26
INQUIRE DEFAULT CHOICE DEVICE DATA	11-28
INQUIRE DEFAULT CHOICE DEVICE DATA 3	11-30
INQUIRE DEFAULT DEFERRAL STATE VALUES	11-32
INQUIRE DEFAULT LOCATOR DEVICE DATA	11-33
INQUIRE DEFAULT LOCATOR DEVICE DATA 3	11-34
INQUIRE DEFAULT PICK DEVICE DATA	11-35
INQUIRE DEFAULT PICK DEVICE DATA 3	11-37
INQUIRE DEFAULT STRING DEVICE DATA	11-39
INQUIRE DEFAULT STRING DEVICE DATA 3	11-41
INQUIRE DEFAULT STROKE DEVICE DATA	11-43
INQUIRE DEFAULT STROKE DEVICE DATA 3	11-45
INQUIRE DEFAULT VALUATOR DEVICE DATA	11-47
INQUIRE DEFAULT VALUATOR DEVICE DATA 3	11-49
INQUIRE DISPLAY SPACE SIZE	11-51
INQUIRE DISPLAY SPACE SIZE 3	11-52

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	11-53
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	11-55
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3	11-57
INQUIRE EDGE FACILITIES	11-59
INQUIRE EDGE REPRESENTATION	11-60
INQUIRE FILL AREA FACILITIES	11-62
INQUIRE FILL AREA REPRESENTATION	11-63
INQUIRE GENERALIZED DRAWING PRIMITIVE	11-64
INQUIRE GENERALIZED DRAWING PRIMITIVE 3	11-65
INQUIRE HLHSR FACILITIES	11-66
INQUIRE HLHSR MODE	11-67
INQUIRE INPUT QUEUE OVERFLOW	11-68
INQUIRE LEVEL OF GKS	11-69
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	11-70
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3	11-71
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	11-72
INQUIRE LIST OF COLOUR INDICES	11-73
INQUIRE LIST OF EDGE INDICES	11-74
INQUIRE LIST OF FILL AREA INDICES	11-75
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	11-76
INQUIRE LIST OF PATTERN INDICES	11-77
INQUIRE LIST OF POLYLINE INDICES	11-78
INQUIRE LIST OF POLYMARKER INDICES	11-79
INQUIRE LIST OF TEXT INDICES	11-80
INQUIRE LIST OF VIEW INDICES	11-81
INQUIRE LOCATOR DEVICE STATE	11-82
INQUIRE LOCATOR DEVICE STATE 3	11-84
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	11-86
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3	11-87
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	11-88
INQUIRE MORE SIMULTANEOUS EVENTS	11-89
INQUIRE NAME OF OPEN SEGMENT	11-90
INQUIRE NORMALIZATION TRANSFORMATION	11-91
INQUIRE NORMALIZATION TRANSFORMATION 3	11-92
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	11-93
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	11-94
INQUIRE OPERATING STATE VALUE	11-95
INQUIRE PATTERN FACILITIES	11-96

INQUIRE PATTERN REPRESENTATION	11-97
INQUIRE PICK DEVICE STATE	11-98
INQUIRE PICK DEVICE STATE 3	11-100
INQUIRE PIXEL	11-102
INQUIRE PIXEL ARRAY.....	11-103
INQUIRE PIXEL ARRAY DIMENSIONS	11-105
INQUIRE POLYLINE FACILITIES	11-106
INQUIRE POLYLINE REPRESENTATION.....	11-108
INQUIRE POLYMARKER FACILITIES	11-110
INQUIRE POLYMARKER REPRESENTATION	11-112
INQUIRE PREDEFINED COLOUR REPRESENTATION	11-114
INQUIRE PREDEFINED EDGE REPRESENTATION	11-115
INQUIRE PREDEFINED FILL AREA REPRESENTATION	11-116
INQUIRE PREDEFINED PATTERN REPRESENTATION	11-117
INQUIRE PREDEFINED POLYLINE REPRESENTATION	11-118
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	11-119
INQUIRE PREDEFINED TEXT REPRESENTATION.....	11-120
INQUIRE PREDEFINED VIEW REPRESENTATION.....	11-121
INQUIRE SEGMENT ATTRIBUTES	11-122
INQUIRE SEGMENT ATTRIBUTES 3	11-124
INQUIRE SET OF ACTIVE WORKSTATIONS	11-126
INQUIRE SET OF ASSOCIATED WORKSTATIONS	11-127
INQUIRE SET OF OPEN WORKSTATIONS.....	11-128
INQUIRE SET OF SEGMENT NAMES IN USE	11-129
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	11-130
INQUIRE STRING DEVICE STATE	11-131
INQUIRE STRING DEVICE STATE 3.....	11-133
INQUIRE STROKE DEVICE STATE.....	11-135
INQUIRE STROKE DEVICE STATE 3	11-137
INQUIRE TEXT EXTENT	11-139
INQUIRE TEXT EXTENT 3	11-140
INQUIRE TEXT FACILITIES	11-141
INQUIRE TEXT REPRESENTATION	11-143
INQUIRE VALUATOR DEVICE STATE	11-145
INQUIRE VALUATOR DEVICE STATE 3	11-147
INQUIRE VIEW FACILITIES	11-149
INQUIRE VIEW REPRESENTATION 3	11-150
INQUIRE WORKSTATION CATEGORY	11-152
INQUIRE WORKSTATION CLASSIFICATION	11-153
INQUIRE WORKSTATION CONNECTION AND TYPE	11-154
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	11-155
INQUIRE WORKSTATION MAXIMUM NUMBERS	11-157
INQUIRE WORKSTATION STATE	11-158
INQUIRE WORKSTATION TRANSFORMATION	11-159
INQUIRE WORKSTATION TRANSFORMATION 3	11-160

12 Error-Handling Functions

12.1	Function Descriptions	12-2
	EMERGENCY CLOSE GKS	12-3
	ERROR HANDLING	12-4
	ERROR LOGGING	12-5
	SET ERROR HANDLER	12-6
12.2	Program Example	12-7

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	Implementation-Specific Errors	A-19
A.13	Fatal Errors	A-36

B Constants

C Program Examples

D Language-Specific Programming Information

D.1	Passing Arguments by Descriptor	D-1
D.2	Programming in BASIC	D-2
D.3	Programming in VAX C	D-2
D.4	Programming in VAX COBOL	D-2
D.5	Programming in VAX Pascal	D-5

E Initial Attribute Values

E.1	Initial Polyline Attributes	E-1
E.2	Initial Polymarker Attributes	E-1
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	12-7
D-1	Macro Subroutine Used to Build Array Descriptors	D-3
D-2	A Sample COBOL Program Using the Subroutine BUILDESC	D-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	Implementation-Specific Errors	A-19
A-13	Fatal Errors	A-36
B-1	GKS3D\$ Constants	B-1
C-1	Binding-Specific Code Examples	C-1
F-1	Global Differences	F-1

Preface

This manual contains complete descriptions for the GKS3D\$ native interface 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 GKS3D\$ binding interface.
- Appendix C provides a list of code examples available throughout this manual, listed alphabetically, by function.
- Appendix D provides language-specific programming information.
- 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 C Binding Reference Manual*—for programmers who need specific syntax and argument descriptions for the C 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

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.
Boldface text	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:

Table/List	Description
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 (VAXstation 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:

```
gks3d$inq_level ( err_stat, gks_level )
```

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 *err_stat*. If this function encounters an error condition (see Section 11.1.1 for detailed information), DEC GKS returns an error status code in the *err_stat* argument. This function returns the level of the DEC GKS implementation with which you are working in the write-only argument *gks_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:

```
gks3d$inq_locator_state ( ws_id, dev_num, value_type, max_rec_len, err_stat,  
op_mode, echo_flag, trans_num, world_loc, pet,  
echo_area, ret_buf_len, data_rec, tot_rec_len )
```

The first three arguments (*ws_id*, *dev_num*, *value_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 *err_stat* 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 *err_stat* argument.

If a value is not presently available, the inquiry function returns a number, corresponding to an appropriate DEC GKS error message, in the *err_stat* argument. If the value passed to the *err_stat* 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
GKS3D\$K_VALUE_SET	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.
GKS3D\$K_VALUE_REALIZED	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.

11.2 Function Descriptions

This section describes the DEC GKS inquiry functions in detail.

INQUIRE CHOICE DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_choice_state (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, init_status, choice, pet, echo_area, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Choice device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in choice data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_status	Integer	Write	Reference	Initial choice status.
choice	Integer	Write	Reference	Initial choice value.
pet	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Choice input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode

INQUIRE CHOICE DEVICE STATE

Defined Argument	Constant	Description
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the choice device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the choice input data record before calling the INQUIRE CHOICE DEVICE STATE function.

The information returned by this function depends on the choice input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (choice input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the choice input data record.

The language-specific data types for the choice input data record contain fields for the number of choice alternatives and string, string length, and prompt state arrays. You must initialize these arrays so they are large enough to contain all the existing values. If you first call this function with the data record size equal to SIZEOF (*number* field in the choice input data record), DEC GKS returns the number of choice alternatives in the *number* argument of the choice input data record for the PET number returned in the argument *pr_echo_type*. Once you obtain the number of choice alternatives, initialize the necessary arrays in the input data record to contain all the possible values. Call the function a second time to obtain the full state information.

See Also

INITIALIZE CHOICE

SET CHOICE MODE

SIZEOF

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

INQUIRE CHOICE DEVICE STATE 3

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_choice_state3 (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, init_status, choice, pet, echo_vol, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Choice device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in choice data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_status	Integer	Write	Reference	Initial choice status.
choice	Integer	Write	Reference	Initial choice value.
pet	Integer	Write	Reference	Prompt and echo type.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Choice input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode

INQUIRE CHOICE DEVICE STATE 3

Defined Argument	Constant	Description
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the choice device. See the language-dependent header file (*gks3d*defs.**) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the choice input data record before calling the INQUIRE CHOICE DEVICE STATE 3 function.

The information returned by this function depends on the choice input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (choice input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the choice input data record.

The language-specific data types for the choice input data record contain fields for the number of choice alternatives and string, string length, and prompt state arrays. You must initialize these arrays so they are large enough to contain all the existing values. If you first call this function with the data record size equal to SIZEOF (*number* field in the choice input data record), DEC GKS returns the number of choice alternatives in the *number* argument of the choice input data record for the PET number returned in the argument *pr_echo_type*. Once you obtain the number of choice alternatives, initialize the necessary arrays in the input data record to contain all the possible values. Call the function a second time to obtain the full state information.

See Also

INITIALIZE CHOICE 3

SET CHOICE MODE

SIZEOF

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

INQUIRE CLIPPING

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_clip (err_stat, clip_flag, clip_rect)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
clip_flag	Integer (constant)	Write	Reference	Clipping flag setting
clip_rect	Array of 4 reals	Write	Reference	Clipping rectangle, in the order XMIN, XMAX, YMIN, YMAX

Constants

Defined Argument	Constant	Description
clip_flag	GKS3D\$K_NOCLIP	Clipping disabled
	GKS3D\$K_CLIP	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

INQUIRE CLIPPING 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_clip3 (err_stat, clip_flag, clip_vol)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
clip_flag	Integer (constant)	Write	Reference	Clipping flag setting
clip_vol	Array of 6 reals	Write	Reference	Clipping volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX

Constants

Defined Argument	Constant	Description
clip_flag	GKS3D\$K_NOCLIP	Clipping disabled
	GKS3D\$K_CLIP	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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_colour_fac (ws_type, err_stat, n_colours, colour_flag, n_col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
err_stat	Integer	Write	Reference	Error indicator.
n_colours	Integer	Write	Reference	Number of available colors. If the function returns a value of 0 to this argument, a continuous range of colors is available.
colour_flag	Integer (constant)	Write	Reference	Color or monochrome flag.
n_col_ind	Integer	Write	Reference	Number of color indexes.

Constants

Defined Argument	Constant	Description
colour_flag	GKS3D\$K_MONOCHROME	Monochrome device
	GKS3D\$K_COLOUR	Color 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

INQUIRE COLOUR MODEL

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_colour_model (ws_id, err_stat, model)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
err_stat	Integer	Write	Reference	Error indicator
model	Integer (constant)	Write	Reference	Color model

Constants

Defined Argument	Constant	Description
model	GKS3D\$K_COLOUR_MODEL_RGB	Red, green, and blue color model
	GKS3D\$K_COLOUR_MODEL_CIE	Commission Internationale de l'Eclairage color model
	GKS3D\$K_COLOUR_MODEL_HSV	Hue, saturation, and value color model
	GKS3D\$K_COLOUR_MODEL_HLS	Hue, lightness, and saturation color model

Description

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

INQUIRE COLOUR MODEL FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_colour_model_fac (ws_type, start, err_stat, def_model, n_models,
list_models_d, ret_models, n_col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
start	Integer	Read	Reference	Offset into list of color model identifiers. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
def_model	Integer	Write	Reference	Default color model.
n_models	Integer	Write	Reference	Number of color models available.
list_models_d	Integer	Write	Descriptor	List of color models.
n_col_ind	Integer	Write	Reference	Number of predefined color indexes for the workstation.

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

INQUIRE COLOUR REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_colour_rep (ws_id, col_ind, value_type, err_stat, comp1, comp2, comp3)`

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier
<code>col_ind</code>	Integer	Read	Reference	Color index
<code>value_type</code>	Integer (constant)	Read	Reference	Type of returned values
<code>err_stat</code>	Integer	Write	Reference	Error indicator
<code>comp1,</code> <code>comp2,</code> <code>comp3</code>	Real	Write	Reference	Color specification

Constants

Defined Argument	Constant	Description
<code>value_type</code>	<code>GKS3D\$K_VALUE_SET</code>	Use the exact state list values.
	<code>GKS3D\$K_VALUE_REALIZED</code>	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

gks3d\$inq_hlhrs_id (err_stat, hlhrs_id)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
hlhrs_id	Integer	Write	Reference	HLHSR identifier

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

gks3d\$inq_indiv_attb (err_stat, pline_type, pline_wid, pline_col_ind, pmark_type, pmark_size, pmark_col_ind, text_font, text_prec, char_exp, char_space, text_col_ind, int_style, style_index, fill_col_ind, asf)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator.
pline_type	Integer (constant)	Write	Reference	Line type.
pline_wid	Real	Write	Reference	Line width scale factor.
pline_col_ind	Integer	Write	Reference	Polyline color index.
pmark_type	Integer (constant)	Write	Reference	Marker type.
pmark_size	Real	Write	Reference	Marker size scale factor.
pmark_col_ind	Integer	Write	Reference	Polymarker color index.
text_font	Integer	Write	Reference	Font number.
text_prec	Integer (constant)	Write	Reference	Text precision.
char_exp	Real	Write	Reference	Character expansion factor. This does not affect the character height.
char_space	Real	Write	Reference	Character spacing. Positive values increase the space between characters; negative values decrease the space. The value 0 places the character bodies adjacent to one another.
text_col_ind	Integer	Write	Reference	Text color index.
int_style	Integer (constant)	Write	Reference	Fill area interior style.
style_index	Integer	Write	Reference	Fill area style index.
fill_col_ind	Integer	Write	Reference	Fill area color index.
asf	Array of 13 integers (constant)	Write	Reference	Aspect source flags.

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

Constants

Defined Argument	Constant	Description
pline_type	GKS3D\$K_LINETYPE_SOLID	Solid line.
	GKS3D\$K_LINETYPE_DASHED	Dashed line.
	GKS3D\$K_LINETYPE_DOTTED	Dotted line.
	GKS3D\$K_LINETYPE_DASHED_DOTTED	Dashed-dotted line.
pmark_type	GKS3D\$K_MARKTYPE_DOT	Dot.
	GKS3D\$K_MARKTYPE_PLUS	Plus sign.
	GKS3D\$K_MARKTYPE_ASTERISK	Asterisk.
	GKS3D\$K_MARKTYPE_CIRCLE	Circle.
	GKS3D\$K_MARKTYPE_DIAGONAL_CROSS	Diagonal cross.
text_prec	GKS3D\$K_TEXT_PREC_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS3D\$K_TEXT_PREC_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GKS3D\$K_TEXT_PREC_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
int_style	GKS3D\$K_INTSTYLE_HOLLOW	Hollow interior.
	GKS3D\$K_INTSTYLE_SOLID	Solid interior.
	GKS3D\$K_INTSTYLE_PATTERN	Patterned interior.
	GKS3D\$K_INTSTYLE_HATCH	Hatched interior.
asf	GKS3D\$K_ASF_BUNDLED	Bundled attributes.
	GKS3D\$K_ASF_INDIVIDUAL	Individual attributes.

Description

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

The aspect source flags are as follows:

1. Polyline type
2. Polyline width
3. Polyline color
4. Polymarker type

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

5. Polymarker size
6. Polymarker color
7. Text font and precision
8. Character expansion
9. Character spacing
10. Text color
11. Fill area interior style
12. Fill area style index
13. Fill area color

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

gks3d\$inq_indiv_attb3 (err_stat, pline_type, pline_wid, pline_col_ind, pmark_type, pmark_size, pmark_col_ind, text_font, text_prec, char_exp, char_space, text_col_ind, int_style, style_index, fill_col_ind, edge_flag, edge_type, edge_sf, edge_col_ind, asf3)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator.
pline_type	Integer (constant)	Write	Reference	Line type.
pline_wid	Real	Write	Reference	Line width scale factor.
pline_col_ind	Integer	Write	Reference	Polyline color index.
pmark_type	Integer (constant)	Write	Reference	Marker type.
pmark_size	Real	Write	Reference	Marker size scale factor.
pmark_col_ind	Integer	Write	Reference	Polymarker color index.
text_font	Integer	Write	Reference	Font number.
text_prec	Integer (constant)	Write	Reference	Text precision.
char_exp	Real	Write	Reference	Character expansion factor. This does not affect the height of the characters.
char_space	Real	Write	Reference	Character spacing. Positive values increase the space between characters; negative values decrease the space. The value 0 places the character bodies adjacent to one another.
text_col_ind	Integer	Write	Reference	Text color index.
int_style	Integer (constant)	Write	Reference	Fill area interior style.
style_index	Integer	Write	Reference	Fill area style index.
fill_col_ind	Integer	Write	Reference	Fill area color index.
edge_flag	Integer (constant)	Write	Reference	Edge flag.
edge_type	Integer (constant)	Write	Reference	Edge type.
edge_sf	Real	Write	Reference	Edge width scale factor.
edge_col_ind	Integer	Write	Reference	Edge color index.

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3

Argument	Data Type	Access	Passed by	Description
asf3	Array of 17 integers (constant)	Write	Reference	Aspect source flags.

Constants

Defined Argument	Constant	Description
pline_type	GKS3D\$K_LINETYPE_SOLID	Solid line.
	GKS3D\$K_LINETYPE_DASHED	Dashed line.
	GKS3D\$K_LINETYPE_DOTTED	Dotted line.
	GKS3D\$K_LINETYPE_DASHED_DOTTED	Dashed-dotted line.
pmark_type	GKS3D\$K_MARKTYPE_DOT	Dot.
	GKS3D\$K_MARKTYPE_PLUS	Plus sign.
	GKS3D\$K_MARKTYPE_ASTERISK	Asterisk.
	GKS3D\$K_MARKTYPE_CIRCLE	Circle.
	GKS3D\$K_MARKTYPE_DIAGONAL_CROSS	Diagonal cross.
text_prec	GKS3D\$K_TEXT_PREC_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS3D\$K_TEXT_PREC_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GKS3D\$K_TEXT_PREC_STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
int_style	GKS3D\$K_INTSTYLE_HOLLOW	Hollow interior.
	GKS3D\$K_INTSTYLE_SOLID	Solid interior.
	GKS3D\$K_INTSTYLE_PATTERN	Patterned interior.
	GKS3D\$K_INTSTYLE_HATCH	Hatched interior.
edge_flag	GKS3D\$K_EDGE_OFF	Edge off.
	GKS3D\$K_EDGE_ON	Edge on.

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3

Defined Argument	Constant	Description
edge_type	GKS3D\$K_EDGE_SOLID	Solid edge.
	GKS3D\$K_EDGE_DASHED	Dashed edge.
	GKS3D\$K_EDGE_DOTTED	Dotted edge.
	GKS3D\$K_EDGE_DASHED_DOTTED	Dashed-dotted edge.
asf3	GKS3D\$K_ASF_BUNDLED	Bundled attributes.
	GKS3D\$K_ASF_INDIVIDUAL	Individual attributes.

Description

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

The aspect source flags are as follows:

1. Polyline type
2. Polyline width
3. Polyline color
4. Polymarker type
5. Polymarker size
6. Polymarker color
7. Text font and precision
8. Character expansion
9. Character spacing
10. Text color
11. Fill area interior style
12. Fill area style index
13. Fill area color
14. Edge flag
15. Edge type
16. Edge width
17. Edge color

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

gks3d\$inq_current_xformno (err_stat, trans_num)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
trans_num	Integer	Write	Reference	Current normalization transformation number

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

`gks3d$inq_pick_id (err_stat, pick_id)`

Argument	Data Type	Access	Passed by	Description
<code>err_stat</code>	Integer	Write	Reference	Error indicator
<code>pick_id</code>	Integer	Write	Reference	Pick identifier

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

gks3d\$inq_prim_attr (err_stat, bundle_ind, text_height, char_up_vec, char_width, text_b_vec, char_path, char_align, patt_width, patt_height, patt_ref_pt)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator.
bundle_ind	Array of 4 integers	Write	Reference	Bundle indexes, in the order polyline, polymarker, text, and fill area index.
text_height	Real	Write	Reference	Character height.
char_up_vec	Array of 2 reals	Write	Reference	Character up vector.
char_width	Real	Write	Reference	Character width.
text_b_vec	Array of 2 reals	Write	Reference	Text base vectors.
char_path	Integer (constant)	Write	Reference	Text path.
char_align	Array of 2 integers (constant)	Write	Reference	Text alignment. Specify the horizontal alignment as the first element of the array, and the vertical alignment as the second element.
patt_width	Array of 2 reals	Write	Reference	Pattern width vector.
patt_height	Array of 2 reals	Write	Reference	Pattern height vector.
patt_ref_pt	Array of 2 reals	Write	Reference	Pattern reference point.

Constants

Defined Argument	Constant	Description
char_path	GKS3D\$K_TEXT_PATH_RIGHT	Text reads from left to right.
	GKS3D\$K_TEXT_PATH_LEFT	Text reads from right to left.
	GKS3D\$K_TEXT_PATH_UP	Text reads from bottom to top.
	GKS3D\$K_TEXT_PATH_DOWN	Text reads from top to the bottom.

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

Defined Argument	Constant	Description
Horizontal alignment	GKS3D\$K_TEXT_HALIGN_ NORMAL	Normal.
	GKS3D\$K_TEXT_HALIGN_LEFT	Left.
	GKS3D\$K_TEXT_HALIGN_ CENTRE	Center.
	GKS3D\$K_TEXT_HALIGN_RIGHT	Right.
Vertical alignment	GKS3D\$K_TEXT_VALIGN_ NORMAL	Normal.
	GKS3D\$K_TEXT_VALIGN_TOP	Top.
	GKS3D\$K_TEXT_VALIGN_CAP	Cap.
	GKS3D\$K_TEXT_VALIGN_HALF	Half.
	GKS3D\$K_TEXT_VALIGN_BASE	Base.
	GKS3D\$K_TEXT_VALIGN_ 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

gks3d\$inq_prim_attb3 (err_stat, bundle_ind, text_height, char_up_vec, char_width, text_b_vec, char_path, char_align, patt_width, patt_height, patt_ref_pt, patt_ref_vec1, patt_ref_vec2)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator.
bundle_ind	Array of 6 integers	Write	Reference	Bundle indexes, in the order polyline, polymarker, text, fill area, edge, and view index.
text_height	Real	Write	Reference	Character height.
char_up_vec	Array of 2 reals	Write	Reference	Character up vector.
char_width	Real	Write	Reference	Character width.
text_b_vec	Array of 2 reals	Write	Reference	Text base vectors.
char_path	Integer (constant)	Write	Reference	Text path.
char_align	Array of 2 integers (constant)	Write	Reference	Text alignment. Specify the horizontal alignment as the first element of the array, and the vertical alignment as the second element.
patt_width	Array of 2 reals	Write	Reference	Pattern width vector.
patt_height	Array of 2 reals	Write	Reference	Pattern height vector.
patt_ref_pt	Array of 3 reals	Write	Reference	Pattern reference point.
patt_ref_vec1, patt_ref_vec2	Array of 3 reals	Write	Reference	Current pattern reference vectors.

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3

Constants

Defined Argument	Constant	Description
char_path	GKS3D\$K_TEXT_PATH_RIGHT	Text reads from left to right.
	GKS3D\$K_TEXT_PATH_LEFT	Text reads from right to left.
	GKS3D\$K_TEXT_PATH_UP	Text reads from bottom to top.
	GKS3D\$K_TEXT_PATH_DOWN	Text reads from top to bottom.
Horizontal alignment	GKS3D\$K_TEXT_HALIGN_NORMAL	Normal.
	GKS3D\$K_TEXT_HALIGN_LEFT	Left.
	GKS3D\$K_TEXT_HALIGN_CENTRE	Center.
	GKS3D\$K_TEXT_HALIGN_RIGHT	Right.
Vertical alignment	GKS3D\$K_TEXT_VALIGN_NORMAL	Normal.
	GKS3D\$K_TEXT_VALIGN_TOP	Top.
	GKS3D\$K_TEXT_VALIGN_CAP	Cap.
	GKS3D\$K_TEXT_VALIGN_HALF	Half.
	GKS3D\$K_TEXT_VALIGN_BASE	Base.
	GKS3D\$K_TEXT_VALIGN_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

`gks3d$inq_def_choice_data (ws_type, dev_num, start, in_rec_len, err_stat, n_choice, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_area, act_rec_len, choice, ret_rec_len)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type.
<code>dev_num</code>	Integer	Read	Reference	Choice device number.
<code>start</code>	Integer	Read	Reference	Offset into list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
<code>in_rec_len</code>	Integer	Read	Reference	Maximum number of bytes to put in choice data record.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>n_choice</code>	Integer	Write	Reference	Maximum number of available choices.
<code>n_pr_echo_tot</code>	Integer	Write	Reference	Total number of prompt and echo types available.
<code>pet_list_d</code>	Array of integers	Write	Descriptor	List of prompt and echo types.
<code>n_pr_echo_ret</code>	Integer	Write	Reference	Number of prompt and echo types returned.
<code>echo_area</code>	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
<code>act_rec_len</code>	Integer	Write	Reference	Actual size of the data record.
<code>choice</code>	Record	Write	Reference	Choice input data record. See the appropriate language-dependent header file (<code>gks3d*defs.*</code>) for the specific data type and data record structure.
<code>ret_rec_len</code>	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

INQUIRE DEFAULT CHOICE DEVICE DATA

The default data record returned by this function is the input data record associated with PET 1. No title string is returned by this function. See the Choice Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a choice device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the choice input data record before calling the INQUIRE DEFAULT CHOICE DEVICE DATA function.

The information returned by this function depends on the choice input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (choice input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the choice input data record.

The language-specific data type for the choice input data record contains fields for the number of choice strings, the strings, and the string length arrays. You must initialize these arrays so they are large enough to hold all the choice strings before the complete data record can be returned. If you first call this function with the data record size equal to SIZEOF (*number* field in the choice input data record), the function returns the number of choices in the *number* argument of the choice input data record for PET 1.

Once you obtain the number of choices, initialize the string and string length arrays in the input data record to contain all the choice strings. Call the function a second time to obtain the full default information.

See Also

SIZEOF

INQUIRE DEFAULT CHOICE DEVICE DATA 3

INQUIRE DEFAULT CHOICE DEVICE DATA 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_def_choice_data3 (ws_type, dev_num, start, in_rec_len, err_stat,  
                             n_choice, n_pr_echo_tot, pet_list_d, n_pr_echo_ret,  
                             echo_vol, act_rec_len, choice, ret_rec_len)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Choice device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Maximum number of bytes to put in choice data record.
err_stat	Integer	Write	Reference	Error indicator.
n_choice	Integer	Write	Reference	Maximum number of available choices.
n_pr_echo_tot	Integer	Write	Reference	Total number of prompt and echo types available.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record.
choice	Record	Write	Reference	Choice input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

INQUIRE DEFAULT CHOICE DEVICE DATA 3

The default data record returned by this function is the input data record associated with PET 1. No title string is returned by this function. See the Choice Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a choice device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the choice input data record before calling the INQUIRE DEFAULT CHOICE DEVICE DATA function.

The information returned by this function depends on the choice input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (choice input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the choice input data record.

The language-specific data type for the choice input data record contains fields for the number of choice strings, the strings, and the string length arrays. You must initialize these arrays so they are large enough to hold all the choice strings before the complete data record can be returned. If you first call this function with the data record size equal to SIZEOF (*number* field in the choice input data record), the function returns the number of choices in the *number* argument of the choice input data record for PET 1.

Once you obtain the number of choices, initialize the string and string length arrays in the input data record to contain all the choice strings. Call the function a second time to obtain the full default information.

See Also

SIZEOF

INQUIRE DEFAULT DEFERRAL STATE VALUES

INQUIRE DEFAULT DEFERRAL STATE VALUES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_defer_state (ws_type, err_stat, defer_mode, regen_mode)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
defer_mode	Integer (constant)	Write	Reference	Deferral mode
regen_mode	Integer (constant)	Write	Reference	Implicit regeneration mode

Constants

Defined Argument	Constant	Description
defer_mode	GKS3D\$K_ASAP	Generate images as soon as possible.
	GKS3D\$K_BNIG	Generate images before input is requested globally.
	GKS3D\$K_BNIL	Generate images before input is requested locally.
	GKS3D\$K_ASTI	Generate images sometime. The exact time is not guaranteed.
regen_mode	GKS3D\$K_IRG_SUPPRESSED	Image regeneration is suppressed.
	GKS3D\$K_IRG_ALLOWED	Image 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

gks3d\$inq_def_locator_data (ws_type, dev_num, start, n_buf, err_stat, init_pos, n_pr_echo, pet_list_d, n_pr_echo_ret, echo_area, act_rec_len, locator, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Locator device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
n_buf	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
init_pos	Array of 2 reals	Write	Reference	Default initial position.
n_pr_echo	Integer	Write	Reference	Number of prompt and echo types in array available.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
locator	Array of integers	Write	Reference	Locator input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

The default data record returned by this function is the input data record associated with PET 1. There are no data record components associated with locator-class PET 1. All the default information is returned in the other function arguments and the length of the returned data record is 0.

INQUIRE DEFAULT LOCATOR DEVICE DATA 3

INQUIRE DEFAULT LOCATOR DEVICE DATA 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_def_locator_data3 (ws_type, dev_num, start, n_buf, err_stat, init_pos,  
n_pr_echo, pr_echo_arr, n_pr_echo_ret, echo_vol,  
act_rec_len, locator, ret_rec_len)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Locator device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
n_buf	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
init_pos	Array of 3 reals	Write	Reference	Default initial position.
n_pr_echo	Integer	Write	Reference	Number of prompt and echo types in array.
pr_echo_arr	Array of integers	Write	Descriptor	Available prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
locator	Array of integers	Write	Reference	Locator input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

The default data record returned by this function is the input data record associated with PET 1. There are no data record components associated with locator-class PET 1. All the default information is returned in the other arguments and the length of the returned data record is 0.

INQUIRE DEFAULT PICK DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_pick_data (ws_type, dev_num, start, in_rec_len, err_stat, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_area, act_rec_len, pick, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Pick device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
n_pr_echo_tot	Integer	Write	Reference	Number of prompt and echo types in array.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
pick	Record	Write	Reference	Pick input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

The default data record returned by this function is the input data record associated with PET 1. See the Pick Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a pick device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF

INQUIRE DEFAULT PICK DEVICE DATA

utility function to determine the size of the choice input data record before calling the INQUIRE DEFAULT PICK DEVICE DATA function.

The information returned by this function depends on the pick input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (pick input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the pick input data record.

See Also

SIZEOF

INQUIRE DEFAULT PICK DEVICE DATA 3
Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_def_pick_data3 (ws_type, dev_num, start, in_rec_len, err_stat,
                           n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_vol,
                           act_rec_len, pick, ret_rec_len)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Pick device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
n_pr_echo_tot	Integer	Write	Reference	Number of prompt and echo types available.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
pick	Record	Write	Reference	Pick input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

The default data record returned by this function is the input data record associated with PET 1. See the Pick Class section in the Input Functions chapter for information on the data record component. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a pick device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF

INQUIRE DEFAULT PICK DEVICE DATA 3

utility function to determine the size of the choice input data record before calling the INQUIRE DEFAULT PICK DEVICE DATA 3 function.

The information returned by this function depends on the pick input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (pick input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the pick input data record.

See Also

SIZEOF

INQUIRE DEFAULT STRING DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_string_data (ws_type, dev_num, start, in_rec_len, err_stat, n_string_max, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_area, act_rec_len, string, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	String device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Maximum number of bytes to put in string data record.
err_stat	Integer	Write	Reference	Error indicator.
n_string_max	Integer	Write	Reference	Maximum string size.
n_pr_echo_tot	Integer	Write	Reference	Number of prompt and echo types available.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
string	Array of integers	Write	Reference	String input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

INQUIRE DEFAULT STRING DEVICE DATA

The default data record returned by this function is the input data record associated with PET 1. No title string is returned by this function. See the String Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a string device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the string input data record before calling the INQUIRE DEFAULT STRING DEVICE DATA function.

The information returned by this function depends on the string input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (string input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the string input data record.

See Also

SIZEOF

INQUIRE DEFAULT STRING DEVICE DATA 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_string_data3 (ws_type, dev_num, start, in_rec_len, err_stat, n_string_max, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_vol, act_rec_len, string, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	String device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Maximum number of bytes to put in string data record.
err_stat	Integer	Write	Reference	Error indicator.
n_string_max	Integer	Write	Reference	Maximum string size.
n_pr_echo_tot	Integer	Write	Reference	Number of prompt and echo types available.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Total number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
string	Record	Write	Reference	String input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

INQUIRE DEFAULT STRING DEVICE DATA 3

The default data record returned by this function is the input data record associated with PET 1. No title string is returned by this function. See the String Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a string device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the string input data record before calling the INQUIRE DEFAULT STRING DEVICE DATA 3 function.

The information returned by this function depends on the string input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (string input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the string input data record.

See Also

SIZEOF

INQUIRE DEFAULT STROKE DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_stroke_data (ws_type, dev_num, start, in_rec_len, err_stat, stroke_buf_size, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_area, act_rec_len, stroke, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Stroke device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
stroke_buf_size	Integer	Write	Reference	Maximum stroke buffer size.
n_pr_echo_tot	Integer	Write	Reference	Number of available prompt and echo types.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Number of prompt and echo types returned.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
stroke	Record	Write	Reference	Stroke input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

INQUIRE DEFAULT STROKE DEVICE DATA

The default data record returned by this function is the input data record associated with PET 1. See the Stroke Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a stroke device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the stroke input data record before calling the INQUIRE DEFAULT STROKE DEVICE DATA function.

The information returned by this function depends on the stroke input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (stroke input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the stroke input data record.

See Also

SIZEOF

INQUIRE DEFAULT STROKE DEVICE DATA 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_stroke_data3 (ws_type, dev_num, start, in_rec_len, err_stat, stroke_buf_size, n_pr_echo_tot, pet_list_d, n_pr_echo_ret, echo_vol, act_rec_len, stroke, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Stroke device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
stroke_buf_size	Integer	Write	Reference	Maximum stroke buffer size.
n_pr_echo_tot	Integer	Write	Reference	Number of available prompt and echo types.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
n_pr_echo_ret	Integer	Write	Reference	Number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record.
stroke	Array of integers	Write	Reference	Stroke input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

INQUIRE DEFAULT STROKE DEVICE DATA 3

The default data record returned by this function is the input data record associated with PET 1. See the Stroke Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a stroke device. Each of the data type names corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the stroke input data record before calling the INQUIRE DEFAULT STROKE DEVICE DATA 3 function.

The information returned by this function depends on the stroke input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (stroke input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the stroke input data record.

See Also

SIZEOF

INQUIRE DEFAULT VALUATOR DEVICE DATA

INQUIRE DEFAULT VALUATOR DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_def_valuator_data (ws_type, dev_num, start, in_rec_len, err_stat,  
init_value, num_avail_pets, pet_list_d, num_ret_pets,  
echo_area, act_rec_len, valuator, ret_rec_len)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Valuator device number.
start	Integer	Read	Reference	Offset into the list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
init_value	Integer	Write	Reference	Default initial value.
num_avail_pets	Integer	Write	Reference	Number of available prompt and echo types.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
num_ret_pets	Integer	Write	Reference	Number of prompt and echo types returned.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
valuator	Record	Write	Reference	Valuator input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

Description

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

The default data record returned by this function is the input data record associated with PET 1. See the Valuator Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a valuator device. Each of the data type names corresponds to

INQUIRE DEFAULT VALUATOR DEVICE DATA

the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the valuator input data record before calling the INQUIRE DEFAULT VALUATOR DEVICE DATA function.

The information returned by this function depends on the valuator input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (valuator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the valuator input data record.

See Also

SIZEOF

INQUIRE DEFAULT VALUATOR DEVICE DATA 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_def_valuator_data3 (ws_type, dev_num, start, in_rec_len, err_stat, init_value, num_avail_pets, pet_list_d, num_ret_pets, echo_vol, act_rec_len, valuator, ret_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Valuator device number.
start	Integer	Read	Reference	Offset into list of prompt and echo types. Set <i>start</i> to 0 to inquire from the beginning of the list.
in_rec_len	Integer	Read	Reference	Buffer size.
err_stat	Integer	Write	Reference	Error indicator.
init_value	Real	Write	Reference	Default initial value.
num_avail_pets	Integer	Write	Reference	Number of available prompt and echo types.
pet_list_d	Array of integers	Write	Descriptor	List of prompt and echo types.
num_ret_pets	Integer	Write	Reference	Number of prompt and echo types returned.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
act_rec_len	Integer	Write	Reference	Actual size of the data record, in bytes.
valuator	Record	Write	Reference	Valuator input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
ret_rec_len	Integer	Write	Reference	Size of the returned data record buffer, in bytes. Compare this value to <i>act_rec_len</i> to see if the entire data record was returned.

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.

The default data record returned by this function is the input data record associated with PET 1. See the Valuator Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks3d*defs.*) for the specific data types and structure for PET 1 for a valuator device. Each of the data type names corresponds to

INQUIRE DEFAULT VALUATOR DEVICE DATA 3

the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the valuator input data record before calling the INQUIRE DEFAULT VALUATOR DEVICE DATA 3 function.

The information returned by this function depends on the valuator input record size (the *in_rec_len* argument). If the data record size is equal to SIZEOF (valuator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the valuator input data record.

See Also

SIZEOF

INQUIRE DISPLAY SPACE SIZE

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_max_ds_size (ws_type, err_stat, units_flag, dis_size, raster_x, raster_y)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
units_flag	Integer (constant)	Write	Reference	Measurement units flag
dis_size	Array of 2 reals	Write	Reference	Display surface size, in the order X, Y
raster_x, raster_y	Integer	Write	Reference	Pixel count

Constants

Defined Argument	Constant	Description
units_flag	GKS3D\$K_METRES	Meters
	GKS3D\$K_OTHER_UNITS	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

```
gks3d$inq_max_ds_size3 (ws_type, err_stat, units_flag, dis_vol, raster_x, raster_y,  
                        raster_z)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
units_flag	Integer (constant)	Write	Reference	Measurement units flag
dis_vol	Array of reals	Write	Reference	Display size in device coordinates, in the order X, Y, Z
raster_x, raster_y, raster_z	Integer	Write	Reference	Display size in raster units

Constants

Defined Argument	Constant	Description
units_flag	GKS3D\$K_METRES	Meters
	GKS3D\$K_OTHER_UNITS	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

gks3d\$inq_dyn_mod_seg_attn (ws_type, err_stat, trans_ch_flag, vis_invis, invis_vis, highl_ch_flag, priority_ch_flag, overlap_flag, del_vis_flag)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
trans_ch_flag	Integer (constant)	Write	Reference	Workstation capability for dynamic modification with a segment transformation change
vis_invis, invis_vis	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when visibility changes from on to off
highl_ch_flag	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when highlighting changes
priority_ch_flag	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the segment priority changes
overlap_flag	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when primitives are added to an open segment
del_vis_flag	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when a segment is deleted

Constants

Defined Argument	Constant	Description
trans_ch_flag, vis_invis, invis_vis, highl_ch_flag, priority_ch_flag, overlap_flag, del_vis_flag	GKS3D\$K_IRG GKS3D\$K_IMM	Implicit regeneration Immediate regeneration

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

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

gks3d\$inq_dyn_mod_ws_atlb (ws_type, err_stat, pline_rep, pmark_rep, text_rep, fill_rep, patt_rep, colour_rep, ws_trans)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error status
pline_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the polyline representation changes
pmark_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the polymarker representation changes
text_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the text representation changes
fill_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the fill area representation changes
patt_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the pattern representation changes
colour_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the color representation changes
ws_trans	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the workstation transformation changes

Constants

Defined Argument	Constant	Description
pline_rep, pmark_rep, text_rep, fill_rep, patt_rep, colour_rep, ws_trans	GKS3D\$K_IRG GKS3D\$K_IMM	Implicit regeneration Immediate regeneration

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

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.

See Also

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

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_dyn_mod_ws_atlb3 (ws_type, err_stat, pline_rep, pmark_rep, text_rep, fill_rep, patt_rep, colour_rep, ws_trans, view_rep, edge_rep, hlhr_rep)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error status
pline_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the polyline representation changes
pmark_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the polymarker representation changes
text_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the text representation changes
fill_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the fill area representation changes
patt_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the pattern representation changes
colour_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the color representation changes
ws_trans	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the workstation transformation changes
view_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the view transformation changes
edge_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the edge representation changes
hlhr_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the HLHSR mode changes

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3

Constants

Defined Argument	Constant	Description
pline_rep, pmark_rep, text_rep, fill_rep, patt_rep, colour_rep, ws_trans, view_rep, edge_rep, hlhsr_rep	GKS3D\$K_IRG GKS3D\$K_IMM	Implicit regeneration 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.

See Also

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

INQUIRE EDGE FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_edge_fac (ws_type, start, err_stat, n_types, edge, n_widths, nom_width, min_width, max_width, n_ind, n_types_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_types	Integer	Write	Reference	Number of edge types.
edge	Array of integers (constant)	Write	Descriptor	List of edge types.
n_widths	Integer	Write	Reference	Number of available edge widths.
nom_width	Real	Write	Reference	Nominal edge width.
min_width, max_width	Real	Write	Reference	Minimum and maximum edge widths.
n_ind	Integer	Write	Reference	Number of edge indexes.
n_types_ret	Integer	Write	Reference	Number of edge types returned in list.

Constants

Defined Argument	Constant	Description
edge	GKS3D\$K_EDGE_SOLID	Solid edge
	GKS3D\$K_EDGE_DASHED	Dashed edge
	GKS3D\$K_EDGE_DOTTED	Dotted edge
	GKS3D\$K_EDGE_DASHED_DOTTED	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 REPRESENTATION

INQUIRE EDGE REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_edge_rep (ws_id, edge_index, value_type, err_stat, edge_flag, edge_type, edgewidth_sf, edge_col_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
edge_index	Integer	Read	Reference	Edge index
value_type	Integer (constant)	Read	Reference	Type of returned values
err_stat	Integer	Write	Reference	Error indicator
edge_flag	Integer (constant)	Write	Reference	Edge flag
edge_type	Integer (constant)	Write	Reference	Edge type
edgewidth_sf	Real	Write	Reference	Edge width scale factor
edge_col_ind	Integer	Write	Reference	Edge color index

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
edge_flag	GKS3D\$K_EDGE_OFF	Edge off.
	GKS3D\$K_EDGE_ON	Edge on.
edge_type	GKS3D\$K_EDGE_SOLID	Solid edge.
	GKS3D\$K_EDGE_DASHED	Dashed edge.
	GKS3D\$K_EDGE_DOTTED	Dotted edge.
	GKS3D\$K_EDGE_DASHED_DOTTED	Dashed-dotted edge.

INQUIRE EDGE REPRESENTATION

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 FILL AREA FACILITIES

INQUIRE FILL AREA FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

`gks3d$inq_fill_fac (ws_type, start, err_stat, n_int_styles, int_style_list, n_hatch_styles, hatch_list, n_fill_indices, n_hatch_ret)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type.
<code>start</code>	Integer	Read	Reference	Offset into list. Set <i>start</i> to 0 to inquire from the beginning of the list.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>n_int_styles</code>	Integer	Write	Reference	Number of interior styles.
<code>int_style_list</code>	Array of 4 integers (constant)	Write	Reference	List of interior styles.
<code>n_hatch_styles</code>	Integer	Write	Reference	Number of hatch styles.
<code>hatch_list</code>	Array of integers	Write	Descriptor	List of hatch styles.
<code>n_fill_indices</code>	Integer	Write	Reference	Number of fill area indexes.
<code>n_hatch_ret</code>	Integer	Write	Reference	Number of hatch styles returned.

Constants

Defined Argument	Constant	Description
<code>int_style_list</code>	<code>GKS3D\$K_INTSTYLE_HOLLOW</code>	Hollow interior
	<code>GKS3D\$K_INTSTYLE_SOLID</code>	Solid interior
	<code>GKS3D\$K_INTSTYLE_PATTERN</code>	Patterned interior
	<code>GKS3D\$K_INTSTYLE_HATCH</code>	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 REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_fill_rep (ws_id, fill_area_ind, value_type, err_stat, int_style, style_index, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
fill_area_ind	Integer	Read	Reference	Fill area index
value_type	Integer (constant)	Read	Reference	Type of returned values
err_stat	Integer	Write	Reference	Error indicator
int_style	Integer (constant)	Write	Reference	Fill area interior style
style_index	Integer	Write	Reference	Fill area style index
col_ind	Integer	Write	Reference	Fill area color index

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
int_style	GKS3D\$K_INTSTYLE_HOLLOW	Hollow interior.
	GKS3D\$K_INTSTYLE_SOLID	Solid interior.
	GKS3D\$K_INTSTYLE_PATTERN	Patterned interior.
	GKS3D\$K_INTSTYLE_HATCH	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 GENERALIZED DRAWING PRIMITIVE

INQUIRE GENERALIZED DRAWING PRIMITIVE

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

`gks3d$inq_gdp (ws_type, gdp_id, err_stat, num_attr, att_list, nret_attr)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type.
<code>gdp_id</code>	Integer	Read	Reference	GDP identifier.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>num_attr</code>	Integer	Write	Reference	Number of attributes used by GDP.
<code>att_list</code>	Array of integers (constant)	Write	Descriptor	List of attributes used by GDP.
<code>nret_attr</code>	Integer	Write	Reference	Number of attributes returned in list. Use this argument to check that the specified array is large enough to hold all the attributes.

Constants

Defined Argument	Constant	Description
<code>att_list</code>	<code>GKS3D\$K_POLYLN_ATTRI</code>	GDP polyline attributes
	<code>GKS3D\$K_POLYMR_ATTRI</code>	GDP polymarker attributes
	<code>GKS3D\$K_TEXT_ATTRI</code>	GDP text attributes
	<code>GKS3D\$K_FILLAR_ATTRI</code>	GDP fill area 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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_gdp3 (ws_type, gdp_id, err_stat, num_attr, att_list, nret_attr)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
gdp_id	Integer	Read	Reference	GDP identifier.
err_stat	Integer	Write	Reference	Error indicator.
num_attr	Integer	Write	Reference	Number of attributes used by GDP.
att_list	Array of integers (constant)	Write	Descriptor	List of attributes used by GDP.
nret_attr	Integer	Write	Reference	Number of attributes returned in list. Use this argument to see if the specified array is large enough to hold all the attributes.

Constants

Defined Argument	Constant	Description
att_list	GKS3D\$K_POLYLN_ATTRI	GDP polyline attributes
	GKS3D\$K_POLYMR_ATTRI	GDP polymarker attributes
	GKS3D\$K_TEXT_ATTRI	GDP text attributes
	GKS3D\$K_FILLAR_ATTRI	GDP fill area attributes
	GKS3D\$K_EDGE_ATTRI	GDP edge 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

INQUIRE HLHSR FACILITIES

Operating States:

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_hlhrs_fac (ws_type, id_start, mode_start, err_stat, n_ids, id_list_d,  
                    n_ids_ret, n_modes, mode_list_d, n_modes_ret)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
id_start	Integer	Read	Reference	Offset into the HLHSR identifier array. Set <i>start</i> to 0 to inquire from the beginning of the list.
mode_start	Integer	Read	Reference	Offset into HLHSR mode array.
err_stat	Integer	Write	Reference	Error indicator.
n_ids	Integer	Write	Reference	Number of available HLHSR identifiers.
id_list_d	Array of integers	Write	Descriptor	List of HLHSR identifiers.
n_ids_ret	Integer	Write	Reference	Number of HLHSR identifiers returned in list.
n_modes	Integer	Write	Reference	Number of available HLHSR modes.
mode_list_d	Array of integers	Write	Descriptor	List of available HLHSR modes.
n_modes_ret	Integer	Write	Reference	Number of HLHSR modes returned in list.

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

Operating States

WSOP, WSAC, WGOP

Syntax

gks3d\$inq_hlhrs_mode (ws_id, err_stat, update, req_mode, cur_mode)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
err_stat	Integer	Write	Reference	Error indicator
update	Integer (constant)	Write	Reference	HLHSR update state
req_mode	Integer	Write	Reference	Requested HLHSR mode
cur_mode	Integer	Write	Reference	Current HLHSR mode

Constants

Defined Argument	Constant	Description
update	GKS3D\$K_NOTPENDING	Action not pending
	GKS3D\$K_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

INQUIRE INPUT QUEUE OVERFLOW

Operating States

WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_input_queue_overflow (err_stat, ws_id, input_class, dev_num)
```

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
ws_id	Integer	Write	Reference	Workstation identifier whose input device caused the queue to overflow
input_class	Integer (constant)	Write	Reference	Input class
dev_num	Integer	Write	Reference	Input device number

Constants

Defined Argument	Constant	Description
input_class	GKS3D\$K_INPUT_CLASS_NONE	No input class
	GKS3D\$K_INPUT_CLASS_LOCATOR	Locator input class
	GKS3D\$K_INPUT_CLASS_STROKE	Stroke input class
	GKS3D\$K_INPUT_CLASS_VALUATOR	Valuator input class
	GKS3D\$K_INPUT_CLASS_CHOICE	Choice input class
	GKS3D\$K_INPUT_CLASS_PICK	Pick input class
	GKS3D\$K_INPUT_CLASS_STRING	String input class
	GKS3D\$K_INPUT_CLASS_VIEWPORT	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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_level (err_stat, gks_level)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error status
gks_level	Integer (constant)	Write	Reference	Level of GKS

Constants

Defined Argument	Constant	Description
gks_level	GKS3D\$K_LEVEL_MA	Minimal output, no input
	GKS3D\$K_LEVEL_MB	Minimal output, request input
	GKS3D\$K_LEVEL_MC	Minimal output, full input
	GKS3D\$K_LEVEL_0A	All primitives and attributes, no input
	GKS3D\$K_LEVEL_0B	All primitives and attributes, request input
	GKS3D\$K_LEVEL_0C	All primitives and attributes, full input
	GKS3D\$K_LEVEL_1A	Basic segmentation with full output, no input
	GKS3D\$K_LEVEL_1B	Basic segmentation with full output, request input
	GKS3D\$K_LEVEL_1C	Basic segmentation with full output, full input
	GKS3D\$K_LEVEL_2A	Workstation-independent and segment storage, no input
	GKS3D\$K_LEVEL_2B	Workstation-independent and segment storage, request input
	GKS3D\$K_LEVEL_2C	Workstation-independent and segment storage, full input

Description

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

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

`gks3d$inq_avail_gdp (ws_type, start, err_stat, n_gdps, gdp_list, n_gdps_ret)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type.
<code>start</code>	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>n_gdps</code>	Integer	Write	Reference	Number of available GDPs.
<code>gdp_list</code>	Array of integers	Write	Descriptor	List of GDP identifiers.
<code>n_gdps_ret</code>	Integer	Write	Reference	Number of GDP identifiers returned in the list.

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_avail_gdp3 (ws_type, start, err_stat, n_gdps, gdp_list, n_gdps_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_gdps	Integer	Write	Reference	Number of available GDPs.
gdp_list	Array of integers	Write	Descriptor	List of GDP identifiers.
n_gdps_ret	Integer	Write	Reference	Number of GDP identifiers returned in the list.

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

gks3d\$inq_wstype_list (start, err_stat, num_wstypes, wstype_d, n_ws_ret)

Argument	Data Type	Access	Passed by	Description
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error status.
num_wstypes	Integer	Write	Reference	Number of different workstation types available.
wstype_d	Array of integers	Write	Descriptor	List of supported workstations returned.
n_ws_ret	Integer	Write	Reference	Number of workstation types returned.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_colour_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined color indexes.
list_d	Array of integers	Write	Descriptor	Color index values.
n_ind_ret	Integer	Write	Reference	Number of color indexes returned.

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

INQUIRE LIST OF EDGE INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_edge_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined edge indexes.
list_d	Array of integers	Write	Descriptor	Edge index values.
n_ind_ret	Integer	Write	Reference	Number of edge indexes returned.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_fill_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined fill area indexes.
list_d	Array of integers	Write	Descriptor	Fill area index values.
n_ind_ret	Integer	Write	Reference	Number of fill area indexes returned.

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

gks3d\$inq_xform_list (start, err_stat, n_trans, list_trans_d, n_trans_ret)

Argument	Data Type	Access	Passed by	Description
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_trans	Integer	Write	Reference	Number of defined normalization transformations.
list_trans_d	Array of integers	Write	Descriptor	List of normalization transformation numbers.
n_trans_ret	Integer	Write	Reference	Number of normalization transformation numbers returned.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pat_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined pattern indexes.
list_d	Array of integers	Write	Descriptor	List of pattern index values.
n_ind_ret	Integer	Write	Reference	Number of pattern indexes returned.

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

INQUIRE LIST OF POLYLINE INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pline_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined polyline indexes.
list_d	Array of integers	Write	Descriptor	List of polyline index values.
n_ind_ret	Integer	Write	Reference	Number of polyline indexes returned.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pmark_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined polymarker indexes.
list_d	Array of integers	Write	Descriptor	List of polymarker index values.
n_ind_ret	Integer	Write	Reference	Number of polymarker indexes returned.

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

INQUIRE LIST OF TEXT INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_text_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined text indexes.
list_d	Array of integers	Write	Descriptor	List of text index values.
n_ind_ret	Integer	Write	Reference	Number of text indexes returned.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_view_indexes (ws_id, start, err_stat, n_ind, list_d, n_ind_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ind	Integer	Write	Reference	Number of defined view indexes.
list_d	Array of integers	Write	Descriptor	List of view index values.
n_ind_ret	Integer	Write	Reference	Number of view indexes returned.

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

INQUIRE LOCATOR DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_locator_state (ws_id, dev_num, value_type, max_rec_len, err_stat, op_mode, echo_flag, trans_num, world_loc, pet, echo_area, ret_buf_len, data_rec, tot_rec_len)`

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier.
<code>dev_num</code>	Integer	Read	Reference	Locator device number.
<code>value_type</code>	Integer (constant)	Read	Reference	Type of returned values.
<code>max_rec_len</code>	Integer	Read	Reference	Maximum number of bytes to put in locator data record.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>op_mode</code>	Integer (constant)	Write	Reference	Operating mode.
<code>echo_flag</code>	Integer (constant)	Write	Reference	Echo flag.
<code>trans_num</code>	Integer	Write	Reference	Transformation number.
<code>world_loc</code>	Array of 2 reals	Write	Reference	Initial locator position, in WC values.
<code>pet</code>	Integer	Write	Reference	Prompt and echo type.
<code>echo_area</code>	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
<code>ret_buf_len</code>	Integer	Modify	Reference	Length of returned record buffer. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
<code>data_rec</code>	Record	Write	Reference	Locator input data record. See the appropriate language-specific header file (<code>gks3d*defs.*</code>) for the specific data type and data record structure.
<code>tot_rec_len</code>	Integer	Write	Reference	Total size of the record, in bytes.

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the locator device. See the language-dependent header file (gks3d*defs.*) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the locator input data record before calling the INQUIRE LOCATOR DEVICE STATE function.

The information returned by this function depends on the locator input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (locator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the locator input data record.

See Also

INITIALIZE LOCATOR
 SET LOCATOR MODE
 SIZEOF

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

`gks3d$inq_locator_state3` (ws_id, dev_num, value_type, in_rec_len, err_stat, op_mode, echo_flag, trans_num, view_ind, world_loc, pet, echo_vol, ret_buf_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Locator device number.
value_type	Integer (constant)	Read	Reference	Type of returned value.
in_rec_len	Integer	Read	Reference	Maximum number of bytes to put in locator data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
trans_num	Integer	Write	Reference	Transformation number.
view_ind	Integer	Write	Reference	View index.
world_loc	Array of 3 reals	Write	Reference	Initial locator position, in WC values.
pet	Integer	Write	Reference	Prompt and echo type.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
ret_buf_len	Integer	Modify	Reference	Size of the returned data record, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Locator input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the record, in bytes.

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the locator device. See the language-dependent header file (gks3d*defs.*) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the locator input data record before calling the INQUIRE LOCATOR DEVICE STATE 3 function.

The information returned by this function depends on the locator input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (locator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the locator input data record.

See Also

INITIALIZE LOCATOR 3
 SET LOCATOR MODE
 SIZEOF

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

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_max_ws_state_table (ws_type, err_stat, max_pline, max_pmark,  
                               max_text, max_fill_area, max_pattern, max_colour)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
max_pline	Integer	Write	Reference	Maximum number of polyline bundle table entries
max_pmark	Integer	Write	Reference	Maximum number of polymarker bundle table entries
max_text	Integer	Write	Reference	Maximum number of text bundle table entries
max_fill_area	Integer	Write	Reference	Maximum number of fill area bundle table entries
max_pattern	Integer	Write	Reference	Maximum number of pattern indexes
max_colour	Integer	Write	Reference	Maximum number of color indexes

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

gks3d\$inq_max_ws_state_table3 (ws_type, err_stat, max_pline, max_pmark, max_text, max_fill_area, max_pattern, max_colour, max_view, max_edge)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
max_pline	Integer	Write	Reference	Maximum number of polyline bundle table entries
max_pmark	Integer	Write	Reference	Maximum number of polymarker bundle table entries
max_text	Integer	Write	Reference	Maximum number of text bundle table entries
max_fill_area	Integer	Write	Reference	Maximum number of fill area bundle table entries
max_pattern	Integer	Write	Reference	Maximum number of pattern indexes
max_colour	Integer	Write	Reference	Maximum number of color indexes
max_view	Integer	Write	Reference	Maximum number of view bundle table entries
max_edge	Integer	Write	Reference	Maximum number of edge bundle table entries

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

`gks3d$inq_max_xform (err_stat, max_trans)`

Argument	Data Type	Access	Passed by	Description
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>max_trans</code>	Integer	Write	Reference	Maximum normalization transformation number supported. Window and viewport boundaries can be associated with transformation numbers from 1 to <i>max_trans</i> .

Description

The INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER function returns the maximum normalization transformation number supported by the GKS implementation being used. The maximum number of normalization transformations for the DEC GKS software is 256 (numbered 0 to 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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_more_simul_events (err_stat, more_events_flag)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
more_events_flag	Integer (constant)	Write	Reference	More simultaneous events flag

Constants

Defined Argument	Constant	Description
more_events_flag	GKS3D\$K_NOMORE_EVENTS	No more events
	GKS3D\$K_MORE_EVENTS	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

INQUIRE NAME OF OPEN SEGMENT

Operating States

SGOP

Syntax

```
gks3d$inq_name_open_seg (err_stat, seg_id)
```

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator
seg_id	Integer	Write	Reference	Segment name

Description

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

See Also

CREATE SEGMENT

INQUIRE NORMALIZATION TRANSFORMATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_xform (trans_num, err_stat, window_b, viewport_b)

Argument	Data Type	Access	Passed by	Description
trans_num	Integer	Read	Reference	Normalization transformation number
err_stat	Integer	Write	Reference	Error indicator
window_b	Array of 4 reals	Write	Reference	WC window limits, in the order XMIN, XMAX, YMIN, YMAX
viewport_b	Array of 4 reals	Write	Reference	NDC viewport limits, in the order XMIN, XMAX, YMIN, YMAX

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 256 (numbered 0 to 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

INQUIRE NORMALIZATION TRANSFORMATION 3

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_xform3 (trans_num, err_stat, window_b, viewport_b)

Argument	Data Type	Access	Passed by	Description
trans_num	Integer	Read	Reference	Normalization transformation number
err_stat	Integer	Write	Reference	Error indicator
window_b	Array of 6 reals	Write	Reference	WC window limits, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
viewport_b	Array of 6 reals	Write	Reference	NDC viewport limits, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX

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 256 (numbered 0 to 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

```
gks3d$inq_input_dev (ws_type, err_stat, n_loc_dev, n_stroke_dev, n_val_dev,  
n_choice_dev, n_pick_dev, n_str_dev)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
n_loc_dev	Integer	Write	Reference	Number of locator devices
n_stroke_ dev	Integer	Write	Reference	Number of stroke devices
n_val_dev	Integer	Write	Reference	Number of valuator devices
n_choice_ dev	Integer	Write	Reference	Number of choice devices
n_pick_dev	Integer	Write	Reference	Number of pick devices
n_str_dev	Integer	Write	Reference	Number of string devices

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

`gks3d$inq_seg_priority (ws_type, err_stat, n_priorities)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>n_priorities</code>	Integer	Write	Reference	Number of segment priorities supported. If this function writes the value 0 to this argument, the device supports an infinite number of priorities.

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

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_operating_state (op_state)

Argument	Data Type	Access	Passed by	Description
op_state	Integer (constant)	Write	Reference	Operating state

Constants

Defined Argument	Constant	Description
op_state	GKS3D\$K_GKCL	GKS is closed.
	GKS3D\$K_GKOP	GKS is open.
	GKS3D\$K_WSOP	At least one workstation is open.
	GKS3D\$K_WSAC	At least one workstation is active.
	GKS3D\$K_SGOP	A segment is being created.

Description

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

See Also

OPEN GKS

INQUIRE PATTERN FACILITIES

INQUIRE PATTERN FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pat_fac (ws_type, err_stat, n_patt_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
n_patt_ind	Integer	Write	Reference	Number of pattern indexes supported

Description

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

INQUIRE PATTERN REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pat_rep (ws_id, patt_ind, value_type, err_stat, patt_width, patt_height, pattern_d, c_cols_size, c_rows_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
patt_ind	Integer	Read	Reference	Pattern index
value_type	Integer (constant)	Read	Reference	Type of returned values
err_stat	Integer	Write	Reference	Error indicator
patt_width, patt_height	Integer	Write	Reference	Pattern width and height
pattern_d	Array of integers	Write	Descriptor	Pattern array
c_cols_size, c_rows_size	Integer	Write	Reference	Number of columns and rows written to the pattern array

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	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 PICK DEVICE STATE

INQUIRE PICK DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_pick_state (ws_id, dev_num, value_type, max_rec_len, err_stat, op_mode, echo_flag, pick_status, seg_id, pick_id, pet, echo_area, ret_buf_len, data_rec, tot_rec_len)`

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier.
<code>dev_num</code>	Integer	Read	Reference	Pick device number.
<code>value_type</code>	Integer (constant)	Read	Reference	Type of returned values.
<code>max_rec_len</code>	Integer	Read	Reference	Maximum number of bytes to put in pick data record.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>op_mode</code>	Integer (constant)	Write	Reference	Operating mode.
<code>echo_flag</code>	Integer (constant)	Write	Reference	Echo flag.
<code>pick_status</code>	Integer (constant)	Write	Reference	Initial pick status.
<code>seg_id</code>	Integer	Write	Reference	Segment name.
<code>pick_id</code>	Integer	Write	Reference	Initial pick identifier.
<code>pet</code>	Integer	Write	Reference	Prompt and echo type.
<code>echo_area</code>	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
<code>ret_buf_len</code>	Integer	Modify	Reference	Length of the returned record buffer. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
<code>data_rec</code>	Record	Write	Reference	Pick input data record. See the appropriate language-specific header file (<code>gks3d*defs.*</code>) for the specific data type and data record structure.
<code>tot_rec_len</code>	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.
pick_status	GKS3D\$K_STATUS_OK	Return the initial measure.
	GKS3D\$K_STATUS_NOPICK	Return no pick.

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the pick device. See the language-dependent header file (*gks3d*defs.**) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the pick input data record before calling the INQUIRE PICK DEVICE STATE function.

The information returned by this function depends on the pick input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (pick input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the pick input data record.

See Also

INITIALIZE PICK
 SET PICK MODE
 SIZEOF

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

INQUIRE PICK DEVICE STATE 3

INQUIRE PICK DEVICE STATE 3

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_pick_state3` (`ws_id`, `dev_num`, `value_type`, `max_rec_len`, `err_stat`, `op_mode`, `echo_flag`, `pick_status`, `seg_id`, `pick_id`, `pet`, `echo_vol`, `ret_buf_len`, `data_rec`, `tot_rec_len`)

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier.
<code>dev_num</code>	Integer	Read	Reference	Pick device number.
<code>value_type</code>	Integer (constant)	Read	Reference	Type of returned values.
<code>max_rec_len</code>	Integer	Read	Reference	Maximum number of bytes to put in pick data record.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>op_mode</code>	Integer (constant)	Write	Reference	Operating mode.
<code>echo_flag</code>	Integer (constant)	Write	Reference	Echo flag.
<code>pick_status</code>	Integer (constant)	Write	Reference	Initial pick status.
<code>seg_id</code>	Integer	Write	Reference	Segment name.
<code>pick_id</code>	Integer	Write	Reference	Initial pick identifier.
<code>pet</code>	Integer	Write	Reference	Prompt and echo type.
<code>echo_vol</code>	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
<code>ret_buf_len</code>	Integer	Modify	Reference	Length of the returned record buffer. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
<code>data_rec</code>	Record	Write	Reference	Pick input data record. See the appropriate language-specific header file (<code>gks3d*defs.*</code>) for the specific data type and data record structure.
<code>tot_rec_len</code>	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.
pick_status	GKS3D\$K_STATUS_OK	Return the initial measure.
	GKS3D\$K_STATUS_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 data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the pick device. See the language-dependent header file (*gks3d*defs.**) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the pick input data record before calling the INQUIRE PICK DEVICE STATE 3 function.

The information returned by this function depends on the pick input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (pick input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the pick input data record.

See Also

INITIALIZE PICK 3
 SET PICK MODE
 SIZEOF

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

INQUIRE PIXEL

INQUIRE PIXEL

Operating States

WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_pixel (ws_id, point, err_stat, col_ind)
```

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
point	Array of 2 reals	Read	Reference	WC point
err_stat	Integer	Write	Reference	Error indicator
col_ind	Integer	Write	Reference	Color index

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

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_pixel_array (ws_id, x_start, y_start, num_cols, num_rows, corner_point, err_stat, inv_ind_flag, colour_d)`

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier.
<code>x_start</code> , <code>y_start</code>	Integer	Read	Reference	Start column and row in 2D color index array.
<code>num_cols</code> , <code>num_rows</code>	Integer	Read	Reference	Dimensions of color index array. The values must be less than or equal to the size of the buffer, from <code>x_start</code> and <code>y_start</code> to the last element of <code>colour_d</code> .
<code>corner_point</code>	Array of 2 reals	Read	Reference	Upper left corner of the rectangle.
<code>err_stat</code>	Integer	Write	Reference	Error indicator.
<code>inv_ind_flag</code>	Integer (constant)	Write	Reference	Flag for invalid color index values.
<code>colour_d</code>	Array of integers	Write	Descriptor	2D color index array.

Constants

Defined Argument	Constant	Description
<code>inv_ind_flag</code>	<code>GKS3D\$K_INVALID_ABSENT</code>	Color array contains no invalid indexes.
	<code>GKS3D\$K_INVALID_PRESENT</code>	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.

INQUIRE PIXEL ARRAY

See Also

SET COLOUR REPRESENTATION

INQUIRE PIXEL ARRAY DIMENSIONS

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pixel_array_dim (ws_id, p, q, err_stat, dim_x, dim_y)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
p, q	Array of reals	Read	Reference	Points describing the rectangle (the upper left and lower right corners)
err_stat	Integer	Write	Reference	Error indicator
dim_x, dim_y	Integer	Write	Reference	Dimensions of pixel array

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 FACILITIES

INQUIRE POLYLINE FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pline_fac (ws_type, start, err_stat, n_line_types, line_types, n_line_widths, nom_line_width, line_width_min, line_width_max, n_indexes, n_line_types_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
start	Integer	Write	Reference	Offset into the list Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_line_types	Integer	Write	Reference	Number of line types.
line_types	Array of integers (constant)	Write	Descriptor	Available line types.
n_line_widths	Integer	Write	Reference	Number of line widths.
nom_line_width	Real	Write	Reference	Nominal line width.
line_width_min, line_width_max	Real	Write	Reference	Minimum and maximum line widths.
n_indexes	Integer	Write	Reference	Number of predefined polyline indexes.
n_line_types_ret	Integer	Write	Reference	Number of line types returned.

Constants

Defined Argument	Constant	Description
line_types	GKS3D\$K_LINETYPE_SOLID	Solid line
	GKS3D\$K_LINETYPE_DASHED	Dashed line
	GKS3D\$K_LINETYPE_DOTTED	Dotted line
	GKS3D\$K_LINETYPE_DASHED_DOTTED	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 REPRESENTATION

INQUIRE POLYLINE REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pline_rep (ws_id, pline_ind, value_type, err_stat, line_type, line_width_sf, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
pline_ind	Integer	Read	Reference	Polyline index
value_type	Integer (constant)	Read	Reference	Type of returned values
err_stat	Integer	Write	Reference	Error indicator
line_type	Integer (constant)	Write	Reference	Line type
line_width_sf	Real	Write	Reference	Line width scale factor
col_ind	Integer	Write	Reference	Color index value

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
line_type	GKS3D\$K_LINETYPE_SOLID	Solid line.
	GKS3D\$K_LINETYPE_DASHED	Dashed line.
	GKS3D\$K_LINETYPE_DOTTED	Dotted line.
	GKS3D\$K_LINETYPE_DASHED_DOTTED	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.

INQUIRE POLYLINE REPRESENTATION

See Also

SET POLYLINE REPRESENTATION

INQUIRE POLYMARKER FACILITIES

INQUIRE POLYMARKER FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pmark_fac (ws_type, start, err_start, n_mark_types, mark_types,
n_mark_sizes, nom_mark_size, mark_size_min,
mark_size_max, n_mark_indices, n_mark_type_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_mark_types	Integer	Write	Reference	Number of marker types.
mark_types	Array of integers (constant)	Write	Descriptor	List of marker types.
n_mark_sizes	Integer	Write	Reference	Number of marker sizes.
nom_mark_size	Real	Write	Reference	Nominal marker size.
mark_size_min, mark_size_max	Real	Write	Reference	Minimum and maximum marker sizes.
n_mark_indices	Integer	Write	Reference	Number of predefined marker indexes.
n_mark_type_ret	Integer	Write	Reference	Number of marker types returned in list.

Constants

Defined Argument	Constant	Description
mark_types	GKS3D\$K_MARKTYPE_DOT	Dot
	GKS3D\$K_MARKTYPE_PLUS	Plus sign
	GKS3D\$K_MARKTYPE_ASTERISK	Asterisk
	GKS3D\$K_MARKTYPE_CIRCLE	Circle
	GKS3D\$K_MARKTYPE_DIAGONAL_CROSS	Diagonal cross

INQUIRE POLYMARKER FACILITIES

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 REPRESENTATION

INQUIRE POLYMARKER REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_pmark_rep (ws_id, pmark_ind, value_type, err_stat, marker_type, marker_size_sf, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
pmark_ind	Integer	Read	Reference	Polymarker index
value_type	Integer (constant)	Read	Reference	Type of returned values
err_stat	Integer	Write	Reference	Error indicator
marker_type	Integer (constant)	Write	Reference	Polymarker type
marker_size_sf	Real	Write	Reference	Marker size scale factor
col_ind	Integer	Write	Reference	Polymarker color index

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
marker_type	GKS3D\$K_MARKTYPE_DOT	Dot.
	GKS3D\$K_MARKTYPE_PLUS	Plus sign.
	GKS3D\$K_MARKTYPE_ASTERISK	Asterisk.
	GKS3D\$K_MARKTYPE_CIRCLE	Circle.
	GKS3D\$K_MARKTYPE_DIAGONAL_CROSS	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.

INQUIRE POLYMARKER REPRESENTATION

See Also

SET POLYMARKER REPRESENTATION

INQUIRE PREDEFINED COLOUR REPRESENTATION

INQUIRE PREDEFINED COLOUR REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_predef_colour_rep (ws_type, col_ind, err_stat, comp1, comp2, comp3)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
col_ind	Integer	Read	Reference	Color index
err_stat	Integer	Write	Reference	Error indicator
comp1, comp2, comp3	Real	Write	Reference	Color components

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_predef_edge_rep (ws_type, edge_ind, err_stat, edge_flag, edge_type, edgewidth_sf, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
edge_ind	Integer	Read	Reference	Predefined edge index
err_stat	Integer	Write	Reference	Error indicator
edge_flag	Integer (constant)	Write	Reference	Edge flag
edge_type	Integer (constant)	Write	Reference	Edge type
edgewidth_sf	Integer	Write	Reference	Edge width scale factor
col_ind	Integer	Write	Reference	Edge color index

Constants

Defined Argument	Constant	Description
edge_flag	GKS3D\$K_EDGE_OFF	Edge off
	GKS3D\$K_EDGE_ON	Edge on
edge_type	GKS3D\$K_EDGE_SOLID	Solid edge
	GKS3D\$K_EDGE_DASHED	Dashed edge
	GKS3D\$K_EDGE_DOTTED	Dotted edge
	GKS3D\$K_EDGE_DASHED_DOTTED	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

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

`gks3d$inq_predef_fill_rep (ws_type, fill_ind, err_stat, int_style, style_ind, col_ind)`

Argument	Data Type	Access	Passed by	Description
<code>ws_type</code>	Integer	Read	Reference	Workstation type
<code>fill_ind</code>	Integer	Read	Reference	Fill area index
<code>err_stat</code>	Integer	Write	Reference	Error indicator
<code>int_style</code>	Integer (constant)	Write	Reference	Fill area interior style
<code>style_ind</code>	Integer	Write	Reference	Fill area style index
<code>col_ind</code>	Integer	Write	Reference	Fill area color index

Constants

Defined Argument	Constant	Description
<code>int_style</code>	<code>GKS3D\$K_INTSTYLE_HOLLOW</code>	Hollow interior
	<code>GKS3D\$K_INTSTYLE_SOLID</code>	Solid interior
	<code>GKS3D\$K_INTSTYLE_PATTERN</code>	Patterned interior
	<code>GKS3D\$K_INTSTYLE_HATCH</code>	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

```
gks3d$inq_predef_pat_rep (ws_type, patt_ind, err_stat, height, width, col_array,  
n_rows, n_columns)
```

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
patt_ind	Integer	Read	Reference	Pattern index.
err_stat	Integer	Write	Reference	Error indicator.
height, width	Integer	Write	Reference	Number of rows (height) and columns (width) in the pattern.
col_array	Array of integers	Write	Descriptor	Color pattern array.
n_rows, n_columns	Integer	Write	Reference	Number of returned columns and rows in pattern. These values can be used to traverse only the elements of the array that contain valid color index values.

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

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_predef_pline_rep (ws_type, pline_ind, err_stat, line_type, line_width_sf, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
pline_ind	Integer	Read	Reference	Polyline indicator
err_stat	Integer	Write	Reference	Error indicator
line_type	Integer (constant)	Write	Reference	Line type
line_width_sf	Real	Write	Reference	Line width scale factor
col_ind	Integer	Write	Reference	Color index value

Constants

Defined Argument	Constant	Description
line_type	GKS3D\$K_LINETYPE_SOLID	Solid line
	GKS3D\$K_LINETYPE_DASHED	Dashed line
	GKS3D\$K_LINETYPE_DOTTED	Dotted line
	GKS3D\$K_LINETYPE_DASHED_DOTTED	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

gks3d\$inq_predef_pmark_rep (ws_type, pmark_ind, err_stat, mark_type,
mark_size_sf, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
pmark_ind	Integer	Read	Reference	Polymarker index
err_stat	Integer	Write	Reference	Error indicator
mark_type	Integer (constant)	Write	Reference	Polymarker type
mark_size_ sf	Real	Write	Reference	Marker size scale factor
col_ind	Integer	Write	Reference	Polymarker color index

Constants

Defined Argument	Constant	Description
mark_type	GKS3D\$K_MARKTYPE_DOT	Dot
	GKS3D\$K_MARKTYPE_PLUS	Plus sign
	GKS3D\$K_MARKTYPE_ASTERISK	Asterisk
	GKS3D\$K_MARKTYPE_CIRCLE	Circle
	GKS3D\$K_MARKTYPE_ DIAGONAL_CROSS	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

INQUIRE PREDEFINED TEXT REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_predef_text_rep (ws_type, text_ind, err_stat, font_num, precision, char_exp_factor, char_spacing, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
text_ind	Integer	Read	Reference	Text index
err_stat	Integer	Write	Reference	Error indicator
font_num	Integer	Write	Reference	Text font number
precision	Integer (constant)	Write	Reference	Text precision
char_exp_factor	Real	Write	Reference	Character expansion factor
char_spacing	Real	Write	Reference	Character spacing
col_ind	Integer	Write	Reference	Text color index

Constants

Defined Argument	Constant	Description
precision	GKS3D\$K_TEXT_PRECISION_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS3D\$K_TEXT_PRECISION_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GKS3D\$K_TEXT_PRECISION_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

gks3d\$inq_predef_view_rep (ws_type, view_ind, err_stat, orient_matrix, map_matrix, clip_vol, xy_clip, back_clip, front_clip)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
view_ind	Integer	Read	Reference	View index
err_stat	Integer	Write	Reference	Error indicator
orient_matrix	Array of reals	Write	Reference	View orientation matrix
map_matrix	Array of reals	Write	Reference	View mapping matrix
clip_vol	Array of reals	Write	Reference	View clipping limits
xy_clip	Integer (constant)	Write	Reference	XY clipping indicator
back_clip	Integer	Write	Reference	Back clipping indicator
front_clip	Integer	Write	Reference	Front clipping indicator

Constants

Defined Argument	Constant	Description
xy_clip	GKS3D\$K_NOCLIP	Clipping disabled
	GKS3D\$K_CLIP	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

INQUIRE SEGMENT ATTRIBUTES

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_seg_attn (seg_id, err_stat, trans_matrix, vis_flag, highl_flag, priority, detect_flag)

Argument	Data Type	Access	Passed by	Description
seg_id	Integer	Read	Reference	Segment identifier.
err_stat	Integer	Write	Reference	Error indicator.
trans_matrix	2×3 array of reals	Write	Reference	Segment transformation matrix.
vis_flag	Integer (constant)	Write	Reference	Visibility flag.
highl_flag	Integer (constant)	Write	Reference	Highlighting flag.
priority	Real	Write	Reference	Segment priority. Segment priorities range from 0.0 to 1.0, and each device supports a finite number of priorities.
detect_flag	Integer (constant)	Write	Reference	Detectability flag.

Constants

Defined Argument	Constant	Description
vis_flag	GKS3D\$K_INVISIBLE	DEC GKS does not show the segment on the workstation surface.
	GKS3D\$K_VISIBLE	DEC GKS shows the segment on the workstation surface.
highl_flag	GKS3D\$K_NORMAL	DEC GKS does not highlight the segment.
	GKS3D\$K_HIGHLIGHTED	DEC GKS highlights the segment, if visible.
detect_flag	GKS3D\$K_UNDETECTABLE	Segment cannot be picked.
	GKS3D\$K_DETECTABLE	Segment, if visible, can be picked.

INQUIRE SEGMENT ATTRIBUTES

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.

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

INQUIRE SEGMENT ATTRIBUTES 3

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_seg_attn3 (seg_id, err_stat, trans_matrix, vis_flag, highl_flag, priority, detect_flag)

Argument	Data Type	Access	Passed by	Description
seg_id	Integer	Read	Reference	Segment identifier.
err_stat	Integer	Write	Reference	Error indicator.
trans_matrix	3×4 array of reals	Write	Reference	Segment transformation matrix.
vis_flag	Integer (constant)	Write	Reference	Visibility flag.
highl_flag	Integer (constant)	Write	Reference	Highlighting flag.
priority	Real	Write	Reference	Segment priority. Segment priorities range from 0.0 to 1.0, and each device supports a finite number of priorities.
detect_flag	Integer (constant)	Write	Reference	Detectability flag.

Constants

Defined Argument	Constant	Description
vis_flag	GKS3D\$K_INVISIBLE	DEC GKS does not show the segment on the workstation surface.
	GKS3D\$K_VISIBLE	DEC GKS shows the segment on the workstation surface.
highl_flag	GKS3D\$K_NORMAL	DEC GKS does not highlight the segment.
	GKS3D\$K_HIGHLIGHTED	DEC GKS highlights the segment, if visible.
detect_flag	GKS3D\$K_UNDETECTABLE	Segment cannot be picked.
	GKS3D\$K_DETECTABLE	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.

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

INQUIRE SET OF ACTIVE WORKSTATIONS

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_active_ws (start, err_stat, n_ws, wslst_d, n_ws_ret)

Argument	Data Type	Access	Passed by	Description
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ws	Integer	Write	Reference	Number of active workstations.
wslst_d	Array of integers	Write	Descriptor	Identifiers of currently active workstations.
n_ws_ret	Integer	Write	Reference	Number of workstation identifiers returned in active workstation list. Compare this argument with the argument <i>n_ws</i> to check that the defined array is large enough to hold all the returned values.

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

gks3d\$inq_set_assoc_ws (seg_id, start, err_stat, req_n_ws, wslst_d, ret_n_ws)

Argument	Data Type	Access	Passed by	Description
seg_id	Integer	Read	Reference	Segment identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
req_n_ws	Integer	Write	Reference	Required size for list of workstations.
wslst_d	Array of integers	Write	Descriptor	List of associated workstation identifiers.
ret_n_ws	Integer	Write	Reference	Size of the returned workstation list. Compare this argument with the argument <i>req_n_ws</i> to check that the defined array is large enough to hold all the returned values.

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

INQUIRE SET OF OPEN WORKSTATIONS

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_open_ws (start, err_stat, n_ws, ws_list, n_ws_ret)

Argument	Data Type	Access	Passed by	Description
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_ws	Integer	Write	Reference	Number of open workstations.
ws_list	Array of integers	Write	Reference	Identifiers of currently active workstations.
n_ws_ret	Integer	Write	Reference	Number of workstation identifiers returned in the open workstation list. Compare this argument with the argument <i>n_ws</i> to check that the defined array is large enough to hold all the returned values.

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

gks3d\$inq_seg_names (start, err_stat, n_segs, list_segs_d, n_segs_ret)

Argument	Data Type	Access	Passed by	Description
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_segs	Integer	Write	Reference	Total number of existing segments.
list_segs_d	Array of integers	Write	Descriptor	List of existing segments.
n_segs_ret	Integer	Read	Reference	Number of segment names returned.

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

```
gks3d$inq_seg_names_on_ws (ws_id, start, err_stat, n_seg_names, seg_names_d,  
                           n_seg_ret)
```

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
start	Integer	Read	Reference	Offset into the list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_seg_names	Integer	Write	Reference	Number of names in list.
seg_names_d	Array of integers	Write	Descriptor	List of stored segments.
n_seg_ret	Integer	Write	Reference	Returned size of list. Compare this argument with the argument <i>n_seg_names</i> to check that the defined array is large enough to hold all the returned values.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_string_state (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, ret_str_len, string_d, tot_str_len, pet, echo_area, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	String device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in string data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
ret_str_len	Integer	Write	Reference	Returned string length.
string_d	String descriptor	Write	Descriptor	Initial string.
tot_str_len	Integer	Write	Reference	Total string length.
pet	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned data record, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	String input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

INQUIRE STRING DEVICE STATE

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the string device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the string input data record before calling the INQUIRE STRING DEVICE STATE function.

The information returned by this function depends on the string input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (string input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the string input data record.

See Also

INITIALIZE STRING
SET STRING MODE
SIZEOF

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

INQUIRE STRING DEVICE STATE 3

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_string_state3 (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, ret_str_len, string_d, tot_str_len, pet, echo_vol, ret_buf_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	String device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in string data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
ret_str_len	Integer	Write	Reference	Returned string length.
string_d	String descriptor	Write	Descriptor	Initial string.
tot_str_len	Integer	Write	Reference	Total string length.
pet	Integer	Write	Reference	Prompt and echo type.
echo_vol	Array of 6 reals	Write	Reference	Default echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
ret_buf_len	Integer	Modify	Reference	Length of the returned data record buffer. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	String input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

INQUIRE STRING DEVICE STATE 3

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the string device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the `SIZEOF` utility function to determine the size of the string input data record before calling the INQUIRE STRING DEVICE STATE 3 function.

The information returned by this function depends on the string input record size (the *max_rec_len* argument). If the data record size is equal to `SIZEOF` (string input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the string input data record.

See Also

INITIALIZE STRING 3
SET STRING MODE
SIZEOF

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

INQUIRE STROKE DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_stroke_state (ws_id, dev_num, value_type, max_points, max_rec_len, err_stat, op_mode, echo_flag, trans_num, n_points, stroke_x, stroke_y, tot_pts, pet, echo_area, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Stroke device number.
value_type	Integer (constant)	Read	Reference	Type of returned values.
max_points	Integer	Read	Reference	Maximum stroke buffer size.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in the stroke data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
trans_num	Integer	Write	Reference	Transformation number.
n_points	Integer	Write	Reference	Number of points in stroke returned.
stroke_x, stroke_y	Array of reals	Write	Reference	Initial stroke positions. The points are stored in the order X ₁ , Y ₁ , X ₂ , Y ₂ , X ₃ , Y ₃ and so on, for each of the <i>n_points</i> .
tot_pts	Integer	Write	Reference	Total number of points in the stroke.
pet	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Stroke input data record. See the appropriate language-dependent header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

INQUIRE STROKE DEVICE STATE

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the stroke device. See the language-dependent header file (*gks3d*defs.**) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the stroke input data record before calling the INQUIRE STROKE DEVICE STATE function.

The information returned by this function depends on the stroke input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (stroke input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the stroke input data record.

See Also

INITIALIZE STROKE
SET STROKE MODE
SIZEOF

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

INQUIRE STROKE DEVICE STATE 3

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_stroke_state3 (ws_id, dev_num, value_type, max_points, max_rec_len, err_stat, op_mode, echo_flag, trans_num, view_ind, ret_points, stroke_x, stroke_y, stroke_z, tot_pts, pet, echo_vol, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Stroke device number.
value_type	Integer (constant)	Read	Reference	Type of returned values.
max_points	Integer	Read	Reference	Maximum number of points.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in stroke data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
trans_num	Integer	Write	Reference	Transformation number.
view_ind	Integer	Write	Reference	View index.
ret_pts	Integer	Write	Reference	Number of points.
stroke_x, stroke_y, stroke_z	Array of reals	Write	Reference	Initial stroke positions. The points are stored in the order X ₁ , Y ₁ , Z ₁ , X ₂ , Y ₂ , Z ₂ , X ₃ , Y ₃ , Z ₃ and so on, for each of the <i>ret_pts</i> .
tot_pts	Integer	Write	Reference	Total number of points in the stroke.
pet	Integer	Write	Reference	Prompt and echo type.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Stroke input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

INQUIRE STROKE DEVICE STATE 3

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode.
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS3D\$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS3D\$K_NOECHO	Echo disabled.
	GKS3D\$K_ECHO	Echo enabled.

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the stroke device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the `SIZEOF` utility function to determine the size of the stroke input data record before calling the INQUIRE STROKE DEVICE STATE 3 function.

The information returned by this function depends on the stroke input record size (the *max_rec_len* argument). If the data record size is equal to `SIZEOF` (stroke input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the stroke input data record.

See Also

INITIALIZE STROKE 3
SET STROKE MODE
SIZEOF

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

INQUIRE TEXT EXTENT
Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_text_extent (ws_id, pos, string_d, err_stat, conc_pt, extent)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
pos	Array of 2 reals	Read	Reference	Text position in WC values.
string_d	Character string	Read	Descriptor	Text string to output.
err_stat	Integer	Write	Reference	Error indicator.
conc_pt	Array of 2 reals	Write	Reference	Concatenation point in WC values.
extent	Array of 8 reals	Write	Reference	Four WC X and Y values that specify the text extent parallelogram. The parallelogram is returned as four points in counterclockwise order. The extent parallelogram encloses the character bodies of the specified string.

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

```
gks3d$inq_text_extent3 (ws_id, pos, dir_vec1, dir_vec2, string_d, err_stat, conc_pt, extent)
```

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
pos	Array of 3 reals	Read	Reference	Text position in WC values.
dir_vec1, dir_vec2	Array of 3 reals	Read	Reference	Two 3D vectors describing text plane.
string_d	Character string	Read	Descriptor	Text string to output.
err_stat	Integer	Write	Reference	Error indicator.
conc_pt	Array of 3 reals	Write	Reference	Concatenation point in WC values.
extent	Array of 12 reals	Write	Reference	Four WC X, Y, and Z values that specify the text extent parallelogram. The parallelogram is returned as four points in counterclockwise order. The extent parallelogram encloses the character bodies of the specified string.

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

gks3d\$inq_text_fac (ws_type, start, err_stat, n_font_prec, font_prec, n_heights, height_min, height_max, n_char_exp, char_exp_min, char_exp_max, n_indexes, n_font_prec_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
start	Integer	Read	Reference	Offset into list. Set <i>start</i> to 0 to inquire from the beginning of the list.
err_stat	Integer	Write	Reference	Error indicator.
n_font_prec	Integer	Write	Reference	Number of font-precision pairs.
font_prec	Array of integers (constant)	Write	Descriptor	List of font and precision pairs.
n_heights	Integer	Write	Reference	Number of character heights.
height_min, height_max	Real	Write	Reference	Minimum and maximum character heights.
n_char_exp	Integer	Write	Reference	Number of character expansion factors.
char_exp_min, char_exp_max	Real	Write	Reference	Minimum and maximum character expansion factors.
n_indexes	Integer	Write	Reference	Number of predefined text indexes.
n_font_prec_ret	Integer	Write	Reference	Returned size of font and precision list. Compare this to <i>n_font_prec</i> to determine if the defined array is large enough to hold all the returned values.

Constants

Defined Argument	Constant	Description
font_prec	GKS3D\$K_TEXT_PRECISION_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS3D\$K_TEXT_PRECISION_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.

INQUIRE TEXT FACILITIES

Defined Argument	Constant	Description
	GKS3D\$K_TEXT_PRECISION_ 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.

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 REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_text_rep (ws_id, text_index, value_type, err_stat, text_font, precision, char_exp, char_space, col_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
text_index	Integer	Read	Reference	Text index.
value_type	Integer (constant)	Read	Reference	Type of returned values.
err_stat	Integer	Write	Reference	Error indicator.
text_font	Integer	Write	Reference	Text font number.
precision	Integer (constant)	Write	Reference	Text precision.
char_exp	Real	Write	Reference	Character expansion factor. This does not affect the height of the characters.
char_space	Real	Write	Reference	Character spacing. Positive values increase the space between characters; negative values decrease the space. The value 0 places the character bodies adjacent to one another.
col_ind	Integer	Write	Reference	Color index value.

Constants

Defined Argument	Constant	Description
value_type	GKS3D\$K_VALUE_SET	Use the exact state list values.
	GKS3D\$K_VALUE_REALIZED	Use the values approximated by the graphics handler.
precision	GKS3D\$K_TEXT_PREC_STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS3D\$K_TEXT_PREC_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.

INQUIRE TEXT REPRESENTATION

Defined Argument	Constant	Description
	GKS3D\$K_TEXT_PREC_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

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_valuator_state (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, init_value, pet, echo_area, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Valuator device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in valuator data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_value	Integer	Write	Reference	Initial valuator value.
pet	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of 4 reals	Write	Reference	Echo area, in the order XMIN, XMAX, YMIN, YMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Valuator input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of data record, in bytes.

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode

INQUIRE VALUATOR DEVICE STATE

Defined Argument	Constant	Description
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the valuator device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the valuator input data record before calling the INQUIRE VALUATOR DEVICE STATE function.

The information returned by this function depends on the valuator input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (valuator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the valuator input data record.

See Also

INITIALIZE VALUATOR
SET VALUATOR MODE
SIZEOF

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

INQUIRE VALUATOR DEVICE STATE 3

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_valuator_state3 (ws_id, dev_num, max_rec_len, err_stat, op_mode, echo_flag, init_value, pet, echo_vol, ret_rec_len, data_rec, tot_rec_len)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Valuator device number.
max_rec_len	Integer	Read	Reference	Maximum number of bytes to put in valuator data record.
err_stat	Integer	Write	Reference	Error indicator.
op_mode	Integer (constant)	Write	Reference	Operating mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_value	Real	Write	Reference	Initial valuator value.
pet	Integer	Write	Reference	Prompt and echo type.
echo_vol	Array of 6 reals	Write	Reference	Echo volume, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX.
ret_rec_len	Integer	Modify	Reference	Length of the returned record buffer, in bytes. Compare this value to <i>tot_rec_len</i> to see if the entire data record was returned.
data_rec	Record	Write	Reference	Valuator input data record. See the appropriate language-specific header file (gks3d*defs.*) for the specific data type and data record structure.
tot_rec_len	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

Defined Argument	Constant	Description
op_mode	GKS3D\$K_INPUT_MODE_REQUEST	Request mode
	GKS3D\$K_INPUT_MODE_SAMPLE	Sample mode
	GKS3D\$K_INPUT_MODE_EVENT	Event mode

INQUIRE VALUATOR DEVICE STATE 3

Defined Argument	Constant	Description
echo_flag	GKS3D\$K_NOECHO	Echo disabled
	GKS3D\$K_ECHO	Echo enabled

Description

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

The data record returned by this function is the input data record associated with the PET returned in the argument *pet*. See the introduction to Chapter 9 for a description of the data record information for each of the possible PETs for the valuator device. See the language-dependent header file (`gks3d*defs.*`) for the specific data type and structure for the returned PET. Each of the data type names clearly corresponds to the device type and PET number. Use either a language-dependent operator or function, or the SIZEOF utility function to determine the size of the valuator input data record before calling the INQUIRE VALUATOR DEVICE STATE 3 function.

The information returned by this function depends on the valuator input record size (the *max_rec_len* argument). If the data record size is equal to SIZEOF (valuator input data record), DEC GKS returns all the information. If the data record size is 0, DEC GKS returns all the information except the valuator input data record.

See Also

INITIALIZE VALUATOR 3
SET VALUATOR MODE
SIZEOF

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

INQUIRE VIEW FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_view_fac (ws_type, err_stat, n_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
n_ind	Integer	Write	Reference	Number of view indexes

Description

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

INQUIRE VIEW REPRESENTATION 3

INQUIRE VIEW REPRESENTATION 3

Operating States

WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_view_rep3 (ws_id, view_ind, err_stat, update, req_orient_mat,  
                    cur_orient_mat, req_map_mat, cur_map_mat, req_clip_vol,  
                    cur_clip_vol, req_xy_clip, cur_xy_clip, req_back_clip,  
                    cur_back_clip, req_front_clip, cur_front_clip)
```

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
view_ind	Integer	Read	Reference	View index
err_stat	Integer	Write	Reference	Error indicator
update	Integer (constant)	Write	Reference	Viewing transformation update state
req_orient_mat	4×4 array of reals	Write	Reference	Requested view orientation matrix
cur_orient_mat	4×4 array of reals	Write	Reference	Current view orientation matrix
req_map_mat	4×4 array of reals	Write	Reference	Requested view mapping matrix
cur_map_mat	4×4 array of reals	Write	Reference	Current view mapping matrix
req_clip_vol	4×4 array of reals	Write	Reference	Requested view clipping limits
cur_clip_vol	Array of 6 reals	Write	Reference	Current view clipping limits
req_xy_clip	Integer (constant)	Write	Reference	Requested XY clipping indicator
cur_xy_clip	Integer (constant)	Write	Reference	Current XY clipping indicator
req_back_clip	Integer (constant)	Write	Reference	Requested back clipping indicator
cur_back_clip	Integer (constant)	Write	Reference	Current back clipping indicator
req_front_clip	Integer (constant)	Write	Reference	Requested front clipping indicator
cur_front_clip	Integer (constant)	Write	Reference	Current front clipping indicator

Constants

Defined Argument	Constant	Description
update	GKS3D\$K_NOTPENDING	Transformation is not pending
	GKS3D\$K_PENDING	Transformation is pending
req_xy_clip, cur_xy_clip, req_back_clip, cur_back_clip, req_front_clip, cur_front_clip	GKS3D\$K_NOCLIP GKS3D\$K_CLIP	Clipping disabled 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.

See Also

SET VIEW REPRESENTATION 3

INQUIRE WORKSTATION CATEGORY

INQUIRE WORKSTATION CATEGORY

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_ws_category (ws_type, err_stat, ws_cat)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
ws_cat	Integer (constant)	Write	Reference	Workstation category

Constants

Defined Argument	Constant	Description
ws_cat	GKS3D\$K_WSCAT_OUTPUT	Output category
	GKS3D\$K_WSCAT_INPUT	Input category
	GKS3D\$K_WSCAT_OUTIN	Output/input category
	GKS3D\$K_WSCAT_WISS	Workstation-independent segment storage category
	GKS3D\$K_WSCAT_MO	Metafile output category
	GKS3D\$K_WSCAT_MI	Metafile input category

Description

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

INQUIRE WORKSTATION CLASSIFICATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_ws_classification (ws_type, err_stat, ws_class)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
err_stat	Integer	Write	Reference	Error indicator
ws_class	Integer (constant)	Write	Reference	Workstation class

Constants

Defined Argument	Constant	Description
ws_class	GKS3D\$K_WSCLASS_VECTOR	Vector class
	GKS3D\$K_WSCLASS_RASTER	Raster class
	GKS3D\$K_WSCLASS_OTHER	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

INQUIRE WORKSTATION CONNECTION AND TYPE

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_ws_type (ws_id, err_stat, conid_d, ws_type, log_str_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
err_stat	Integer	Write	Reference	Error indicator.
conid_d	Character string	Write	Descriptor	Connection identifier. Only a portion of the logical name will be returned if <i>conid_d</i> is not large enough to hold the entire logical name.
ws_type	Integer	Write	Reference	Workstation type.
log_str_size	Integer	Write	Reference	Connection identifier length in bytes.

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

gks3d\$inq_ws_defer_and_update (ws_id, err_stat, defer_mode, regen_mode, surface_empty_flag, new_frame_flag)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
err_stat	Integer	Write	Reference	Error indicator
defer_mode	Integer (constant)	Write	Reference	Deferral mode
regen_mode	Integer (constant)	Write	Reference	Regeneration mode
surface_empt y_flag	Integer (constant)	Write	Reference	Display space empty flag
new_ frame_ flag	Integer (constant)	Write	Reference	New frame action necessary at update flag

Constants

Defined Argument	Constant	Description
defer_mode	GKS3D\$K_ASAP	Generate images as soon as possible.
	GKS3D\$K_BNIG	Generate images before input is requested globally.
	GKS3D\$K_BNIL	Generate images before input is requested locally.
	GKS3D\$K_ASTI	Generate images sometime. The exact time is not guaranteed.
regen_mode	GKS3D\$K_IRG_SUPPRESSED	Image regeneration is suppressed.
	GKS3D\$K_IRG_ALLOWED	Image regeneration is allowed.
surface_empt y_flag	GKS3D\$K_EMPTY	Display surface is empty.
	GKS3D\$K_NOTEMPTY	Display surface is not empty.

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

Defined Argument	Constant	Description
new_frame_ flag	GKS3D\$K_NEWFRAME _NOTNECESSARY	No new frame action on update.
	GKS3D\$K_NEWFRAME _NECESSARY	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.

See Also

SET DEFERRAL STATE

INQUIRE WORKSTATION MAXIMUM NUMBERS

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$inq_ws_max_num (err_stat, max_open_ws, max_active_ws, max_ws_seg)

Argument	Data Type	Access	Passed by	Description
err_stat	Integer	Write	Reference	Error indicator.
max_open_ws	Integer	Write	Reference	Maximum number of open workstations.
max_active_ws	Integer	Write	Reference	Maximum number of active workstations.
max_ws_seg	Integer	Write	Reference	Maximum number of workstations associated with segment.

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

INQUIRE WORKSTATION STATE

Operating States

WSOP, WSAC, SGOP

Syntax

`gks3d$inq_ws_state (ws_id, err_stat, ws_state)`

Argument	Data Type	Access	Passed by	Description
<code>ws_id</code>	Integer	Read	Reference	Workstation identifier
<code>err_stat</code>	Integer	Write	Reference	Error indicator
<code>ws_state</code>	Integer (constant)	Write	Reference	Workstation state

Constants

Defined Argument	Constant	Description
<code>ws_state</code>	<code>GKS3D\$K_WS_INACTIVE</code>	Workstation is not active.
	<code>GKS3D\$K_WS_ACTIVE</code>	Workstation is active.

Description

The INQUIRE WORKSTATION STATE function identifies the workstation state value as either `GKS3D$K_WS_INACTIVE` or `GKS3D$K_WS_ACTIVE`.

See Also

ACTIVATE WORKSTATION

INQUIRE WORKSTATION TRANSFORMATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks3d\$inq_ws_xform (ws_id, err_stat, update, req_window, cur_window,
req_viewport, cur_viewport)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
err_stat	Integer	Write	Reference	Error indicator
update	Integer (constant)	Write	Reference	Transformation update state
req_ window, cur_window	Array of 4 reals	Write	Reference	Requested and current workstation windows, in the order XMIN, XMAX, YMIN, YMAX
req_ viewport, cur_ viewport	Array of 4 reals	Write	Reference	Requested and current workstation viewports, in the order XMIN, XMAX, YMIN, YMAX

Constants

Defined Argument	Constant	Description
update	GKS3D\$K_NOTPENDING	Action not pending
	GKS3D\$K_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.

See Also

SET WORKSTATION VIEWPORT
SET WORKSTATION WINDOW

INQUIRE WORKSTATION TRANSFORMATION 3

INQUIRE WORKSTATION TRANSFORMATION 3

Operating States

WSOP, WSAC, SGOP

Syntax

```
gks3d$inq_ws_xform3 (ws_id, err_stat, update, req_window, cur_window,  
req_viewport, cur_viewport)
```

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
err_stat	Integer	Write	Reference	Error indicator
update	Integer (constant)	Write	Reference	Transformation update state
req_ window, cur_window	Array of 6 reals	Write	Reference	Requested and current workstation windows, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
req_ viewport, cur_ viewport	Array of 6 reals	Write	Reference	Requested and current workstation viewports, in the order XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX

Constants

Defined Argument	Constant	Description
update	GKS3D\$K_NOTPENDING	Action not pending.
	GKS3D\$K_PENDING	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 a function with three parameters corresponding to the values listed previously: the DEC GKS error number, the name of the DEC GKS function that generated the error, and the name of the error file. Pass the address of your error-handling function to SET ERROR HANDLER. For more information, see the SET ERROR HANDLER function in this chapter.

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

```
gks3d$emergency_close ( )
```

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

`gks3d$error_handler (err_num, func_name, err_file)`

Argument	Data Type	Access	Passed by	Description
<code>err_num</code>	Integer	Read	Reference	Error message number specified by the GKS standard or, in the case of negative error numbers, specified by the Digital implementation of GKS
<code>func_name</code>	Character string	Read	Descriptor	Text string containing the name of the function that detected the error
<code>err_file</code>	Character string	Read	Descriptor	Text string containing the name of the error logging file that was specified in a call to OPEN GKS

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. If you choose, you can write your own error handler to replace this function. See the introduction to Chapter 12 for more information.

See Also

EMERGENCY CLOSE GKS
ERROR LOGGING
OPEN GKS
SET ERROR HANDLER

ERROR LOGGING

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

gks3d\$log_error (err_num, func_name, err_file)

Argument	Data Type	Access	Passed by	Description
err_num	Integer	Read	Reference	Error message number specified by the GKS standard or, in the case of negative error numbers, specified by the Digital implementation of GKS
func_name	Character string	Read	Descriptor	Text string containing the name of the function that detected the error
err_file	Character string	Read	Descriptor	Text string containing the name of the error logging file that was specified in a call to OPEN GKS

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

SET ERROR HANDLER

SET ERROR HANDLER

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

```
gks3d$set_error_handler (new_handler, old_handler)
```

Argument	Data Type	Access	Passed by	Description
new_handler	Bound procedure value	Read	Reference	Name of error handler
old_handler	Bound procedure value	Read	Reference	Name of previous error handler

Description

The SET ERROR HANDLER function establishes an application-defined error handler as the function that DEC GKS calls upon error generation.

The error handler, whose address you pass to this function, replaces the error handler function supplied by DEC GKS (ERROR HANDLING).

Within your user-defined error handler, you can call only the DEC GKS functions EMERGENCY CLOSE GKS, ERROR HANDLING, ERROR LOGGING, or any of the inquiry functions.

See Also

Example 12–1 for a program example using the SET ERROR HANDLER function

12.2 Program Example

Example 12-1 illustrates the use of the error-handling functions.

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:
 *
 * SET ERROR HANDLER
 * ERROR LOGGING
 * EMERGENCY CLOSE GKS
 */

# include <stdio.h>
# include <gksdescrip.h>          /* GKS descriptor file */
# include <gks3d_defs.h>         /* GKS3D$ binding definition file */

static int error_handler ();

main ()
{
    int      default_conid;
    int      default_wstype;
    struct  dsc$descriptor_s  errorfile_dsc;
    char    *errorfile_name = "ERROR.DAT";
    struct  { int (* error_hand1)();
              int (* error_hand2)(); } new_error_handler;
    struct  { int (* error_hand1)();
              int (* error_hand2)(); } old_error_handler;
    struct  dsc$descriptor_s  string_dsc;
    char    *string_name     = "Test";
    float   tx[2];
    int     ws_id            = 1;

/* Set up the string descriptors. */

    errorfile_dsc.dsc$a_pointer = errorfile_name;
    errorfile_dsc.dsc$b_dtype   = DSC$K_DTYPE_T;
    errorfile_dsc.dsc$b_class   = DSC$K_CLASS_S;
    errorfile_dsc.dsc$w_length  = strlen( errorfile_name );

    string_dsc.dsc$a_pointer    = string_name;
    string_dsc.dsc$b_dtype      = DSC$K_DTYPE_T;
    string_dsc.dsc$b_class      = DSC$K_CLASS_S;
    string_dsc.dsc$w_length     = strlen( string_name );

/* Open the GKS and workstation environments. */

    default_conid = GKS3D$K_CONID_DEFAULT;
    default_wstype = GKS3D$K_WSTYPE_DEFAULT;

    gks3d$open_gks (&errorfile_dsc, 0);
    gks3d$open_ws (&ws_id, &default_conid, &default_wstype);

/* Replace the error handler. */

    new_error_handler.error_hand1 = error_handler;
    new_error_handler.error_hand2 = NULL;

    gks3d$set_error_handler (&new_error_handler, &old_error_handler);
}
```

(continued on next page)

Error-Handling Functions

12.2 Program Example

Example 12–1 (Cont.) Creating an Error Handler

```
/*
 * 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[0] = tx[1] = 0.5;
    gks3d$text (tx, &string_dsc);

/* Release the GKS and workstation environments. */

    gks3d$close_ws (&ws_id);
    gks3d$close_gks ();
}

/* User-defined error handler. */
static int error_handler ( int *error_number,
                          char *function_name[80],
                          char *error_file_name[80] )
{
    static int 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. */

    gks3d$log_error (error_number, function_name, error_file_name);

/* If ERROR_5, abort. */

    if (*error_number == ws_not_active)
        {
        gks3d$emergency_close ();
        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. */  
# include <gks3d$defs.h>  
/* Check for success. */  
    if ( NO_ERROR = gks3d$open_ws( &ws_id, &default_conid,  
        &default_wstype ) );  
    .  
    .  
/* Check for an invalid workstation identifier. */  
    if ( GKS$ERROR_20 = gks3d$activate_ws( &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

Error	Error Message
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

Error	Error Message
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 GKS3D\$K_WSTYPE_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

Error	Error Message
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

Error	Error Message
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	Rectangle definition is invalid in routine **** 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.
52	Viewport is not within the Normalized Device Coordinate unit square in routine **** User Action: DEC GKS allows unclipped primitives to exceed the NDC unit square ($[1,0,0] \times [0,1,0] \times [0,0,1]$), but does not allow you to define a normalization viewport whose boundaries exceed this square. Redefine the function normalization viewport.
53	Workstation window is not within the Normalized Device Coordinate unit square in routine **** User Action: Redefine the function normalization viewport to be within the NDC square ($[1,0,0] \times [0,1,0] \times [0,0,1]$).
54	Workstation viewport is not within the display space in routine **** 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.

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	Polyline index is invalid in routine **** User Action: Make sure the arguments are specified in the correct order and the index is an integer.
61	A representation for the specified polyline index has not been defined on this workstation in routine **** User Action: Use SET POLYLINE REPRESENTATION to define a representation for the index, or use another predefined index value.
62	A representation for the specified polyline index has not been predefined on this workstation in routine ****

(continued on next page)

DEC GKS Error Messages A.4 Attribute Function Errors

Table A-4 (Cont.) Attribute Function Errors

Error	Error Message
	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 marker 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

Error	Error Message
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

Error	Error Message
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.
308	Arithmetic error has occurred in routine ****

(continued on next page)

Table A-11 (Cont.) System Errors

Error	Error Message
	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.
410	A representation for the specified view index has not been predefined on this workstation in routine ****

(continued on next page)

DEC GKS Error Messages

A.11 System Errors

Table A-11 (Cont.) System Errors

Error	Error Message
	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.
411	Workstation window limits are not within the NPC cube in routine **** User Action: Make sure the workstation window limits are in the range 0.0 to 1.0.
412	Back clipping plane is in front of front clipping plane in routine **** User Action: Make sure the back clipping plane is behind the front clipping plane.
413	View clipping limits not within NPC range in routine **** User Action: Make sure the view clipping limits are in the range 0.0 and 1.0.
420	Edge index is invalid in routine **** User Action: Make sure the edge index is greater than or equal to 1.
421	A representation for the specified edge index has not been defined on this workstation in routine **** User Action: Make sure the edge representation is predefined. Use the INQUIRE EDGE FACILITIES function for information on the predefined edge representations.
422	A representation for the specified edge index has not been predefined on this workstation in routine **** User Actions: Make sure the specified edge index has been predefined.
423	Edge type is equal to zero in routine **** User Actions: Make sure the edge type is one of the permitted values. See Appendix B for a list of edge type values.
424	Specified edge type is not supported on this workstation in routine **** 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.
425	Edge width scale factor is less than zero in routine **** User Action: Make sure the edge width scale factor is greater than or equal to 0.0.
426	Pattern reference vectors are collinear in routine **** User Action: Make sure the pattern reference vectors are not collinear.
427	Specified HLHSR mode not supported on workstation in routine ****

(continued on next page)

Table A-11 (Cont.) System Errors

Error	Error Message
	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 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-12 lists the errors that are implementation specific.

Table A-12 Implementation-Specific Errors

Error	Error Message
-2	Requested color map could not be created as specified in routine **** 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.
-3	Invalid data in workstation description file in routine **** User Action: Make sure the format of your description file is valid for your particular workstation.

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-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 GKS3D\$K_ASF_BUNDLED (0) or GKS3D\$K_ASF_INDIVIDUAL (1).</p>
-11	<p>Invalid value specified for fill area interior style in routine ****</p> <p>User Action: Make sure you passed one of the values GKS3D\$K_INTSTYLE_HOLLOW (0), GKS3D\$K_INTSTYLE_SOLID (1), GKS3D\$K_INTSTYLE_PATTERN(2), or GKS3D\$K_INTSTYLE_HATCH (3).</p>
-12	<p>Invalid value specified for horizontal component of text alignment in routine ****</p>

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	User Action: Make sure you passed one of the values GKS3D\$K_TEXT_HALIGN_NORMAL (0), GKS3D\$K_TEXT_HALIGN_LEFT (1), GKS3D\$K_TEXT_HALIGN_CENTER (2), or GKS3D\$K_TEXT_HALIGN_RIGHT (3).
-13	Invalid value specified for vertical component of text alignment in routine **** User Action: Make sure you passed one of the values GKS3D\$K_TEXT_VALIGN_NORMAL (0), GKS3D\$K_TEXT_VALIGN_TOP (1), GKS3D\$K_TEXT_VALIGN_CAP (2), GKS3D\$K_TEXT_VALIGN_HALF (3), GKS3D\$K_TEXT_VALIGN_BASE (4), or GKS3D\$K_TEXT_VALIGN_BOTTOM (5).
-14	Invalid value specified for text precision in routine **** User Action: Make sure you passed one of the values GKS3D\$K_TEXT_PRECISION_STRING (0), GKS3D\$K_TEXT_PRECISION_CHAR (1), or GKS3D\$K_TEXT_PRECISION_STROKE (2).
-15	Invalid value specified for text path in routine **** User Action: Make sure you passed one of the values GKS3D\$K_TEXT_PATH_RIGHT (0), GKS3D\$K_TEXT_PATH_LEFT (1), GKS3D\$K_TEXT_PATH_UP (2), or GKS3D\$K_TEXT_PATH_DOWN (3).
-16	Echo switch is invalid in routine **** User Action: Make sure you passed the one of the values GKS3D\$K_NOECHO (0) or GKS3D\$K_ECHO (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.
-17	Inquired device values not set or realized in routine **** User Action: Check the value type argument to make sure it is either GKS3D\$K_VALUE_SET (0) or GKS3D\$K_VALUE_REALIZED (1).
-18	The following error occurred when GKS was interpreting an item **** User Action: An error occurred while interpreting a metafile item. DEC GKS follows this error message with another message that signals the appropriate action.
-19	Invalid error status parameter specified in routine **** 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.
-20	GKS not in proper state: GKS in the ERROR state in routine **** 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.
-21	Function is not supported in this level of GKS in routine ****

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	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 GKS3D\$K_NOCLIP (0) or GKS3D\$K_CLIP (1).
-24	Invalid value specified for viewport priority flag in routine **** User Action: Make sure you passed either the value GKS3D\$K_INPUT_PRIORITY_HIGHER (0) or GKS3D\$K_INPUT_PRIORITY_LOWER (1).
-25	Invalid value specified for update workstation flag in routine **** User Action: Make sure you passed either the value GKS3D\$K_POSTPONE_FLAG (0) or GKS3D\$K_PERFORM_FLAG (1).
-26	Invalid value specified for deferral mode in routine **** User Action: Make sure you passed one of the values GKS3D\$K_ASAP (0), GKS3D\$K_BNIG (1), GKS3D\$K_BNIL (2), or GKS3D\$K_ASTI (3).
-27	Invalid value specified for regeneration mode in routine **** User Action: Make sure you passed either the value GKS3D\$K_IRG_SUPPRESSED (0) or GKS3D\$K_IRG_ALLOWED (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 ****

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	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> .
-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.12 Implementation-Specific Errors

Table A-12 (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.

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-93	Internal GKS error: Prompt and echo type not supported in routine **** 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 GKS3D\$K_ESC_SET_EDGE_INDEX 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 (GKS3D\$K_ESC_SET_EDGE_WIDTH) 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.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-155	Display speed is less than zero in routine **** User Action: Pass a positive real value to the escape Set Speed (GKS3D\$K_ESC_SET_SPEED).
-156	Loudness is outside range [0,1] in routine **** User Action: Pass a valid value to the escape Beep (GKS3D\$K_ESC_BEEP) .
-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 GKS3D\$K_NORMAL (0) or GKS3D\$K_HIGHLIGHTED (1).

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-301	<p>Invalid value specified for visibility in routine ****</p> <p>User Action: Make sure you specify either GKS3D\$K_INVISIBLE (0) or GKS3D\$K_VISIBLE (1).</p>
-302	<p>Invalid value specified for detectability in routine ****</p> <p>User Action: Make sure you specify either GKS3D\$K_UNDETECTABLE (0) or GKS3D\$K_DETECTABLE (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>

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	<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>
-452	<p>PCM initialization failed—hardware dials and buttons not available in routine ****</p> <p>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>.</p>
-453	<p>PCM configuration failed—valuator or choice device not initialized in routine ****</p> <p>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>.</p>
-1517	<p>Error trying to open UID database</p> <p>User Action: Verify that the UID files <code>gfx_decw.uid</code> and <code>gfx_decw_XX.uid</code> (where <code>XX</code> is the language code) exist in either the specified system area or user path.</p>
-1518	<p>Error trying to fetch from UID database</p> <p>User Action: Verify that the name used in the <code>DwtFetchWidget</code> or <code>DwtFetchWidgetOverride</code> 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.</p>
-2000	<p>Illegal call to routine **** while GKS is in an error state</p> <p>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.</p>
-2001	<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 <code>GKS3D\$K_ASF_BUNDLED</code> (0) or <code>GKS3D\$K_ASF_INDIVIDUAL</code> (1).</p>
-2002	<p>Invalid clipping indicator in routine ****</p> <p>User Action: Make sure you passed either the value <code>GKS3D\$K_NOCLIP</code> (0) or <code>GKS3D\$K_CLIP</code> (1).</p>
-2003	<p>Invalid normalization transformation priority in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the normalization priority value is either <code>GKS3D\$K_INPUT_PRIORITY_HIGHER</code> (0) or <code>GKS3D\$K_INPUT_PRIORITY_LOWER</code> (1).</p>

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-2004	<p>Invalid view index specified in routine ****</p> <p>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).</p>
-2005	<p>Invalid metafile format found in routine ****</p> <p>User Action: Make sure the input file you are trying to interpret is a metafile that was generated by DEC GKS.</p>
-2006	<p>Null text direction vector found in routine ****</p> <p>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).</p>
-2007	<p>Invalid text path specified in routine ****</p> <p>User Action: Make sure you passed one of the values GKS3D\$K_TEXT_PATH_RIGHT (0), GKS3D\$K_TEXT_PATH_LEFT (1), GKS3D\$K_TEXT_PATH_UP (2), or GKS3D\$K_TEXT_PATH_DOWN (3).</p>
-2008	<p>Invalid text character alignment specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the text alignment specified is a permitted value (GKS3D\$K_TEXT_HALIGN_NORMAL (0), GKS3D\$K_TEXT_HALIGN_LEFT (1), GKS3D\$K_TEXT_HALIGN_CENTER (2), or GKS3D\$K_TEXT_HALIGN_RIGHT (3) for the horizontal component, and GKS3D\$K_TEXT_VALIGN_NORMAL (0), GKS3D\$K_TEXT_VALIGN_TOP (1), GKS3D\$K_TEXT_VALIGN_CAP (2), GKS3D\$K_TEXT_VALIGN_HALF (3), GKS3D\$K_TEXT_VALIGN_BASE (4), or GKS3D\$K_TEXT_VALIGN_BOTTOM (5) for the vertical component).</p>
-2009	<p>Invalid fill area interior style specified in routine ****</p> <p>User Action: Make sure you passed one of the values GKS3D\$K_INTSTYLE_HOLLOW (0), GKS3D\$K_INTSTYLE_SOLID (1), GKS3D\$K_INTSTYLE_PATTERN(2), or GKS3D\$K_INTSTYLE_HATCH (3).</p>
-2010	<p>Invalid edge flag specified in routine ****</p> <p>User Action: Make sure the arguments are specified in the correct order and that the specified edge flag is a permitted value (GKS3D\$EDGE_FLAG_OFF (0) or GKS3D\$EDGE_FLAG_ON (1)).</p>
-2011	<p>Invalid aspect source flag specified in routine ****</p> <p>User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GKS3D\$K_ASF_BUNDLED (0) or GKS3D\$K_ASF_INDIVIDUAL (1).</p>
-2012	<p>Null pattern reference vector found in routine ****</p>

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	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 (GKS3D\$K_CLEAR_CONDITIONALLY (0) or GKS3D\$K_CLEAR_ALWAYS (1)).
-2015	Invalid update control flag specified in routine **** User Action: Make sure you passed one of the values GKS3D\$K_ASAP (0), GKS3D\$K_BNIG (1), GKS3D\$K_BNIL (2), or GKS3D\$K_ASTI (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 (GKS3D\$K_IRG_SUPPRESSED (0) or GKS3D\$K_IRG_ALLOWED (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 (GKS3D\$K_IRG_SUPPRESSED (0) or GKS3D\$K_IRG_ALLOWED (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 (GKS3D\$K_INPUT_PRIORITY_HIGHER (0) or GKS3D\$K_INPUT_PRIORITY_LOWER (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 (GKS3D\$K_TEXT_PRECISION_STRING (0), GKS3D\$K_TEXT_PRECISION_CHAR (1), or GKS3D\$K_TEXT_PRECISION_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 (GKS3D\$K_COORDINATES_WC (0) or GKS3D\$K_COORDINATES_NDC (1)).
-2021	Invalid projection type argument in routine ****

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	User Action: Make sure the arguments are specified in the correct order and that the projection type argument is either GKS3D\$K_PARALLEL_PROJECTION (0) or GKS3D\$K_PERSPECTIVE_PROJECTION (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: GKS3D\$K_INPUT_MODE_REQUEST (0), GKS3D\$K_INPUT_MODE_SAMPLE (1), or GKS3D\$K_INPUT_MODE_EVENT (2).
-2025	Invalid input device echo state specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the input device echo state is either GKS3D\$K_NOECHO (0) or GKS3D\$K_ECHO (1).
-2026	Invalid polyline/fill area control flag in data record in routine **** User Action: Make sure the arguments are specified in the correct order and that the polyline /fill area control flag is a permitted value (GKS3D\$K_ACF_POLYLINE (0) or GKS3D\$K_ACF_FILL_AREA (1)).
-2027	Invalid attribute control flag in data record in routine **** 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 (GKS3D\$K_ACF_CURRENT (0) or GKS3D\$K_ACF_SPECIFIED (1)).
-2028	Invalid data record size for specified prompt and echo type in routine **** User Action: Check the specified data record size against the data record required for the specified PET.
-2029	Invalid normalization transformation number specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified normalization transformation number is a permitted value.
-2030	NDC position(s) outside NDC unit cube in routine **** User Action: Make sure the specified NDC values are in the range 0.0 to 1.0.
-2031	Invalid input device class specified in routine ****

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	<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: GKS3D\$K_INPUT_CLASS_NONE (0), GKS3D\$K_INPUT_CLASS_LOCATOR (1), GKS3D\$K_INPUT_CLASS_STROKE (2), GKS3D\$K_INPUT_CLASS_VALUATOR (3), GKS3D\$K_INPUT_CLASS_CHOICE (4), GKS3D\$K_INPUT_CLASS_PICK (5), GKS3D\$K_INPUT_CLASS_STRING (6), or GKS3D\$K_INPUT_CLASS_VIEWPORT (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 GKS\$K_VISIBLE (0) or GKS\$K_INVISIBLE (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 GKS3D\$K_NORMAL (0) or GKS3D\$K_HIGHLIGHTED (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 GKS3D\$K_UNDETECTABLE (0) or GKS3D\$K_DETECTABLE (1).</p>
-2036	<p>Invalid matrix specified, matrix has a zero fourth row in routine *****</p> <p>User Action: Make sure the specified matrix does not have a zero fourth row.</p>
-2037	<p>Warning, the specified view matrix is not invertible in routine *****</p> <p>User Action: Specify a matrix that is invertible.</p>
-2038	<p>Warning, a 3D item cannot be written into GKSM (2D metafile output) in routine *****</p> <p>User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.</p>
-2039	<p>Warning, the normalization transformation is 3D and cannot be used with GKSM (2D metafile output) in routine *****</p> <p>User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.</p>
-2040	<p>Warning, the segment transformation is 3D and cannot be used with GKSM (2D metafile output) in routine *****</p> <p>User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.</p>
-2041	<p>Warning, 3D inquiries are invalid with GKSM (2D metafile output) in routine *****</p>

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	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.
-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 gks_descrip.h 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> .

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
-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 GKS\$LIST_TYPES is as expected.
-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 ****

(continued on next page)

DEC GKS Error Messages A.12 Implementation-Specific Errors

Table A-12 (Cont.) Implementation-Specific Errors

Error	Error Message
	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 **** 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.

(continued on next page)

DEC GKS Error Messages

A.12 Implementation-Specific Errors

Table A–12 (Cont.) Implementation-Specific Errors

Error	Error Message
–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.

A.13 Fatal Errors

Table A–13 lists the fatal DEC GKS and system errors.

Table A–13 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.

(continued on next page)

Table A-13 (Cont.) Fatal Errors

Error	Error Message
-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.
-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 GKS3D\$ interface. Using constants in your DEC GKS programs makes your code easier to read.

To use constants in your program, you must include a language definitions file in your code. The language definitions files located in SYS\$LIBRARY are as follows:

- GKS3D\$DEFS.ADA for VAX Ada
- GKS3D\$DEFS.BAS for VAX BASIC
- GKS3D\$DEFS.FOR for VAX FORTRAN using the GKS3D\$ functions
- GKS3D\$DEFS.H for VAX C
- GKS3D\$DEFS.PAS for VAX Pascal
- GKS3D\$DEFS.PLI for VAX PL/I routines declared as procedures (no value returns)
- GKS3D\$DEFS.R32 for VAX BLISS

Table B-1 lists the GKS3D\$ constants. It contains the defined constant exactly as it is used in programming, its associated value as defined in GKS, and a brief description of the constant. The constants are listed by category and numerically ordered by value within the category.

Note

In GKS3D\$ constants names, you can use either COLOUR or COLOR, and METRES or METERS. Both constants are defined with the same value. For example, GKS3D\$K_COLOUR_MODEL_RGB and GKS3D\$K_COLOR_MODEL_RGB are both defined with the value 1.

Table B-1 GKS3D\$ Constants

Constant	Value	Description
Aspect Source Flags:		
GKS3D\$K_ASF_BUNDLED	0	Bundled
GKS3D\$K_ASF_INDIVIDUAL	1	Individual

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Attribute Control Flags:		
GKS3D\$K_ACF_CURRENT	0	Attribute control value
GKS3D\$K_ACF_SPECIFIED	1	Attribute control value current
GKS3D\$K_ACF_POLYLINE	0	Data record polyline control flag
GKS3D\$K_ACF_FILL_AREA	1	Data record fill area control flag
Choice Input Prompt Flags:		
GKS3D\$K_CHOICE_PROMPT_OFF	0	Prompt off
GKS3D\$K_CHOICE_PROMPT_ON	1	Prompt on
Choice Status Types:		
GKS3D\$K_STATUS_NONE	0	No input obtained
GKS3D\$K_STATUS_OK	1	Input obtained
GKS3D\$K_STATUS_NOCHOICE	2	No choice input
Clear Screen States:		
GKS3D\$K_CLEAR_CONDITIONALLY	0	Clear conditionally
GKS3D\$K_CLEAR_ALWAYS	1	Clear always
Clipping Flags:		
GKS3D\$K_NOCLIP	0	Clipping off
GKS3D\$K_CLIP	1	Clipping on
Color Availability Flags:		
GKS3D\$K_MONOCHROME	0	Monochrome
GKS3D\$K_COLOUR	1	Color
Color Models:		
GKS3D\$K_COLOUR_MODEL_RGB	1	Red, green, and blue color model
GKS3D\$K_COLOUR_MODEL_CIE	2	Commision Internationale de l'Eclairage color model
GKS3D\$K_COLOUR_MODEL_HSV	3	Hue, saturation, and value color model
GKS3D\$K_COLOUR_MODEL_HLS	4	Hue, lightness, and saturation color model
Color Validity States:		
GKS3D\$K_INVALID_ABSENT	0	Invalid indexes absent
GKS3D\$K_INVALID_PRESENT	1	Invalid indexes present

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Coordinate Switch:		
GKS3D\$K_COORDINATES_WC	0	World coordinates
GKS3D\$K_COORDINATES_NDC	1	Normalized device coordinates
Data Type Identifiers:		
GKS3D\$K_TYP_INTEGER	0	Integer
GKS3D\$K_TYP_CHOICE_REC	1	Choice input data record
GKS3D\$K_TYP_LOC_REC	2	Locator input data record
GKS3D\$K_TYP_PICK_REC	3	Pick input data record
GKS3D\$K_TYP_STRING_REC	4	String input data record
GKS3D\$K_TYP_STROKE_REC	5	Stroke input data record
GKS3D\$K_TYP_VAL_REC	6	Valuator input data record
GKS3D\$K_TYP_ESCAPE_REC	7	Escape data record
GKS3D\$K_TYP_GDP_REC	8	GDP data record
GKS3D\$K_TYP_REAL	9	Real
GKS3D\$K_TYP_ADDRESS	10	Address
Default Connection Identifier:		
GKS3D\$K_CONID_DEFAULT	0	Default connection identifier
Deferral Modes:		
GKS3D\$K_ASAP	0	As soon as possible
GKS3D\$K_BNIG	1	Before the next global interaction
GKS3D\$K_BNIL	2	Before the next local interaction
GKS3D\$K_ASTI	3	At some time
Detectability Flags:		
GKS3D\$K_UNDETECTABLE	0	Segment cannot be picked.
GKS3D\$K_DETECTABLE	1	Segment, if visible, can be picked.
Device Coordinate Units:		
GKS3D\$K_METRES	0	Meters
GKS3D\$K_OTHER_UNITS	1	Other units
Display Surface States:		
GKS3D\$K_NOTEMPTY	0	Display surface not empty
GKS3D\$K_EMPTY	1	Display surface empty

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Dynamic Modification States:		
GKS3D\$K_IRG	0	Implicit regeneration
GKS3D\$K_IMM	1	Immediate
Echo States:		
GKS3D\$K_NOECHO	0	Echo disabled
GKS3D\$K_ECHO	1	Echo enabled
Edge Flags:		
GKS3D\$K_EDGE_FLAG_OFF	0	Edge flag off
GKS3D\$K_EDGE_FLAG_ON	1	Edge flag on
Edge Types:		
GKS3D\$K_EDGE_SOLID	1	Line type solid
GKS3D\$K_EDGE_DASHED	2	Line type dashed
GKS3D\$K_EDGE_DOTTED	3	Line type dotted
GKS3D\$K_EDGE_DASHED_DOTTED	4	Line type dash-dotted
GKS3D\$K_EDGE_DASH_2_DOT	–1	Line type double-dashed dotted
GKS3D\$K_EDGE_DASH_3_DOT	–2	Line type triple-dashed dotted
GKS3D\$K_EDGE_LONG_DASH	–3	Line type long-dashed
GKS3D\$K_EDGE_LONG_SHORT_DASH	–4	Line type long-short-dashed
GKS3D\$K_EDGE_SPACED_DASH	–5	Line type spaced-dashed
GKS3D\$K_EDGE_SPACED_DOT	–6	Line type spaced-dotted
GKS3D\$K_EDGE_DOUBLE_DOT	–7	Line type double dotted
GKS3D\$K_EDGE_TRIPLE_DOT	–8	Line type triple dotted
Error Handling Modes:		
GKS3D\$K_ERROR_OFF	0	No error handling
GKS3D\$K_ERROR_ON	1	Error handling
Escape Identifiers:		
GKS3D\$K_ESC_SET_SPEED	–100	Set Display Speed
GKS3D\$K_ESC_PRINT	–101	Generate Hardcopy of Workstation Surface
GKS3D\$K_ESC_BEEP	–103	Beep
GKS3D\$K_ESC_POP_WORKSTATION	–106	Pop Workstation
GKS3D\$K_ESC_PUSH_WORKSTATION	–107	Push Workstation
GKS3D\$K_ESC_ERROR_HANDLING_MODE	–108	Set Error Handling Mode

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Escape Identifiers:		
GKS3D\$K_ESC_SET_VIEWPORT_EVENT	-109	Set Viewport Event
GKS3D\$K_ESC_ASSOC_WSTYPE_CONID	-110	Associated Workstation Type and Connection ID
GKS3D\$K_ESC_SET_SOFT_CLIP	-111	Software Clipping
GKS3D\$K_ESC_SET_WRITING_MODE	-150	Set Writing Mode
GKS3D\$K_ESC_SET_LINE_CAP	-151	Set Line Cap Style
GKS3D\$K_ESC_SET_LINE_JOIN	-152	Set Line Join Style
GKS3D\$K_ESC_SET_EDGE_CTL	-153	Set Edge Control Flag
GKS3D\$K_ESC_SET_EDGE_TYPE	-154	Set Edge Type
GKS3D\$K_ESC_SET_EDGE_WIDTH	-155	Set Edge Width Scale Factor
GKS3D\$K_ESC_SET_EDGE_COLOUR_IND	-156	Set Edge Color Index
GKS3D\$K_ESC_SET_EDGE_INDEX	-157	Set Edge Index
GKS3D\$K_ESC_SET_EDGE_ASF	-158	Set Edge Aspect Source Flag
GKS3D\$K_ESC_BEGIN_TRANS_BLOCK	-160	Begin Transformation Block
GKS3D\$K_ESC_END_TRANS_BLOCK	-161	End Transformation Block
GKS3D\$K_ESC_SET_SEG_HIGHL_MODE	-162	Set Segment Highlighting Method
GKS3D\$K_ESC_SET_HIGHL_MODE	-163	Set Highlighting Method
GKS3D\$K_ESC_BEGIN_TRANS_BLOCK_3	-164	Begin Transformation Block 3
GKS3D\$K_ESC_SET_EDGE_REP	-200	Set Edge Representation
GKS3D\$K_ESC_SET_WINDOW_TITLE	-202	Set Window Title
GKS3D\$K_ESC_SET_RESET_STRING	-203	Set Reset String
GKS3D\$K_ESC_SET_CANCEL_STRING	-204	Set Cancel String
GKS3D\$K_ESC_SET_ENTER_STRING	-205	Set Enter String
GKS3D\$K_ESC_SET_ICON_BITMAPS	-206	Set Icon Bit Maps
GKS3D\$K_ESC_INQ_WRITING_MODE	-251	Inquire Current Writing Mode
GKS3D\$K_ESC_INQ_LINE_CAP	-252	Inquire Current Line Cap Style
GKS3D\$K_ESC_INQ_LINE_JOIN	-253	Inquire Current Line Join Style
GKS3D\$K_ESC_INQ_EDGE_ATTR	-254	Inquire Current Edge Attributes
GKS3D\$K_ESC_INQ_VIEWPORT_DATA	-255	Inquire Viewport Data
GKS3D\$K_ESC_INQ_SPEED	-300	Inquire Current Display Speed
GKS3D\$K_ESC_INQ_LIST_EDGE_INDEX	-302	Inquire List of Edge Indexes
GKS3D\$K_ESC_INQ_SEGMENT_EXTENT	-303	Inquire Segment Extent
GKS3D\$K_ESC_INQ_WINDOW_IDS	-304	Inquire Window Identifiers
GKS3D\$K_ESC_INQ_SEG_HIGHL_MODE	-305	Inquire Segment Highlighting Method
GKS3D\$K_ESC_INQ_HIGHL_MODE	-306	Inquire Highlighting Method
GKS3D\$K_ESC_INQ_PASTEBOARD_ID	-307	Inquire Pasteboard Identifier
GKS3D\$K_ESC_INQ_MENU_BAR_ID	-308	Inquire Menu Bar Identifier

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Escape Identifiers:		
GKS3D\$K_ESC_INQ_SHELL_ID	–309	Inquire Shell Identifier
GKS3D\$K_ESC_INQ_LIST_ESC	–350	Inquire List of Available Escapes
GKS3D\$K_ESC_INQ_DEF_SPEED	–351	Inquire Default Display Speed
GKS3D\$K_ESC_INQ_LINE_CAP_JOIN_F	–352	Inquire Line Cap and Join Facilities
GKS3D\$K_ESC_INQ_EDGE_FAC	–354	Inquire Edge Facilities
GKS3D\$K_ESC_INQ_PREDEF_EDGE_REP	–355	Inquire Predefined Edge Representation
GKS3D\$K_ESC_INQ_MAX_EDGE_BUNDLE	–356	Inquire Maximum Number of Edge Bundles
GKS3D\$K_ESC_INQ_LIST_HIGHL_MODE	–358	Inquire List of Highlighting Methods
GKS3D\$K_ESC_INQ_EDGE_REP	–359	Inquire Edge Representation
GKS3D\$K_ESC_MAP_NDC_OF_WC	–400	Evaluate NDC Mapping of WC Point
GKS3D\$K_ESC_MAP_DC_OF_NDC	–401	Evaluate DC Mapping of NDC Point
GKS3D\$K_ESC_MAP_WC_OF_NDC	–402	Evaluate WC Mapping of NDC Point
GKS3D\$K_ESC_MAP_NDC_OF_DC	–403	Evaluate NDC Mapping of DC Point
GKS3D\$K_ESC_INQ_GDP_EXTENT	–404	Inquire Extent of a GDP
GKS3D\$K_ESC_DOUBLE_BUFFER	–500	Set Double Buffering
GKS3D\$K_ESC_SET_BCKGRND_PIXMAP	–501	Set Background Pixmap
GKS3D\$K_ESC_INQ_DBUFFER_PIXMAP	–502	Inquire Double Buffer Pixmap
GKS3D\$K_ESC_INQ_BCKGRND_PIXMAP	–503	Inquire Background Pixmap
Fill Area Hatch Styles:		
GKS3D\$K_HATCH_CROSS	–1	Cross-hatches
GKS3D\$K_HATCH_UP_DIAG	–2	Diagonal lines at 45 degrees
GKS3D\$K_HATCH_DOWN_DIAG	–3	Diagonal lines at –45 degrees
GKS3D\$K_HATCH_HORIZ	–4	Horizontal lines
GKS3D\$K_HATCH_VERT	–5	Vertical lines
GKS3D\$K_HATCH_UP_DIAG_SPARSE	–6	Diagonal lines at 45 degrees—sparse
GKS3D\$K_HATCH_DOWN_DIAG_SPARSE	–7	Diagonal lines at –45 degrees—sparse
GKS3D\$K_HATCH_HORIZ_SPARSE	–8	Horizontal lines—sparse
GKS3D\$K_HATCH_VERT_SPARSE	–9	Vertical lines—sparse
Fill Area Interior Styles:		
GKS3D\$K_INTSTYLE_HOLLOW	0	Interior style hollow
GKS3D\$K_INTSTYLE_SOLID	1	Interior style solid
GKS3D\$K_INTSTYLE_PATTERN	2	Interior style pattern
GKS3D\$K_INTSTYLE_HATCH	3	Interior style hatched

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
GDP Bundle Types:		
GKS3D\$K_POLYLN_ATTRI	0	GDP polyline bundle
GKS3D\$K_POLYMR_ATTRI	1	GDP polymarker bundle
GKS3D\$K_TEXT_ATTRI	2	GDP text bundle
GKS3D\$K_FILLAR_ATTRI	3	GDP fill area bundle
GKS3D\$K_EDGE_ATTRI	4	GDP edge bundle
GDP Graphic Primitives:		
GKS3D\$K_GDP_DISJOINT_PLINE	-100	Disjoint Polyline
GKS3D\$K_GDP_CIRCLE_CTR_PT	-101	Circle: Center and Point on Circumference
GKS3D\$K_GDP_CIRCLE_3PT	-102	Circle: Three Points on Circumference
GKS3D\$K_GDP_CIRCLE_CTR_RAD	-103	Circle: Center and Radius
GKS3D\$K_GDP_CIRCLE_2PT_RAD	-104	Circle: Two Points on Circumference, and Radius
GKS3D\$K_GDP_ARC_CTR_2PT	-106	Arc: Center and Two Points on Arc
GKS3D\$K_GDP_ARC_3PT	-107	Arc: Three Points on Circumference
GKS3D\$K_GDP_ARC_CTR_2V_RAD	-108	Arc: Center, 2 Vectors, and a Radius
GKS3D\$K_GDP_ARC_2PT_RAD	-109	Arc: Two Points on Arc and Radius
GKS3D\$K_GDP_ARC_CTR_PT_ANG	-110	Arc: Center, Starting Point, and Angle
GKS3D\$K_GDP_ELLIPSE_CTR_AXES	-111	Ellipse: Center, and Two Axis Vectors
GKS3D\$K_GDP_ELLIPSE_FOCII_PT	-113	Ellipse: Focal Points and Point on Circumference
GKS3D\$K_GDP_ELIARC_CTR_AXES_2V	-114	Elliptic Arc: Center, Two Axis Vectors, and Two Vectors
GKS3D\$K_GDP_ELIARC_FOCII_2PT	-116	Elliptic Arc: Focal Points and Two Points on Circumference
GKS3D\$K_GDP_RECT_2PT	-125	Rectangle: Two Corners
GKS3D\$K_GDP_FILL_AREA_SET	-332	Fill Area Set
GKS3D\$K_GDP_FCIRCLE_CTR_PT	-333	Filled Circle: Center and Point on Circumference
GKS3D\$K_GDP_FCIRCLE_3PT	-334	Filled Circle: Three Points on Circumference
GKS3D\$K_GDP_FCIRCLE_CTR_RAD	-335	Filled Circle: Center and Radius
GKS3D\$K_GDP_FCIRCLE_2PT_RAD	-336	Filled Circle: Two Points on Circumference, and Radius
GKS3D\$K_GDP_FARC_CTR_2PT	-338	Filled Arc: Center and Two Points on Arc
GKS3D\$K_GDP_FARC_3PT	-339	Filled Arc: Three Points on Circumference
GKS3D\$K_GDP_FARC_CTR_2V_RAD	-340	Filled Arc: Center, Two Vectors, and a Radius
GKS3D\$K_GDP_FARC_2PT_RAD	-341	Filled Arc: Two Points on Arc, and Radius

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
GDP Graphic Primitives:		
GKS3D\$K_GDP_FARC_CTR_PT_ANG	–342	Filled Arc: Center, Starting Point, and Angle
GKS3D\$K_GDP_FELLIPSE_CTR_AXES	–343	Filled Ellipse: Center and Two Axis Vectors
GKS3D\$K_GDP_FELLIPSE_FOCII_PT	–345	Filled Ellipse: Focal Points and Point on Circumference
GKS3D\$K_GDP_FELIARC_CTR_AXES_2V	–346	Filled Elliptic Arc: Center, Two Axis Vectors, and Two Vectors
GKS3D\$K_GDP_FELIARC_FOCII_2PT	–348	Filled Elliptic Arc: Focal Points and Two Points on Circumference
GKS3D\$K_GDP_FRECT_2PT	–349	Filled Rectangle: Two Corners
GKS3D\$K_GDP_IMAGE_ARRAY	–400	Packed Cell Array
GKS Level Types:		
GKS3D\$K_LEVEL_MA	–3	Minimal output, no input
GKS3D\$K_LEVEL_MB	–2	Minimal output, request input
GKS3D\$K_LEVEL_MC	–1	Minimal output, full input
GKS3D\$K_LEVEL_0A	0	All primitives and attributes, no input
GKS3D\$K_LEVEL_0B	1	All primitives and attributes, request input
GKS3D\$K_LEVEL_0C	2	All primitives and attributes, full input
GKS3D\$K_LEVEL_1A	3	Basic segmentation with full output, no input
GKS3D\$K_LEVEL_1B	4	Basic segmentation with full output, request input
GKS3D\$K_LEVEL_1C	5	Basic segmentation with full output, full input
GKS3D\$K_LEVEL_2A	6	Workstation-independent and segment storage, no input
GKS3D\$K_LEVEL_2B	7	Workstation-independent and segment storage, request input
GKS3D\$K_LEVEL_2C	8	Workstation-independent and segment storage, full input
GKS Operating States:		
GKS3D\$K_GKCL	0	GKS closed
GKS3D\$K_GKOP	1	GKS open
GKS3D\$K_WSOP	2	At least one workstation open
GKS3D\$K_WSAC	3	At least one workstation active
GKS3D\$K_SGOP	4	At least one segment open

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
HLHSR Identifiers:		
GKS3D\$K_HLHSR_ID_NONE	0	HLHSR computation off
GKS3D\$K_HLHSR_ID_PAINTER	1	HLHSR computation on
HLHSR Modes:		
GKS3D\$K_HLHSR_MODE_OFF	0	HLHSR computation off
GKS3D\$K_HLHSR_MODE_NONE	0	HLHSR computation off
GKS3D\$K_HLHSR_MODE_ON	1	HLHSR computation on
GKS3D\$K_HLHSR_MODE_PAINTER	1	Painter's algorithm
Highlighting Flags:		
GKS3D\$K_NORMAL	0	DEC GKS does not highlight the segment.
GKS3D\$K_HIGHLIGHTED	1	DEC GKS highlights the segment
Highlighting Methods:		
GKS3D\$K_HIGH_METHOD_DEFAULT	0	Default highlighting
GKS3D\$K_HIGH_METHOD_COMP	1	Highlight with complement mode
GKS3D\$K_HIGH_METHOD_COLOUR	2	Highlight with color
GKS3D\$K_HIGH_METHOD_LINE	3	Highlight with extent line box
GKS3D\$K_HIGH_METHOD_FILL	4	Highlight with extent fill area
GKS3D\$K_HIGH_METHOD_DUAL	5	Highlight with extent line box and fill area
Implicit Regeneration States:		
GKS3D\$K_IRG_SUPPRESSED	0	Implicit regeneration suppressed
GKS3D\$K_IRG_ALLOWED	1	Implicit regeneration allowed
Input Classes:		
GKS3D\$K_INPUT_CLASS_NONE	0	No input class
GKS3D\$K_INPUT_CLASS_LOCATOR	1	Locator input class
GKS3D\$K_INPUT_CLASS_STROKE	2	Stroke input class
GKS3D\$K_INPUT_CLASS_VALUATOR	3	Valuator input class
GKS3D\$K_INPUT_CLASS_CHOICE	4	Choice input class
GKS3D\$K_INPUT_CLASS_PICK	5	Pick input class
GKS3D\$K_INPUT_CLASS_STRING	6	String input class
GKS3D\$K_INPUT_CLASS_VIEWPORT	7	Viewport input class

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Input Mode Types:		
GKS3D\$K_INPUT_MODE_REQUEST	0	Request mode
GKS3D\$K_INPUT_MODE_SAMPLE	1	Sample mode
GKS3D\$K_INPUT_MODE_EVENT	2	Event mode
Input Priority States:		
GKS3D\$K_INPUT_PRIORITY_HIGHER	0	Relative input priority higher
GKS3D\$K_INPUT_PRIORITY_LOWER	1	Relative input priority lower
Input Status Types:		
GKS3D\$K_STATUS_NONE	0	No input obtained
GKS3D\$K_STATUS_OK	1	Input obtained
GKS3D\$K_STATUS_NOCHOICE	2	Input is NOCHOICE
GKS3D\$K_STATUS_NOPICK	2	Input is NOPICK
Line Cap Styles:		
GKS3D\$K_LINE_CAP_BUTT	2	Line cap type butted
GKS3D\$K_LINE_CAP_ROUND	3	Line cap type rounded
GKS3D\$K_LINE_CAP_SQUARE	4	Line cap type square
Line Join Styles:		
GKS3D\$K_LINE_JOIN_MITRE	2	Line join type miter
GKS3D\$K_LINE_JOIN_ROUND	3	Line join type round
GKS3D\$K_LINE_JOIN_BEVEL	4	Line join type bevel
Line Types (Standard):		
GKS3D\$K_LINETYPE_SOLID	1	Line type solid
GKS3D\$K_LINETYPE_DASHED	2	Line type dashed
GKS3D\$K_LINETYPE_DOTTED	3	Line type dotted
GKS3D\$K_LINETYPE_DASHED_DOTTED	4	Line type dash-dotted
Line Types (DEC GKS Implementation-Specific):		
GKS3D\$K_LINETYPE_DASH_2_DOT	–1	Line type dash-2-dots
GKS3D\$K_LINETYPE_DASH_3_DOT	–2	Line type dash-3-dots
GKS3D\$K_LINETYPE_LONG_DASH	–3	Line type long-dash
GKS3D\$K_LINETYPE_LONGSHORT_DASH	–4	Line type long-short-dash
GKS3D\$K_LINETYPE_SPACED_DASH	–5	Line type spaced-dash

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Line Types (DEC GKS Implementation-Specific):		
GKS3D\$K_LINETYPE_SPACED_DOT	-6	Line type spaced-dot
GKS3D\$K_LINETYPE_DOUBLE_DOT	-7	Line type double dots
GKS3D\$K_LINETYPE_TRIPLE_DOT	-8	Line type triple dots
Logical Types:		
GKS3D\$K_FALSE	0	Logical FALSE
GKS3D\$K_TRUE	1	Logical TRUE
Marker Types (Standard):		
GKS3D\$K_MARKTYPE_DOT	1	Marker type dot (.)
GKS3D\$K_MARKTYPE_PLUS	2	Marker type plus (+)
GKS3D\$K_MARKTYPE_ASTERISK	3	Marker type asterisk (*)
GKS3D\$K_MARKTYPE_CIRCLE	4	Marker type circle (O)
GKS3D\$K_MARKTYPE_DIAGONAL_CROSS	5	Marker type diagonal cross (X)
Marker Types (DEC GKS Implementation-Specific):		
GKS3D\$K_MARKTYPE_SOLID_CIRCLE	-1	Marker type solid circle
GKS3D\$K_MARKTYPE_TRIANGLE_UP	-2	Marker type hollow up triangle
GKS3D\$K_MARKTYPE_SOLID_TRI_UP	-3	Marker type solid up triangle
GKS3D\$K_MARKTYPE_TRIANGLE_DOWN	-4	Marker type hollow down triangle
GKS3D\$K_MARKTYPE_SOLID_TRI_DOWN	-5	Marker type solid down triangle
GKS3D\$K_MARKTYPE_SQUARE	-6	Marker type hollow square
GKS3D\$K_MARKTYPE_SOLID_SQUARE	-7	Marker type solid square
GKS3D\$K_MARKTYPE_BOWTIE	-8	Marker type hollow bow tie
GKS3D\$K_MARKTYPE_SOLID_BOWTIE	-9	Marker type solid bow tie
GKS3D\$K_MARKTYPE_HOURLASS	-10	Marker type hollow hourglass
GKS3D\$K_MARKTYPE_SOLID_HGLASS	-11	Marker type solid hourglass
GKS3D\$K_MARKTYPE_DIAMOND	-12	Marker type hollow diamond
GKS3D\$K_MARKTYPE_SOLID_DIAMOND	-13	Marker type solid diamond
New Frame Action Necessary States:		
GKS3D\$K_NEWFRAME_NOTNECESSARY	0	No new frame action on update
GKS3D\$K_NEWFRAME_NECESSARY	1	New frame action on update

(continued on next page)

Constants

Table B–1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Pick Status Types:		
GKS3D\$K_STATUS_NONE	0	No input obtained
GKS3D\$K_STATUS_OK	1	Input obtained
GKS3D\$K_STATUS_NOPICK	2	No pick input
Projection Types:		
GKS3D\$K_PARALLEL_PROJECTION	0	Parallel projection type
GKS3D\$K_PERSPECTIVE_PROJECTION	1	Perspective projection type
Regeneration Flag States:		
GKS3D\$K_PERFORM_FLAG	0	Implicit regeneration performed
GKS3D\$K_POSTPONE_FLAG	1	Implicit regeneration postponed
Returned Type Values:		
GKS3D\$K_VALUE_SET	0	Type of returned value set
GKS3D\$K_VALUE_REALIZED	1	Type of returned value realized
Simultaneous Events Flags:		
GKS3D\$K_NOMORE_EVENTS	0	No more simultaneously generated events
GKS3D\$K_MORE_EVENTS	1	More simultaneously generated events
Text Horizontal Alignment Types:		
GKS3D\$K_TEXT_HALIGN_NORMAL	0	Normal horizontal alignment
GKS3D\$K_TEXT_HALIGN_LEFT	1	Left horizontal alignment
GKS3D\$K_TEXT_HALIGN_CENTRE	2	Center horizontal alignment
GKS3D\$K_TEXT_HALIGN_RIGHT	3	Right horizontal alignment
Text Path Types:		
GKS3D\$K_TEXT_PATH_RIGHT	0	Text string reads from left to right.
GKS3D\$K_TEXT_PATH_LEFT	1	Text string reads from right to left.
GKS3D\$K_TEXT_PATH_UP	2	Text string reads from bottom to top.
GKS3D\$K_TEXT_PATH_DOWN	3	Text string reads from top to bottom.

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Text Precision Types:		
GKS3D\$K_TEXT_PRECISION_STRING	0	String precision. DEC GKS evaluates character height and width attributes only.
GKS3D\$K_TEXT_PRECISION_CHAR	1	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
GKS3D\$K_TEXT_PRECISION_STROKE	2	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
Text Vertical Alignment Types:		
GKS3D\$K_TEXT_VALIGN_NORMAL	0	Normal vertical alignment
GKS3D\$K_TEXT_VALIGN_TOP	1	Top vertical alignment
GKS3D\$K_TEXT_VALIGN_CAP	2	Cap vertical alignment
GKS3D\$K_TEXT_VALIGN_HALF	3	Half vertical alignment
GKS3D\$K_TEXT_VALIGN_BASE	4	Base vertical alignment
GKS3D\$K_TEXT_VALIGN_BOTTOM	5	Bottom vertical alignment
Update States:		
GKS3D\$K_NOTPENDING	0	Not pending
GKS3D\$K_PENDING	1	Pending
Viewport Priority States:		
GKS3D\$K_INPUT_PRIORITY_HIGHER	0	Relative input priority higher
GKS3D\$K_INPUT_PRIORITY_LOWER	1	Relative input priority lower
Visibility Flags:		
GKS3D\$K_VISIBLE	0	DEC GKS does not show the segment on the workstation surface.
GKS3D\$K_INVISIBLE	1	DEC GKS shows the segment on the workstation surface.
Workstation Category Types:		
GKS3D\$K_WSCAT_OUTPUT	0	Output
GKS3D\$K_WSCAT_INPUT	1	Input
GKS3D\$K_WSCAT_OUTIN	2	Output/input
GKS3D\$K_WSCAT_WISS	3	Workstation independent segment storage
GKS3D\$K_WSCAT_MO	4	Metafile output
GKS3D\$K_WSCAT_MI	5	Metafile input

(continued on next page)

Constants

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Workstation Class Types:		
GKS3D\$K_WSCLASS_VECTOR	0	Vector
GKS3D\$K_WSCLASS_RASTER	1	Raster
GKS3D\$K_WSCLASS_OTHERD	2	Other device
Workstation Color Availability States:		
GKS3D\$K_MONOCHROME	0	Monochrome
GKS3D\$K_COLOUR	1	Color
Workstation MI Types:		
GKS3D\$K_GKSM_METAFILE	0	DEC GKSM two-dimensional metafile
GKS3D\$K_GKS3_METAFILE	1	DEC GKS3 three-dimensional metafile
Workstation MO Types:		
GKS3D\$K_GKSM_MF_OUT	0	DEC GKSM two-dimensional metafile
GKS3D\$K_GKS3_MF_OUT	1	DEC GKS3 three-dimensional metafile
Workstation States:		
GKS3D\$K_WS_INACTIVE	0	Inactive
GKS3D\$K_WS_ACTIVE	1	Active
Workstation Types:		
GKS3D\$K_WSTYPE_DEFAULT	0	Default workstation type
GKS3D\$K_GKS3DM_OUTPUT	2	GKS three-dimensional output metafile
GKS3D\$K_GKS3DM_INPUT	3	GKS three-dimensional input metafile
GKS3D\$K_WSTYPE_WISS	5	GKS workstation independent segment storage
GKS3D\$K_CGM_OUTPUT	7	CGM output metafile
GKS3D\$K_VT_OUTPUT	10	Digital VT125 (output only)
GKS3D\$K_VT125BW_OUTPUT	10	Monochrome Digital VT125
GKS3D\$K_VT125	11	Digital VT125 with color option
GKS3D\$K_VT125BW	12	Monochrome Digital VT125
GKS3D\$K_VT240	13	Digital VT240 with color option
GKS3D\$K_VT240BW	14	Monochrome Digital VT240
GKS3D\$K_LCP01	15	Digital LCP01 printer
GKS3D\$K_LCG01	15	Digital LCG01 printer
GKS3D\$K_VT330	16	Digital VT330 (monochrome)
GKS3D\$K_VT340	17	Digital VT340 (color)

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Workstation Types:		
GKS3D\$K_LA34	31	Digital LA34 with graphics option
GKS3D\$K_LA100	31	Digital LA100
GKS3D\$K_LA50	32	Digital LA50 with 2:1 aspect ratio
GKS3D\$K_LA210	34	Digital LA210
GKS3D\$K_LA75	35	Digital LA75
GKS3D\$K_LASER_2200	37	DEClaser 2200 printer
GKS3D\$K_LNO3_PLUS	38	Digital LN03 PLUS
GKS3D\$K_LASER_2100	37	DEClaser 2100 printer
GKS3D\$K_VAXSTATION_VWS	41	Digital VAXstation II
GKS3D\$K_VSII	41	Digital VAXstation II
GKS3D\$K_VSII_GPX	41	Digital VAXstation II/GPX
GKS3D\$K_VS2000	41	Digital VAXstation 2000
GKS3D\$K_VS3200	41	Digital VAXstation 3200
GKS3D\$K_VS3500	41	Digital VAXstation 3500
GKS3D\$K_LVP16A	51	Digital LVP16 color plotter (8.5×11 inches)
GKS3D\$K_HP7475	51	Hewlett-Packard HP7475
GKS3D\$K_LVP16B	52	Digital LVP16 color plotter (11×17 inches)
GKS3D\$K_HP7550	53	Hewlett-Packard HP7550 pen plotter
GKS3D\$K_HP7580	54	Hewlett-Packard HP7580 pen plotter
GKS3D\$K_LG_MPS2000	55	LaserGraphics film recorder
GKS3D\$K_HP7585	56	Hewlett-Packard HP7585 pen plotter
GKS3D\$K_POSTSCRIPT	61	PostScript graphics handler
GKS3D\$K_COLOUR_POSTSCRIPT	62	Color PostScript graphics handler
GKS3D\$K_EPSF	65	Encapsulated PostScript
GKS3D\$K_COLOUR_EPSF	66	Color Encapsulated PostScript
GKS3D\$K_TEK4014_OUTPUT	70	Tektronix-4014 (output only)
GKS3D\$K_TEK4014	72	Tektronix-4014
GKS3D\$K_TEK4107_OUTPUT	80	Tektronix-4107 (output only)
GKS3D\$K_TEK4107	82	Tektronix-4107
GKS3D\$K_TEK4207_OUTPUT	83	Tektronix 4207 (output only)
GKS3D\$K_TEK4207	84	Tektronix 4207
GKS3D\$K_TEK4128_OUTPUT	85	Tektronix 4128 (output only)
GKS3D\$K_TEK4128	86	Tektronix 4128
GKS3D\$K_TEKVS500_OUTPUT	87	VS500 (output only)
GKS3D\$K_VS500_OUTPUT	87	VS500 (output only)

(continued on next page)

Constants

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
Workstation Types:		
GKS3D\$K_TEK4129_OUTPUT	87	Tektronix 4129 (output only)
GKS3D\$K_TEK4129	88	Tektronix 4129
GKS3D\$K_TEKVS500	88	VS500 interactive
GKS3D\$K_VS500	88	VS500 interactive
GKS3D\$K_LJ250	91	Digital LJ250 90 DPI
GKS3D\$K_LJ250_180DPI	92	Digital LJ250 180 DPI
GKS3D\$K_LA324	93	Digital LJ250 ink jet and LA324 printer
GKS3D\$K_DECWINDOWS_OUTPUT	210	DECwindows output
GKS3D\$K_DECWINDOWS	211	DECwindows
GKS3D\$K_DECWINDOWS_DRAWABLE	212	DECwindows drawable
GKS3D\$K_DECWINDOWS_WIDGET	213	DECwindows widget
GKS3D\$K_PEX_OUTPUT	220	PEX output
GKS3D\$K_VS3520_OUTPUT	220	VAXstation 3520 output
GKS3D\$K_VS3540_OUTPUT	220	VAXstation 3540 output
GKS3D\$K_PEX	221	PEX
GKS3D\$K_VS3520	221	VAXstation 3520
GKS3D\$K_VS3540	221	VAXstation 3540
GKS3D\$K_PEX_DRAWABLE	222	PEX drawable
GKS3D\$K_VS3520_DRAWABLE	222	VAXstation 3520 drawable
GKS3D\$K_VS3540_DRAWABLE	222	VAXstation 3540 drawable
GKS3D\$K_PEX_WIDGET	223	PEX widget
GKS3D\$K_VS3520_WIDGET	223	VAXstation 3520 widget
GKS3D\$K_VS3540_WIDGET	223	VAXstation 3540 widget
GKS3D\$K_MOTIF_OUTPUT	230	Motif output
GKS3D\$K_MOTIF	231	Motif
GKS3D\$K_MOTIF_DRAWABLE	232	Motif drawable
GKS3D\$K_MOTIF_WIDGET	233	Motif widget
GKS3D\$K_MOTIF_PEX_OUTPUT	240	OSF/Motif® PEX (output only)
GKS3D\$K_MOTIF_PEX	241	OSF/Motif PEX output and input
GKS3D\$K_MOTIF_PEX_DRAWABLE	242	OSF/Motif PEX drawable
GKS3D\$K_MOTIF_PEX_WIDGET	243	OSF/Motif PEX widget
GKS3D\$K_DDIF	250	DDIF
Writing Modes:		
GKS3D\$K_WRT_MODE_COMPLEMENT	2	Complement writing mode
GKS3D\$K_WRT_MODE_ERASE	3	Erase writing mode
GKS3D\$K_WRT_MODE_OVERLAY	4	Overlay writing mode

(continued on next page)

Table B-1 (Cont.) GKS3D\$ Constants

Constant	Value	Description
WSTYPE/CONID Association Modes:		
GKS3D\$K_WC_ASSOC_CANCEL	0	Cancel wstype/conid association
GKS3D\$K_WC_ASSOC_SET	1	Set wstype/conid association

Program Examples

Insert tabbed divider here. Then discard this sheet.



Program Examples

This appendix provides information on binding-specific code examples contained in this manual.

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
INQUIRE DISPLAY SPACE SIZE	Example 7-4

(continued on next page)

Program Examples

Table C-1 (Cont.) Binding-Specific Code Examples

Function	Example
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 ERROR HANDLER	Example 12-1
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

Language-Specific

Programming Information

Insert tabbed divider here. Then discard this sheet.



Language-Specific Programming Information

This appendix contains information specific to the DEC GKS supported languages. For a general overview of DEC GKS programming information (such as call sequences, including definition files, and so on), see Chapter 1.

Note

When you use languages that need to declare DEC GKS functions as external functions, you should check the language definition file to determine the function's parameter names. The various language definition files are described in Chapter 1.

D.1 Passing Arguments by Descriptor

DEC GKS requires array descriptors of class A or NCA, which include a bounds block for two-dimensional arrays. Array descriptors of class NCA must be contiguous. You can find defined macros for building array descriptors in C in the `descrip.h` file. See Example 5-1 for an example of how to use the macros contained in this file.

For languages that do not provide methods of creating such array descriptors, you can construct your own descriptor according to the specifications in the *Introduction to VMS System Routines*. If you choose, you can use the `BUILDESC` routine described in Section D.4 to build the required descriptor.

The following is a list of DEC GKS functions that require arguments passed by array descriptor:

- CELL ARRAY
- CELL ARRAY 3
- ERROR HANDLING
- ERROR LOGGING
- GET STRING
- GET STROKE
- GET STROKE 3
- INITIALIZE STRING
- INITIALIZE STRING 3
- INQUIRE POLYMARKER FACILITIES
- INQUIRE PREDEFINED PATTERN REPRESENTATION
- INQUIRE STRING DEVICE STATE

Language-Specific Programming Information

D.1 Passing Arguments by Descriptor

- MESSAGE
- OPEN GKS
- OPEN WORKSTATION
- REQUEST STROKE
- SAMPLE STRING
- SAMPLE STROKE
- SET PATTERN REPRESENTATION
- TEXT
- TEXT 3

D.2 Programming in BASIC

When you declare string variables to be passed to DEC GKS functions as write-only or modifiable arguments, you must declare the variable to be the length of the largest string that can be returned by the function. In addition, you should use the string length returned by the DEC GKS function instead of values obtained by the LEN built-in function to determine this size. For more information, see the *VAX BASIC Reference Manual*.

D.3 Programming in VAX C

To use the DEC GKS functions that require passing arguments by descriptor, you must build an array descriptor. To learn about building an array descriptor, see the *Introduction to VMS System Routines*. For VAX C specific information concerning descriptors, see the mixed-language programming chapter in *Guide to VAX C*. As another option, you can use the BUILDDESC routine described in Section D.4. Section D.1 lists the DEC GKS functions that require passing arguments by descriptor.

D.4 Programming in VAX COBOL

VAX COBOL variables passed to DEC GKS as integers, real numbers, or character strings must be declared in working storage as, respectively, COMPUTATIONAL, COMPUTATIONAL-1, or DISPLAY to obtain the correct internal representation. COMPUTATIONAL variables up to S9(9) are represented internally as 32-bit words. COMP-1 variables are represented in single-precision floating point format. DISPLAY character strings can be any length.

Integer and real numeric arguments to DEC GKS functions are passed by reference. Character or text strings are passed by descriptor.

The current VAX COBOL compiler does not produce class A array descriptors. However, certain DEC GKS functions require these descriptors. See Section D.1 for a list of the DEC GKS functions that require arrays passed by descriptor.

The following MACRO subroutine, named BUILDDESC, can serve as a temporary tool to allow VAX COBOL programs that use the above functions to generate class A array descriptors. The subroutine is needed only for programs that call any of the functions listed in Section D.1. Example D-1 shows how to build a descriptor.

Language-Specific Programming Information

D.4 Programming in VAX COBOL

Example D-1 Macro Subroutine Used to Build Array Descriptors

```
.TITLE BUILDESC Subroutine to build VMS array descriptor
.IDENT /01/
.ENTRY BUILDESC, ^M<R2>
$SSDEF          ; Define SS$ symbols
$DSCDEF         ; Define DSC$ symbols
;
; Fill in first two longwords of descriptor
;
    MOVL      8(AP), R0
    MOVL      4(AP), R1
    MOVW     DSC$W_LENGTH(R0), DSC$W_LENGTH(R1)
    MOVB     DSC$B_DTYPE(R0), DSC$B_DTYPE(R1)
    MOVB     #DSC$K_CLASS_A, DSC$B_CLASS(R1)
    MOVL     DSC$A_POINTER(R0), DSC$A_POINTER(R1)
;
; Fill in Block 1 - Prototype
;
    CLRB     DSC$B_SCALE(R1)
    CLRB     DSC$B_DIGITS(R1)
    MOVB     -
    #<<1@DSC$V_FL_COEFF>!<1@DSC$V_FL_BOUNDS>>, DSC$B_AFLAGS(R1)
    SUBB3    #2, (AP), DSC$B_DIMCT(R1)
    MOVL     12(AP), DSC$L_ARSIZE(R1)
    MOVL     #1, R0
    MOVL     #4, R2
    CMPB     #1, DSC$B_DIMCT(R1)
    BEQL     10$
    MULL2    16(AP), DSC$L_ARSIZE(R1)
    ADDL2    16(AP), R0
    INCL     R0
;
; Fill Blocks 2 and 3 (Multipliers, Bounds) for 2nd dim. (if present)
;
    MOVL     16(AP), DSC$L_M2(R1)
    MOVL     #1, DSC$L_M2+12(R1)
    MOVL     16(AP), DSC$L_M2+16(R1)
    ADDL2    #4, R2
;
; Fill in Blocks 2 (Multipliers) and 3 (Bounds) for 1st dimension
;
10$:  MULW2    DSC$W_LENGTH(R1), R0
    SUBL3    R0, DSC$A_POINTER(R1), DSC$A_A0(R1)
    MOVL     12(AP), DSC$L_M1(R1)
    ADDL2    R2, R1
    MOVL     #1, DSC$L_M1(R1)
    MOVL     12(AP), DSC$L_M1+4(R1)
    MOVZWL   #SS$_NORMAL, R0
    RET
.END
```

The subroutine builds an array descriptor from the arguments it is passed. For information on descriptor formats, see the VAX Procedure Calling and Condition Handling Standard in the *VMS Run-Time Library Routines Reference Manual*.

You can use MACRO to assemble the subroutine and then call it from the VAX COBOL program. The following is a sample VAX COBOL calling sequence for two-dimensional arrays (assuming BUILDESC as the subroutine name):

Language-Specific Programming Information

D.4 Programming in VAX COBOL

```
CALL "BUILDESC" USING
BY REFERENCE descriptor-buffer,
BY DESCRIPTOR array(1,1),
BY VALUE number-of-rows,
BY VALUE number-of-columns.
```

For a one-dimensional array, the COBOL calling sequence is as follows:

```
CALL "BUILDESC" USING
BY REFERENCE descriptor-buffer,
BY DESCRIPTOR array(1),
BY VALUE number-of-elements.
```

The descriptor buffer is an area of storage into which BUILDESC builds the class A descriptor. This should be at least 44 bytes in length. The descriptor buffer is filled with the information required to make it a class A descriptor.

The argument *array(1, 1)* should always be the first element of the array.

Example D-2 shows a COBOL program using the function CELL ARRAY.

Example D-2 A Sample COBOL Program Using the Subroutine BUILDESC

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                C09.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.           VAX-11.
OBJECT-COMPUTER.           VAX-11.
DATA DIVISION.
WORKING-STORAGE SECTION.
01    valthree PIC S9(9)    COMP VALUE 3.
01    valfour  PIC S9(9)
COMP VALUE 4.
01    valone   PIC S9(9)
COMP VALUE 1.
01    valzero  PIC S9(9)
COMP VALUE 0.
01    valpoint1 USAGE IS COMP-1 VALUE 0.1.
01    valpoint5 USAGE IS COMP-1 VALUE 0.5.
01    colidx.
      05 dim1    OCCURS 3 TIMES.
      /n        10 colia OCCURS 4 TIMES
PIC S9(9) COMP.
01    colidx_d.
      05 desc OCCURS 11 TIMES PIC S9(9) COMP.

PROCEDURE DIVISION.
0000-COB9.
      MOVE 1 TO colia(1,1).
      MOVE 0 TO colia(1,2).
      MOVE 1 TO colia(1,3).
      MOVE 2 TO colia(1,4).
      MOVE 0 TO colia(2,1).
      MOVE 1 TO colia(2,2).
      MOVE 2 TO colia(2,3).
      MOVE 1 TO colia(2,4).
      MOVE 1 TO colia(3,1).
      MOVE 2 TO colia(3,2).
      MOVE 1 TO colia(3,3).
      MOVE 0 TO colia(3,4).
```

(continued on next page)

Language-Specific Programming Information

D.4 Programming in VAX COBOL

Example D-2 (Cont.) A Sample COBOL Program Using the Subroutine BUILDESC

```
CALL "GKS$OPEN_GKS" USING
    BY DESCRIPTOR 'GKS.ERR'.
CALL "GKS$OPEN_WS" USING
    BY REFERENCE valone, valzero, valzero.
CALL "GKS$ACTIVATE_WS" USING
    BY REFERENCE valone.
CALL "BUILDESC" USING
    BY REFERENCE colidx_d,
    BY DESCRIPTOR colia(1,1),
    BY VALUE valthree, valfour.
CALL "GKS$CELL_ARRAY" USING
    BY REFERENCE valpoint1, valpoint1, valpoint5, valpoint5,
    BY REFERENCE valone, valone,
    BY REFERENCE valthree, valfour,
    BY REFERENCE colidx_d.
CALL "GKS$DEACTIVATE_WS" USING
    BY REFERENCE valone.
CALL "GKS$CLOSE_WS" USING
    BY REFERENCE valone.
CALL "GKS$CLOSE_GKS".
EXIT PROGRAM.
END PROGRAM C09.
```

To use the subroutine, type it in, assemble it, compile your VAX COBOL program that calls the subroutine, and then link the VAX COBOL program with the subroutine, as follows:

```
$ MACRO BUILDESC [RETURN]
$ COBOL ARRAY [RETURN]
$ LINK ARRAY, BUILDESC [RETURN]
```

DEC GKS calls can be written with or without a status return. When used, the status code is defined as PIC S9(6) COMP, which yields a 32-bit integer internal representation.

D.5 Programming in VAX Pascal

DEC GKS functions called from a VAX Pascal program must be declared as external functions in the program. The variables passed to these functions and the way they are to be passed must also be described, and the type of the return specified. To gather these declarations, perform the following tasks:

1. Copy SYS\$LIBRARY:GKSDEFS.PAS to your local directory.
2. Use the following command to compile the file:

```
$ PASCAL/ENVIRONMENT GKSDEFS.PAS [RETURN]
```

3. Place the following code before the PROGRAM or MODULE statement:

```
[INHERIT ('gksdefs')]
```

Variables passed to DEC GKS by a VAX Pascal program must be declared as types INTEGER, REAL, or an array of these types. Metafile items are declared as packed arrays of characters because the length of a metafile item may exceed the allowable length for a variable length string. Data records for the input functions are declared as arrays of integers. Where a REAL data item is called for in a data record, the type cast operator must be used to force the variable

Language-Specific Programming Information

D.5 Programming in VAX Pascal

to be placed properly. Addresses for data records may be generated using the ADDRESS function and the type cast operator to override the type of integer.

Character strings are declared as VARYING OF CHAR. When you declare string variables to be passed to DEC GKS functions as write-only or modifiable arguments, you must declare the variable to be the length of the largest string that can be returned by the function. In addition, you should use the string length returned by the DEC GKS function instead of values obtained by the LEN built-in function to determine this size. Strings should be padded with spaces to their greatest length using the VAX Pascal PAD function. For more information, see the *VAX Pascal Language Reference Manual*.

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	GKS3D\$K_LINETYPE_ SOLID	Solid line
Line width	1.0	Minimum width
Line color index	1	Workstation dependent value
Line type ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current line type
Line width ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current line width
Line color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current line color index

E.2 Initial Polymarker Attributes

This section lists the initial values for the polymarker attributes.

Initial Attribute Values

E.2 Initial Polymarker Attributes

Attribute	Initial Value	Description
Polymarker index	1	Polymarker bundle number 1
Marker type	GKS3D\$K_MARKERTYPE_ ASTERISK	Asterisk for marker
Marker size	1.0	Nominal size
Color index	1	Workstation dependent value
Marker type ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current marker type
Marker size ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current marker size
Marker color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	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 GKS3D\$K_TEXT_PRECISION_ 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	GKS3D\$K_ASF_ INDIVIDUAL	Use current font and precision
Character expansion factor ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current width and height ratio
Character spacing ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current character space
Text color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	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	GKS3D\$K_TEXT_PATH_RIGHT	Right angle clockwise from up vector
Text alignment	GKS3D\$K_TEXT_HALIGN_ NORMAL GKS3D\$K_TEXT_VALIGN_ 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.

Attribute	Initial Value	Description
Fill area index	1	Fill area bundle number 1
Interior style	GKS3D\$K_INTSTYLE_ HOLLOW	Boundary of polygonal area
Style index	1	Workstation-dependent pattern or hatch style
Color index	1	Workstation-dependent value
Interior style ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current interior style
Style index ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current pattern or hatch style
Color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	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.

Attribute	Initial Value	Description
Fill area index	1	Fill area bundle number 1
Interior style	GKS3D\$K_INTSTYLE_ HOLLOW	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	GKS3D\$K_ASF_ INDIVIDUAL	Use current interior style
Fill area style index ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current pattern or hatch style
Fill area color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	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	GKS3D\$K_EDGE_ FLAG_OFF	Edge not drawn
Edge type	GKS3D\$K_LINETYPE_ SOLID	Solid edge

Initial Attribute Values

E.5 Initial Fill Area Set Attributes

Attribute	Initial Value	Description
Edge width scale factor	1.0	Minimum width
Edge color index	1	Workstation-dependent value
Edge flag ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current edge flag
Edge type ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current edge type
Edge width scale factor ASF	GKS3D\$K_ASF_ INDIVIDUAL	Use current edge width scale factor
Edge color index ASF	GKS3D\$K_ASF_ INDIVIDUAL	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	GKS3D\$K_ VISIBLE	The segment is visible.
Highlighting	GKS3D\$K_ NORMAL	The segment is not highlighted.
Segment priority	0.0	The segment has the lowest priority.
Detectability	GKS3D\$K_ UNDETECTABLE	The segment is undetectable.

The default segment transformation is called the identity transformation. The identity transformation uses a 3×4 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 (GKS3D\$K_CLIP) 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
Functional Scope	
GKS level	DEC GKS is level 2c.
Capacity	
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
Capacity	
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.
Miscellaneous	
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

- Access type, *Part 1*, 1–6
- ACCUMULATE TRANSFORMATION MATRIX 3
 - function, *Part 1*, 7–12
- ACCUMULATE TRANSFORMATION MATRIX 3
 - function, *Part 1*, 7–10
 - example, *Part 1*, 7–36
- Accumulating
 - segment transformations, *Part 1*, 8–9
- Action pending states
 - list of, *Part 2*, B–13
- ACTIVATE WORKSTATION function, *Part 1*, 4–9
 - example, *Part 1*, 4–27
- 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
- Applications
 - programming information, *Part 2*, D–1
- Arguments
 - characteristics of, *Part 1*, 1–6
 - descriptions, *Part 1*, 1–6
 - inquiry error status, *Part 2*, 11–3
 - inquiry value type argument, *Part 2*, 11–4
 - passing by descriptor, *Part 2*, D–1
- Arrays
 - color index, *Part 1*, 5–4, 5–6
 - descriptors, *Part 2*, D–1
- ASAP, *Part 1*, 1–7
- ASFs, *Part 1*, 6–4
- Aspect ratio
 - See also Transformations
- Aspect source flags
 - list of, *Part 2*, B–1
- 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 control flags
 - list of, *Part 2*, B–2
- Attribute functions, *Part 1*, 6–1 to 6–72
 - 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
 - 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*, 9–87
- 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
- BASIC programming information, *Part 2*, D–2
- Binding
 - attributes to primitives, *Part 1*, 6–1
 - list of GKS3D\$ constants, *Part 2*, B–1 to B–17

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
 color, *Part 1*, 6–16
 edge, *Part 1*, 6–21
 fill area, *Part 1*, 6–26, 6–29
 pattern styles, *Part 1*, 6–40
 polyline, *Part 1*, 6–44, 6–45
 polymarker, *Part 1*, 6–48, 6–49
 text, *Part 1*, 6–56, 6–59

C

Calling sequences, *Part 2*, D–2
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
CELL ARRAY 3 function, *Part 1*, 5–6
CELL ARRAY function, *Part 1*, 5–4
 example, *Part 1*, 5–23
Cell arrays, *Part 1*, 5–4, 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*, 5–21, 5–22
Choice
 See also Input
 input class, *Part 1*, 9–4
 specifying NOCHOICE input, *Part 1*, 9–16
Choice input prompt flags
 list of, *Part 2*, B–2
Choice status types
 list of, *Part 2*, B–2
Classes
 See also Input
 See also Logical input devices
 choice, *Part 1*, 9–4
 locator, *Part 1*, 9–4
Classes (cont'd)
 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 screen states
 list of, *Part 2*, B–2
CLEAR WORKSTATION function, *Part 1*, 4–10
 example, *Part 1*, 4–27
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
Clipping flags
 list of, *Part 2*, B–2
CLOSE GKS function, *Part 1*, 4–11
 example, *Part 1*, 4–27
CLOSE SEGMENT function, *Part 1*, 8–12
 example, *Part 1*, 8–28
CLOSE WORKSTATION function, *Part 1*, 4–12
 example, *Part 1*, 4–27
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
COBOL programming information, *Part 2*, D–2 to D–5
Color
 See also Attributes
 background, *Part 1*, 6–5
 fill area, *Part 1*, 6–25
 foreground, *Part 1*, 6–5
 markers, *Part 1*, 6–47
 model, *Part 1*, 6–15
 polyline, *Part 1*, 6–43
 representation, *Part 1*, 6–16
 text, *Part 1*, 6–53
Color availability flags
 list of, *Part 2*, B–2
Color models
 list of, *Part 2*, B–2
Colors
 indexes
 2D arrays, *Part 1*, 5–4

- Colors
 - indexes (cont'd)
 - 3D arrays, *Part 1*, 5–6
- Color validity states
 - list of, *Part 2*, B–2
- Compiling
 - ULTRIX programs, *Part 1*, 3–2
 - 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 to 3–8
 - customizing
 - system level, *Part 1*, 3–7
 - user level, *Part 1*, 3–8
- Connection identifier
 - default value, *Part 2*, B–3
- Connection identifiers, *Part 1*, 4–21
 - metafiles, *Part 1*, 10–1, 10–4
 - specifying on ULTRIX, *Part 1*, 3–3
 - specifying on VMS, *Part 1*, 2–2
- Constants
 - action pending states, *Part 2*, B–13
 - aspect source flags, *Part 2*, B–1
 - attribute control flags, *Part 2*, B–2
 - choice input prompt flags, *Part 2*, B–2
 - choice status types, *Part 2*, B–2
 - clear screen states, *Part 2*, B–2
 - clipping flags, *Part 2*, B–2
 - color availability flags, *Part 2*, B–2
 - color models, *Part 2*, B–2
 - color validity states, *Part 2*, B–2
 - coordinate switch, *Part 2*, B–3
 - data type identifiers, *Part 2*, B–3
 - default connection identifier, *Part 2*, B–3
 - deferral modes, *Part 2*, B–3
 - detectability flags, *Part 2*, B–3
 - device coordinate units, *Part 2*, B–3
 - display surface states, *Part 2*, B–3
 - dynamic modification states, *Part 2*, B–4
 - echo states, *Part 2*, B–4
 - edge flags, *Part 2*, B–4
 - edge types, *Part 2*, B–4
 - error handling modes, *Part 2*, B–4
 - escape identifiers, *Part 2*, B–4
 - fill area hatch styles, *Part 2*, B–6
 - fill area interior styles, *Part 2*, B–6
 - GDP bundle types, *Part 2*, B–7
 - GDPs, *Part 2*, B–7
 - GKS3D\$ binding, *Part 2*, B–1 to B–17
- Constants (cont'd)
 - GKS level types, *Part 2*, B–8
 - GKS operating states, *Part 2*, B–8
 - highlighting flags, *Part 2*, B–9
 - highlighting methods, *Part 2*, B–9
 - HLHSR identifiers, *Part 2*, B–9
 - HLHSR modes, *Part 2*, B–9
 - horizontal alignment types, *Part 2*, B–12
 - implicit regeneration states, *Part 2*, B–9
 - input classes, *Part 2*, B–9
 - input mode types, *Part 2*, B–10
 - input priority states, *Part 2*, B–10
 - input status types, *Part 2*, B–10
 - line cap styles, *Part 2*, B–10
 - line join styles, *Part 2*, B–10
 - line types (DEC GKS implementation-specific), *Part 2*, B–10
 - line types (standard), *Part 2*, B–10
 - list of, *Part 2*, B–1 to B–17
 - logical types, *Part 2*, B–11
 - marker types (DEC GKS implementation-specific), *Part 2*, B–11
 - marker types (standard), *Part 2*, B–11
 - new frame action necessary states, *Part 2*, B–11
 - pick status types, *Part 2*, B–12
 - projection types, *Part 2*, B–12
 - regeneration flag states, *Part 2*, B–12
 - requirements, *Part 1*, 2–1, 3–1
 - returned type values, *Part 2*, B–12
 - simultaneous events flags, *Part 2*, B–12
 - text horizontal alignment types, *Part 2*, B–12
 - text path types, *Part 2*, B–12
 - text precision types, *Part 2*, B–13
 - text vertical alignment types, *Part 2*, B–13
 - update states, *Part 2*, B–13
 - vertical alignment types, *Part 2*, B–13
 - viewport priority states, *Part 2*, B–13
 - visibility flags, *Part 2*, B–13
 - workstation category types, *Part 2*, B–13
 - workstation class types, *Part 2*, B–14
 - workstation color availability states, *Part 2*, B–14
 - workstation MI types, *Part 2*, B–14
 - workstation MO types, *Part 2*, B–14
 - workstation states, *Part 2*, B–14
 - workstation types, *Part 2*, B–14
 - writing modes, *Part 2*, B–16
 - WSTYPE/CONID association modes, *Part 2*, B–17
- Control
 - error handling, *Part 2*, 12–1
 - workstation surface, *Part 1*, 4–6
- Control functions, *Part 1*, 4–1 to 4–35
 - 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, 9–19

Coordinate switch
 list of, *Part 2*, B–3

Copying segments, *Part 1*, 8–3

COPY SEGMENT TO WORKSTATION function,
Part 1, 8–13
 example, *Part 1*, 8–28

C programming information, *Part 2*, D–2

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*, 9–15
 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
 escape
 standard, *Part 1*, 4–16
 input, *Part 1*, 9–8
 prompt and echo types, *Part 1*, 9–5 to 9–14
 sizes, *Part 1*, 9–20
 standard, *Part 1*, 9–8
 using inquiry functions, *Part 1*, 9–20
 metafile
 item, *Part 1*, 10–2

Data structures
 See also GKS

Data type identifiers
 list of, *Part 2*, B–3

Data types
 arguments, *Part 1*, 1–6

DEACTIVATE WORKSTATION function, *Part 1*,
 4–13
 example, *Part 1*, 4–27

Deactivating
 See also Workstations
 workstations, *Part 1*, 4–5

Declaring
 GKS functions
 externally, *Part 2*, D–1

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*, 1–7
 output, *Part 1*, 4–6, 5–3

Deferral mode
 GKS3D\$K_ASAP, *Part 1*, 4–23

Deferral modes
 list of, *Part 2*, B–3

Definition files
 declaring external functions, *Part 2*, D–1
 including, *Part 1*, 2–1, 3–1
 list of, *Part 1*, 2–1; *Part 2*, B–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

Descriptors
 passing arguments, *Part 2*, D–1
 problems passing, *Part 2*, D–2 to D–5

Detectability flags
 list of, *Part 2*, B–3

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 coordinate units
 - list of, *Part 2*, B-3
- 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
 - connection, *Part 1*, 4-21
 - logical input, *Part 1*, 9-1 to 9-3
 - manipulation
 - GKS3D\$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*, 4-10, 4-22, 4-23, 4-25, 7-1
- Display surface states
 - list of, *Part 2*, B-3
- Dynamic modification
 - See also Implicit regenerations
 - attributes, *Part 1*, 4-7
 - workstation transformations, *Part 1*, 4-7
- Dynamic modification states
 - list of, *Part 2*, B-4

E

- Echo
 - See also Input
 - cycling and disabled echo, *Part 1*, 9-15
 - input values, *Part 1*, 9-2, 9-3, 9-14
 - prompt and echo types, *Part 1*, 9-5 to 9-14
- Echo area, *Part 1*, 9-2
- Echo states
 - list of, *Part 2*, B-4
- Edge
 - index, *Part 1*, 6-20
 - representation, *Part 1*, 6-21
 - type, *Part 1*, 6-23
 - width scale factor, *Part 1*, 6-24
- Edge flags
 - list of, *Part 2*, B-4
- Edge types
 - list of, *Part 2*, B-4
- Emergency
 - closure of GKS, *Part 2*, 12-1
- EMERGENCY CLOSE GKS function, *Part 2*, 12-3
 - example, *Part 2*, 12-7

- Ending
 - GKS session, *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 csh, *Part 1*, 3-4
 - at sh, *Part 1*, 3-4
 - in file, *Part 1*, 3-4
 - GKSsaf, *Part 1*, 3-5
 - GKSconid, *Part 1*, 3-3, 3-5
 - .GKSdefaults, *Part 1*, 3-4
 - GKSdefmode, *Part 1*, 3-5
 - GKSerrfile, *Part 1*, 3-6, 3-7
 - GKSError, *Part 1*, 3-6
 - GKSirg, *Part 1*, 3-6
 - GKSmetafile_type, *Part 1*, 3-6
 - GKSndc_clip, *Part 1*, 3-6
 - GKSstroke_font1, *Part 1*, 3-6
 - GKSwstype, *Part 1*, 3-6
 - search order, *Part 1*, 3-4
 - stderr, *Part 1*, 3-7
 - system defaults file, *Part 1*, 3-3
 - types, *Part 1*, 3-5
 - general, *Part 1*, 3-5
 - user defaults file, *Part 1*, 3-3
- Error codes
 - defined, *Part 1*, 2-5, 3-6
 - ULTRIX, *Part 1*, 3-6
 - VMS, *Part 1*, 2-5
- Error files
 - default, *Part 1*, 2-5
 - defined, *Part 1*, 3-7
 - ULTRIX, *Part 1*, 3-7
 - VMS, *Part 1*, 2-5
- Error handling, *Part 1*, 2-4, 2-5, 3-6 to 3-7
 - GKS, *Part 1*, 1-4
- ERROR HANDLING function, *Part 2*, 12-4
- Error-handling functions, *Part 2*, 12-1 to 12-8
 - introduction to, *Part 2*, 12-1 to 12-2
- Error handling modes
 - list of, *Part 2*, B-4
- ERROR LOGGING function, *Part 2*, 12-5
 - example, *Part 2*, 12-7
- Errors
 - 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
 - escapes, *Part 2*, A-15
 - fatal, *Part 2*, A-36 to A-37

Errors

- messages (cont'd)
 - implementation-specific, *Part 2*, A-19 to A-36
 - 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*, 2-2, 3-1
- ESCAPE function, *Part 1*, 4-14
 - example, *Part 1*, 4-29
- Escape identifiers
 - list of, *Part 2*, B-4
- Escapes
 - data records, *Part 1*, 4-16
 - GKS3D\$ESCAPE, *Part 1*, 4-14
 - 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*, 7-40
- EVALUATE VIEW MAPPING MATRIX 3 function,
 - Part 1*, 7-18
- EVALUATE VIEW ORIENTATION MATRIX 3
 - function, *Part 1*, 7-20
- Event functions, *Part 1*, 9-17
- Event input queue, *Part 1*, 9-17
 - overflow, *Part 1*, 9-18
- Event mode, *Part 1*, 9-17 to 9-19
 - 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*, 3-2
 - 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

External functions

- declaring GKS functions, *Part 2*, D-1

F

- Fatal errors, *Part 2*, 12-1
 - list of, *Part 2*, A-36 to A-37
- Files
 - definition, *Part 2*, B-1
 - list of, *Part 1*, 2-1
 - error, *Part 2*, 12-2
 - error status
 - list of, *Part 1*, 2-2, 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*, 6-61
- Fill area hatch styles
 - list of, *Part 2*, B-6
- Fill area interior styles
 - list of, *Part 2*, B-6
- Fill areas, *Part 1*, 5-8
 - See also Attributes
 - See also gks3d\$fill_area_set3 function
 - See also gks3d\$fill_area_set function
 - attributes
 - SET FILL AREA COLOUR INDEX, *Part 1*, 6-25
 - SET FILL AREA INDEX, *Part 1*, 6-26
 - 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*, 6-26
 - 3D, *Part 1*, 5-9
 - initial attributes, *Part 2*, E-3
 - interior styles, *Part 1*, 6-27
 - 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*, 6-19

Flush
 event queue, *Part 1*, 9–18
FLUSH DEVICE EVENTS, *Part 1*, 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
 description, *Part 1*, 1–6
 header, *Part 1*, 1–4
 operating states, *Part 1*, 1–5
 presentation, *Part 1*, 1–4 to 1–8
 Program Examples sections, *Part 1*, 1–7
 See Also sections, *Part 1*, 1–6
 syntax, *Part 1*, 1–5
Functional standards
 See also GKS
Functions
 See also GKS
 arguments passed by descriptor, *Part 2*, D–1
 attribute, *Part 1*, 6–6
 control, *Part 1*, 4–8
 DEC GKS categories, *Part 1*, 1–1
 declaring GKS functions, *Part 2*, D–1
 error-handling, *Part 2*, 12–1 to 12–2
 input, *Part 1*, 9–21
 metafile, *Part 1*, 10–5
 output, *Part 1*, 5–3
 segment, *Part 1*, 8–10
 transformation, *Part 1*, 7–9

G

GDP
 list of, *Part 2*, B–7
GDP bundle types
 list of, *Part 2*, B–7
GDPs, *Part 1*, 5–12 to 5–16
 attributes, *Part 1*, 6–3
GENERALIZED DRAWING PRIMITIVE 3
 function, *Part 1*, 5–15
GENERALIZED DRAWING PRIMITIVE function,
 Part 1, 5–12
 example, *Part 1*, 5–26
Generalized drawing primitives
 See also 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
GET CHOICE function, *Part 1*, 9–25
GET functions, *Part 1*, 9–17
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*, 9–87
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
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
 functions
 declared as external, *Part 2*, D–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*, 2–4
GKS\$CONID, *Part 1*, 2–4
GKS\$DEF_MODE, *Part 1*, 2–4
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
gks3d\$accum_xform_matrix3 function, *Part 1*,
 7–12
gks3d\$accum_xform_matrix function, *Part 1*, 7–10
gks3d\$activate_ws function, *Part 1*, 4–9
gks3d\$assoc_seg_with_ws function, *Part 1*, 8–11
gks3d\$await_event function, *Part 1*, 9–22
GKS3D\$ binding files
 ULTRIX, *Part 1*, 3–1
 VMS, *Part 1*, 2–1
gks3d\$cell_array3 function, *Part 1*, 5–6

gks3d\$cell_array function, *Part 1*, 5–4
gks3d\$clear_ws function, *Part 1*, 4–10
gks3d\$close_gks function, *Part 1*, 4–11
gks3d\$close_seg function, *Part 1*, 8–12
gks3d\$close_ws function, *Part 1*, 4–12
GKS3D\$ constants, *Part 2*, B–1 to B–17
gks3d\$copy_seg_to_ws function, *Part 1*, 8–13
gks3d\$create_seg function, *Part 1*, 8–14
gks3d\$deactivate_ws function, *Part 1*, 4–13
gks3d\$delete_seg function, *Part 1*, 8–15
gks3d\$delete_seg_from_ws function, *Part 1*, 8–16
gks3d\$emergency_close function, *Part 2*, 12–3
gks3d\$error_handler function, *Part 2*, 12–4
gks3d\$escape function, *Part 1*, 4–14
gks3d\$eval_view_map_matrix3 function, *Part 1*, 7–18
gks3d\$eval_view_orient_matrix3 function, *Part 1*, 7–20
gks3d\$eval_xform_matrix3 function, *Part 1*, 7–16
gks3d\$eval_xform_matrix function, *Part 1*, 7–14
gks3d\$fill_area3 function, *Part 1*, 5–9
gks3d\$fill_area function, *Part 1*, 5–8
gks3d\$fill_area_set3 function, *Part 1*, 5–11
gks3d\$fill_area_set function, *Part 1*, 5–10
gks3d\$flush_device_events function, *Part 1*, 9–24
gks3d\$gdp3 function, *Part 1*, 5–15
gks3d\$gdp function, *Part 1*, 5–12
gks3d\$get_choice function, *Part 1*, 9–25
gks3d\$get_item function, *Part 1*, 10–6
gks3d\$get_locator3 function, *Part 1*, 9–27
gks3d\$get_locator function, *Part 1*, 9–26
gks3d\$get_pick function, *Part 1*, 9–28
gks3d\$get_string function, *Part 1*, 9–29
gks3d\$get_stroke3 function, *Part 1*, 9–32
gks3d\$get_stroke function, *Part 1*, 9–30
gks3d\$get_valuator function, *Part 1*, 9–34
gks3d\$init_choice3 function, *Part 1*, 9–37
gks3d\$init_choice function, *Part 1*, 9–35
gks3d\$init_locator3 function, *Part 1*, 9–41
gks3d\$init_locator function, *Part 1*, 9–39
gks3d\$init_pick3 function, *Part 1*, 9–45
gks3d\$init_pick function, *Part 1*, 9–43
gks3d\$init_string3 function, *Part 1*, 9–49
gks3d\$init_string function, *Part 1*, 9–47
gks3d\$init_stroke3 function, *Part 1*, 9–53
gks3d\$init_stroke function, *Part 1*, 9–51
gks3d\$init_valuator3 function, *Part 1*, 9–57
gks3d\$init_valuator function, *Part 1*, 9–55
gks3d\$inq_active_ws function, *Part 2*, 11–126
gks3d\$inq_avail_gdp3 function, *Part 2*, 11–71
gks3d\$inq_avail_gdp function, *Part 2*, 11–70
gks3d\$inq_choice_state3 function, *Part 2*, 11–7
gks3d\$inq_choice_state function, *Part 2*, 11–5
gks3d\$inq_clip3 function, *Part 2*, 11–10
gks3d\$inq_clip function, *Part 2*, 11–9
gks3d\$inq_colour_fac function, *Part 2*, 11–11
gks3d\$inq_colour_indexes function, *Part 2*, 11–73
gks3d\$inq_colour_model function, *Part 2*, 11–12
gks3d\$inq_colour_model_fac function, *Part 2*, 11–13
gks3d\$inq_colour_rep function, *Part 2*, 11–14
gks3d\$inq_current_xformno function, *Part 2*, 11–22
gks3d\$inq_def_choice_data3 function, *Part 2*, 11–30
gks3d\$inq_def_choice_data function, *Part 2*, 11–28
gks3d\$inq_def_defer_state function, *Part 2*, 11–32
gks3d\$inq_def_locator_data3 function, *Part 2*, 11–34
gks3d\$inq_def_locator_data function, *Part 2*, 11–33
gks3d\$inq_def_pick_data3 function, *Part 2*, 11–37
gks3d\$inq_def_pick_data function, *Part 2*, 11–35
gks3d\$inq_def_string_data3 function, *Part 2*, 11–41
gks3d\$inq_def_string_data function, *Part 2*, 11–39
gks3d\$inq_def_stroke_data3 function, *Part 2*, 11–45
gks3d\$inq_def_stroke_data function, *Part 2*, 11–43
gks3d\$inq_def_valuator_data3 function, *Part 2*, 11–49
gks3d\$inq_def_valuator_data function, *Part 2*, 11–47
gks3d\$inq_dyn_mod_seg_atlb function, *Part 2*, 11–53
gks3d\$inq_dyn_mod_ws_atlb3 function, *Part 2*, 11–57
gks3d\$inq_dyn_mod_ws_atlb function, *Part 2*, 11–55
gks3d\$inq_edge_fac function, *Part 2*, 11–59
gks3d\$inq_edge_indexes function, *Part 2*, 11–74
gks3d\$inq_edge_rep function, *Part 2*, 11–60
gks3d\$inq_fill_fac function, *Part 2*, 11–62
gks3d\$inq_fill_indexes function, *Part 2*, 11–75
gks3d\$inq_fill_rep function, *Part 2*, 11–63
gks3d\$inq_gdp3 function, *Part 2*, 11–65
gks3d\$inq_gdp function, *Part 2*, 11–64
gks3d\$inq_hlshr_fac function, *Part 2*, 11–66
gks3d\$inq_hlshr_id function, *Part 2*, 11–15
gks3d\$inq_hlshr_mode function, *Part 2*, 11–67
gks3d\$inq_indiv_atlb3 function, *Part 2*, 11–19
gks3d\$inq_indiv_atlb function, *Part 2*, 11–16
gks3d\$inq_input_dev function, *Part 2*, 11–93
gks3d\$inq_input_queue_overflow function, *Part 2*, 11–68
gks3d\$inq_level function, *Part 2*, 11–69
gks3d\$inq_locator_state3 function, *Part 2*, 11–84
gks3d\$inq_locator_state function, *Part 2*, 11–82
gks3d\$inq_max_ds_size3 function, *Part 2*, 11–52
gks3d\$inq_max_ds_size function, *Part 2*, 11–51

gks3d\$inq_max_ws_state_table3 function, *Part 2*, 11–87
gks3d\$inq_max_ws_state_table function, *Part 2*, 11–86
gks3d\$inq_max_xform function, *Part 2*, 11–88
gks3d\$inq_more_simul_events function, *Part 2*, 11–89
gks3d\$inq_name_open_seg function, *Part 2*, 11–90
gks3d\$inq_open_ws function, *Part 2*, 11–128
gks3d\$inq_operating_state function, *Part 2*, 11–95
gks3d\$inq_pat_fac function, *Part 2*, 11–96
gks3d\$inq_pat_indexes function, *Part 2*, 11–77
gks3d\$inq_pat_rep function, *Part 2*, 11–97
gks3d\$inq_pick_id function, *Part 2*, 11–23
gks3d\$inq_pick_state3 function, *Part 2*, 11–100
gks3d\$inq_pick_state function, *Part 2*, 11–98
gks3d\$inq_pixel function, *Part 2*, 11–102
gks3d\$inq_pixel_array function, *Part 2*, 11–103
gks3d\$inq_pixel_array_dim function, *Part 2*, 11–105
gks3d\$inq_pline_fac function, *Part 2*, 11–106
gks3d\$inq_pline_indexes function, *Part 2*, 11–78
gks3d\$inq_pline_rep function, *Part 2*, 11–108
gks3d\$inq_pmark_fac function, *Part 2*, 11–110
gks3d\$inq_pmark_indexes function, *Part 2*, 11–79
gks3d\$inq_pmark_rep function, *Part 2*, 11–112
gks3d\$inq_predef_colour_rep function, *Part 2*, 11–114
gks3d\$inq_predef_edge_rep function, *Part 2*, 11–115
gks3d\$inq_predef_fill_rep function, *Part 2*, 11–116
gks3d\$inq_predef_pat_rep function, *Part 2*, 11–117
gks3d\$inq_predef_pline_rep function, *Part 2*, 11–118
gks3d\$inq_predef_pmark_rep function, *Part 2*, 11–119
gks3d\$inq_predef_text_rep function, *Part 2*, 11–120
gks3d\$inq_predef_view_rep function, *Part 2*, 11–121
gks3d\$inq_prim_atlb3 function, *Part 2*, 11–26
gks3d\$inq_prim_atlb function, *Part 2*, 11–24
gks3d\$inq_seg_atlb3 function, *Part 2*, 11–124
gks3d\$inq_seg_atlb function, *Part 2*, 11–122
gks3d\$inq_seg_names function, *Part 2*, 11–129
gks3d\$inq_seg_names_on_ws function, *Part 2*, 11–130
gks3d\$inq_seg_priority function, *Part 2*, 11–94
gks3d\$inq_set_assoc_ws function, *Part 2*, 11–127
gks3d\$inq_string_state3 function, *Part 2*, 11–133
gks3d\$inq_string_state function, *Part 2*, 11–131
gks3d\$inq_stroke_state3 function, *Part 2*, 11–137
gks3d\$inq_stroke_state function, *Part 2*, 11–135
gks3d\$inq_text_extent3 function, *Part 2*, 11–140
gks3d\$inq_text_extent function, *Part 2*, 11–139
gks3d\$inq_text_fac function, *Part 2*, 11–141
gks3d\$inq_text_indexes function, *Part 2*, 11–80
gks3d\$inq_text_rep function, *Part 2*, 11–143
gks3d\$inq_valuator_state3 function, *Part 2*, 11–147
gks3d\$inq_valuator_state function, *Part 2*, 11–145
gks3d\$inq_view_fac function, *Part 2*, 11–149
gks3d\$inq_view_indexes function, *Part 2*, 11–81
gks3d\$inq_view_rep3 function, *Part 2*, 11–150
gks3d\$inq_wstype_list function, *Part 2*, 11–72
gks3d\$inq_ws_category function, *Part 2*, 11–152
gks3d\$inq_ws_classification function, *Part 2*, 11–153
gks3d\$inq_ws_defer_and_update function, *Part 2*, 11–155
gks3d\$inq_ws_max_num function, *Part 2*, 11–157
gks3d\$inq_ws_state function, *Part 2*, 11–158
gks3d\$inq_ws_type function, *Part 2*, 11–154
gks3d\$inq_ws_xform3 function, *Part 2*, 11–160
gks3d\$inq_ws_xform function, *Part 2*, 11–159
gks3d\$inq_xform3 function, *Part 2*, 11–92
gks3d\$inq_xform function, *Part 2*, 11–91
gks3d\$inq_xform_list function, *Part 2*, 11–76
gks3d\$insert_seg3 function, *Part 1*, 8–19
gks3d\$insert_seg function, *Part 1*, 8–17
gks3d\$interpret_item function, *Part 1*, 10–7
GKS3D\$K_PERFORM_FLAG, *Part 1*, 4–25
GKS3D\$K_POSTPONE_FLAG, *Part 1*, 4–25
gks3d\$log_error function, *Part 2*, 12–5
gks3d\$message function, *Part 1*, 4–19
gks3d\$open_gks function, *Part 1*, 4–20
gks3d\$open_ws function, *Part 1*, 4–21
gks3d\$polyline3 function, *Part 1*, 5–18
gks3d\$polyline function, *Part 1*, 5–17
gks3d\$polymarker3 function, *Part 1*, 5–20
gks3d\$polymarker function, *Part 1*, 5–19
gks3d\$read_item function, *Part 1*, 10–8
gks3d\$redraw_seg_on_ws function, *Part 1*, 4–22
gks3d\$rename_seg function, *Part 1*, 8–21
gks3d\$request_choice function, *Part 1*, 9–59
gks3d\$request_locator3 function, *Part 1*, 9–61
gks3d\$request_locator function, *Part 1*, 9–60
gks3d\$request_pick function, *Part 1*, 9–62
gks3d\$request_string function, *Part 1*, 9–63
gks3d\$request_stroke3 function, *Part 1*, 9–67
gks3d\$request_stroke function, *Part 1*, 9–65
gks3d\$request_valuator function, *Part 1*, 9–69
gks3d\$sample_choice function, *Part 1*, 9–70
gks3d\$sample_locator3 function, *Part 1*, 9–72
gks3d\$sample_locator function, *Part 1*, 9–71
gks3d\$sample_pick function, *Part 1*, 9–73
gks3d\$sample_string function, *Part 1*, 9–74
gks3d\$sample_stroke3 function, *Part 1*, 9–77
gks3d\$sample_stroke function, *Part 1*, 9–75
gks3d\$sample_valuator function, *Part 1*, 9–79
gks3d\$select_xform function, *Part 1*, 7–22

`gks3d$set_asf3` function, *Part 1*, 6–9
`gks3d$set_asf` function, *Part 1*, 6–7
`gks3d$set_choice_mode` function, *Part 1*, 9–80
`gks3d$set_clipping` function, *Part 1*, 7–23
`gks3d$set_colour_model` function, *Part 1*, 6–15
`gks3d$set_colour_rep` function, *Part 1*, 6–16
`gks3d$set_defer_state` function, *Part 1*, 4–23
`gks3d$set_edgetype` function, *Part 1*, 6–23
`gks3d$set_edgewidth_scale_fac` function, *Part 1*, 6–24
`gks3d$set_edge_colour_index` function, *Part 1*, 6–18
`gks3d$set_edge_flag` function, *Part 1*, 6–19
`gks3d$set_edge_index` function, *Part 1*, 6–20
`gks3d$set_error_handler` function, *Part 2*, 12–6
`gks3d$set_fill_colour_index` function, *Part 1*, 6–25
`gks3d$set_fill_index` function, *Part 1*, 6–26
`gks3d$set_fill_int_style` function, *Part 1*, 6–27
`gks3d$set_fill_rep` function, *Part 1*, 6–29
`gks3d$set_fill_style_index` function, *Part 1*, 6–31
`gks3d$set_hlshr_id` function, *Part 1*, 6–32
`gks3d$set_hlshr_mode` function, *Part 1*, 6–33
`gks3d$set_locator_mode` function, *Part 1*, 9–81
`gks3d$set_pat_ref_pt` function, *Part 1*, 6–38
`gks3d$set_pat_ref_pt_vec` function, *Part 1*, 6–39
`gks3d$set_pat_rep` function, *Part 1*, 6–40
`gks3d$set_pat_size` function, *Part 1*, 6–41
`gks3d$set_pick_id` function, *Part 1*, 6–42
`gks3d$set_pick_mode` function, *Part 1*, 9–82
`gks3d$set_pline_colour_index` function, *Part 1*, 6–43
`gks3d$set_pline_index` function, *Part 1*, 6–44
`gks3d$set_pline_linetype` function, *Part 1*, 6–34
`gks3d$set_pline_linewidth` function, *Part 1*, 6–35
`gks3d$set_pline_rep` function, *Part 1*, 6–45
`gks3d$set_pmark_colour_index` function, *Part 1*, 6–47
`gks3d$set_pmark_index` function, *Part 1*, 6–48
`gks3d$set_pmark_rep` function, *Part 1*, 6–49
`gks3d$set_pmark_size` function, *Part 1*, 6–36
`gks3d$set_pmark_type` function, *Part 1*, 6–37
`gks3d$set_seg_detectability` function, *Part 1*, 8–22
`gks3d$set_seg_highlighting` function, *Part 1*, 8–23
`gks3d$set_seg_priority` function, *Part 1*, 8–24
`gks3d$set_seg_visibility` function, *Part 1*, 8–27
`gks3d$set_seg_xform3` function, *Part 1*, 8–26
`gks3d$set_seg_xform` function, *Part 1*, 8–25
`gks3d$set_string_mode` function, *Part 1*, 9–83
`gks3d$set_stroke_mode` function, *Part 1*, 9–84
`gks3d$set_text_align` function, *Part 1*, 6–51
`gks3d$set_text_colour_index` function, *Part 1*, 6–53
`gks3d$set_text_expfac` function, *Part 1*, 6–11
`gks3d$set_text_fontprec` function, *Part 1*, 6–54
`gks3d$set_text_height` function, *Part 1*, 6–12
`gks3d$set_text_index` function, *Part 1*, 6–56
`gks3d$set_text_path` function, *Part 1*, 6–57
`gks3d$set_text_rep` function, *Part 1*, 6–59
`gks3d$set_text_spacing` function, *Part 1*, 6–13
`gks3d$set_text_upvec` function, *Part 1*, 6–14
`gks3d$set_valuator_mode` function, *Part 1*, 9–85
`gks3d$set_viewport3` function, *Part 1*, 7–28
`gks3d$set_viewport` function, *Part 1*, 7–27
`gks3d$set_viewport_priority` function, *Part 1*, 7–29
`gks3d$set_view_ind` function, *Part 1*, 7–24
`gks3d$set_view_rep3` function, *Part 1*, 7–25
`gks3d$set_view_xform_priority` function, *Part 1*, 7–26
`gks3d$set_window3` function, *Part 1*, 7–31
`gks3d$set_window` function, *Part 1*, 7–30
`gks3d$set_ws_viewport3` function, *Part 1*, 7–33
`gks3d$set_ws_viewport` function, *Part 1*, 7–32
`gks3d$set_ws_window3` function, *Part 1*, 7–35
`gks3d$set_ws_window` function, *Part 1*, 7–34
`gks3d$sizeof`, *Part 1*, 9–86
`gks3d$text3` function, *Part 1*, 5–22
`gks3d$text` function, *Part 1*, 5–21
`gks3d$update_ws` function, *Part 1*, 4–25
`gks3d$write_item` function, *Part 1*, 10–9
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
gksd3d\$set_edge_rep function, *Part 1*, 6–21
GKSdefmode, *Part 1*, 3–5
GKSerrfile, *Part 1*, 3–6
GKSerror, *Part 1*, 3–6
GKSirg, *Part 1*, 3–6
GKS level types
 list of, *Part 2*, B–8
GKSmetafile_type, *Part 1*, 3–6
GKSM metafiles, *Part 1*, 10–1
 creating, *Part 1*, 10–1 to 10–3
GKSndc_clip, *Part 1*, 3–6
GKS operating states
 list of, *Part 2*, B–8
GKSstroke_font1, *Part 1*, 3–6
GKSswstype, *Part 1*, 3–6
gks_decw_config.c, *Part 1*, 3–7
Graphics handlers, *Part 1*, 4–1
 See also Devices
 See also Workstations
 input, *Part 1*, 9–5
 nominal sizes, *Part 1*, 6–1

H

Handlers

 See also Devices
 See also Workstations
 See Graphics handlers
 errors, *Part 2*, 12–1

- Handlers (cont'd)
 - set and realized values, *Part 2*, 11–4
- Hardware fonts, *Part 1*, 6–54
 - See also Fonts
- Hatches, *Part 1*, 6–27
 - 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
 - text, *Part 1*, 6–12
- Highlighting
 - segments, *Part 1*, 8–6
- Highlighting flags
 - list of, *Part 2*, B–9
- Highlighting methods
 - list of, *Part 2*, B–9
- HLHSR identifiers
 - list of, *Part 2*, B–9
- HLHSR modes
 - list of, *Part 2*, B–9
- Hollow
 - fill area interior style, *Part 1*, 6–27
 - fill areas, *Part 1*, 5–8, 5–9
- Horizontal alignment types
 - list of, *Part 2*, B–12

I

- Identifiers
 - 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–36
- 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
- Implicit regeneration states
 - list of, *Part 2*, B–9
- 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
 - 2D arrays, *Part 1*, 5–4
 - 3D arrays, *Part 1*, 5–6
 - edge, *Part 1*, 6–21
 - edge color, *Part 1*, 6–18
 - fill area, *Part 1*, 6–26, 6–29

- Index
 - fill area (cont'd)
 - styles, *Part 1*, 6–31
 - into bundle tables, *Part 1*, 6–3
 - pattern styles, *Part 1*, 6–40
 - polyline, *Part 1*, 6–45
 - polymarker, *Part 1*, 6–49
 - text, *Part 1*, 6–56, 6–59
- Individual attributes, *Part 1*, 6–3
- Initialize
 - See also GKS
 - See also Workstations
 - GKS environment, *Part 1*, 4–20
 - workstation environment, *Part 1*, 4–21
- 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*, 9–41
- INITIALIZE LOCATOR function, *Part 1*, 9–39
 - example, *Part 1*, 9–87
- INITIALIZE PICK 3 function, *Part 1*, 9–45
- INITIALIZE PICK function, *Part 1*, 9–43
 - example, *Part 1*, 9–92
- INITIALIZE STRING 3 function, *Part 1*, 9–49
- INITIALIZE STRING function, *Part 1*, 9–47
 - example, *Part 1*, 9–99
- INITIALIZE STROKE 3 function, *Part 1*, 9–53
- INITIALIZE STROKE function, *Part 1*, 9–51
- INITIALIZE VALUATOR 3 function, *Part 1*, 9–57
- INITIALIZE VALUATOR function, *Part 1*, 9–55
 - example, *Part 1*, 9–103
- Initializing a logical input device, *Part 1*, 9–3
- Initializing input, *Part 1*, 9–14
- 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*, 9–20
 - cycling device control, *Part 1*, 9–14
 - data record
 - sizes, *Part 1*, 9–20
 - standard, *Part 1*, 9–8
 - using inquiry functions, *Part 1*, 9–20
 - default values, *Part 1*, 9–20
 - device-independent programming, *Part 1*, 9–20
 - event mode, *Part 1*, 9–17 to 9–19
 - flushing the queue, *Part 1*, 9–18
 - event queue, *Part 1*, 9–17
 - event queue overflow, *Part 1*, 9–18
 - initializing, *Part 1*, 9–14
 - 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 9–19

Input (cont'd)

- pick
 - visibility, *Part 1*, 8–10
 - pick identification, *Part 1*, 8–2
 - request mode, *Part 1*, 9–15 to 9–16
 - sample mode, *Part 1*, 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, 9–19
 - workstation categories, *Part 1*, 4–2
- Input classes
 - list of, *Part 2*, B–9
- Input data records
 - determining size of, *Part 1*, 9–9
 - sizes, *Part 1*, 9–20
- Input functions, *Part 1*, 9–1 to 9–108
 - gks3d\$sizeof, *Part 1*, 9–86
 - introduction to, *Part 1*, 9–1 to 9–19
- Input mode types
 - list of, *Part 2*, B–10
- Input operating modes, *Part 1*, 9–14
- Input priority
 - initial value, *Part 2*, E–4
- Input priority states
 - list of, *Part 2*, B–10
- Input status types
 - list of, *Part 2*, B–10
- INQUIRE ASPECT SOURCE FLAGS
 - See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function
- INQUIRE CHARACTER BASE VECTOR
 - See INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function
- INQUIRE CHARACTER EXPANSION FACTOR
 - See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function
- INQUIRE CHARACTER HEIGHT
 - See INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function
- INQUIRE CHARACTER SPACING
 - See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function
- INQUIRE CHARACTER UP VECTOR
 - See INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function
- INQUIRE CHARACTER WIDTH
 - See INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function
- INQUIRE CHOICE DEVICE STATE 3 function, *Part 2*, 11–7

- INQUIRE CHOICE DEVICE STATE function, *Part 2*, 11–5
- INQUIRE CLIPPING 3 function, *Part 2*, 11–10
- INQUIRE CLIPPING function, *Part 2*, 11–9
- INQUIRE COLOUR FACILITIES function, *Part 2*, 11–11
- INQUIRE COLOUR MODEL FACILITIES function, *Part 2*, 11–13
- INQUIRE COLOUR MODEL function, *Part 2*, 11–12
- INQUIRE COLOUR REPRESENTATION function, *Part 2*, 11–14
- INQUIRE CURRENT HLHSR IDENTIFIER VALUE function, *Part 2*, 11–15
- INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 3 function, *Part 2*, 11–19
- INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function, *Part 2*, 11–16
- INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER function, *Part 2*, 11–22
- INQUIRE CURRENT PICK IDENTIFIER VALUE function, *Part 2*, 11–23
- INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 3 function, *Part 2*, 11–26
- INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function, *Part 2*, 11–24
- INQUIRE DEFAULT CHOICE DEVICE DATA 3 function, *Part 2*, 11–30
- INQUIRE DEFAULT CHOICE DEVICE DATA function, *Part 2*, 11–28
- INQUIRE DEFAULT DEFERRAL STATE VALUES function, *Part 2*, 11–32
- INQUIRE DEFAULT LOCATOR DEVICE DATA 3 function, *Part 2*, 11–34
- INQUIRE DEFAULT LOCATOR DEVICE DATA function, *Part 2*, 11–33
- INQUIRE DEFAULT PICK DEVICE DATA 3 function, *Part 2*, 11–37
- INQUIRE DEFAULT PICK DEVICE DATA function, *Part 2*, 11–35
- INQUIRE DEFAULT STRING DEVICE DATA 3 function, *Part 2*, 11–41
- INQUIRE DEFAULT STRING DEVICE DATA function, *Part 2*, 11–39
- INQUIRE DEFAULT STROKE DEVICE DATA 3 function, *Part 2*, 11–45
- INQUIRE DEFAULT STROKE DEVICE DATA function, *Part 2*, 11–43
- INQUIRE DEFAULT VALUATOR DEVICE DATA 3 function, *Part 2*, 11–49
- INQUIRE DEFAULT VALUATOR DEVICE DATA function, *Part 2*, 11–47
- INQUIRE DISPLAY SPACE SIZE 3 function, *Part 2*, 11–52

INQUIRE DISPLAY SPACE SIZE function, *Part 2*, 11–51
 example, *Part 1*, 7–49

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES function, *Part 2*, 11–53

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 3 function, *Part 2*, 11–57

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function, *Part 2*, 11–55
 example, *Part 1*, 7–49

INQUIRE EDGE FACILITIES function, *Part 2*, 11–59

INQUIRE EDGE REPRESENTATION function, *Part 2*, 11–60

INQUIRE FILL AREA COLOUR INDEX
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function

INQUIRE FILL AREA FACILITIES function, *Part 2*, 11–62

INQUIRE FILL AREA INTERIOR STYLE
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function
 See INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES function

INQUIRE FILL AREA REPRESENTATION function, *Part 2*, 11–63

INQUIRE FILL AREA STYLE INDEX
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function

INQUIRE GENERALIZED DRAWING PRIMITIVE 3 function, *Part 2*, 11–65

INQUIRE GENERALIZED DRAWING PRIMITIVE function, *Part 2*, 11–64

INQUIRE HLHSR FACILITIES function, *Part 2*, 11–66

INQUIRE HLHSR MODE function, *Part 2*, 11–67

INQUIRE INPUT QUEUE OVERFLOW function, *Part 1*, 9–18; *Part 2*, 11–68

INQUIRE LEVEL OF GKS function, *Part 2*, 11–69

INQUIRE LINETYPE
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function

INQUIRE LINEWIDTH SCALE FACTOR
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 function, *Part 2*, 11–71

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES function, *Part 2*, 11–70

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES function, *Part 2*, 11–72

INQUIRE LIST OF COLOUR INDICES function, *Part 2*, 11–73

INQUIRE LIST OF EDGE INDICES function, *Part 2*, 11–74

INQUIRE LIST OF FILL AREA INDICES function, *Part 2*, 11–75

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS function, *Part 2*, 11–76

INQUIRE LIST OF PATTERN INDICES function, *Part 2*, 11–77

INQUIRE LIST OF POLYLINE INDICES function, *Part 2*, 11–78

INQUIRE LIST OF POLYMARKER INDICES function, *Part 2*, 11–79

INQUIRE LIST OF TEXT INDICES function, *Part 2*, 11–80

INQUIRE LIST OF VIEW INDICES function, *Part 2*, 11–81

INQUIRE LOCATOR DEVICE STATE 3 function, *Part 2*, 11–84

INQUIRE LOCATOR DEVICE STATE function, *Part 2*, 11–82
 example, *Part 1*, 9–87

INQUIRE MARKER SIZE SCALE FACTOR
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function

INQUIRE MARKERTYPE
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES function
 See INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUESfunction

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 3 function, *Part 2*, 11–87

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES function, *Part 2*, 11–86

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER function, *Part 2*, 11–88

INQUIRE MORE SIMULTANEOUS EVENTS function, *Part 2*, 11–89

INQUIRE NAME OF OPEN SEGMENT function, *Part 2*, 11–90

INQUIRE NORMALIZATION TRANSFORMATION 3 function, *Part 2*, 11–92

INQUIRE NORMALIZATION
 TRANSFORMATION function, *Part 2*,
 11-91

INQUIRE NUMBER OF AVAILABLE LOGICAL
 INPUT DEVICES function, *Part 2*, 11-93

INQUIRE NUMBER OF SEGMENT PRIORITIES
 SUPPORTED function, *Part 2*, 11-94

INQUIRE OPERATING STATE function, *Part 2*,
 11-95

INQUIRE PATTERN FACILITIES function, *Part*
 2, 11-96

INQUIRE PATTERN REFERENCE POINT
 See INQUIRE CURRENT PRIMITIVE
 ATTRIBUTE VALUES function

INQUIRE PATTERN REPRESENTATION
 function, *Part 2*, 11-97

INQUIRE PATTERN SIZE
 See INQUIRE CURRENT PRIMITIVE
 ATTRIBUTE VALUES function

INQUIRE PICK DEVICE STATE 3 function, *Part*
 2, 11-100

INQUIRE PICK DEVICE STATE function, *Part 2*,
 11-98
 example, *Part 1*, 9-92

INQUIRE PIXEL ARRAY DIMENSIONS function,
 Part 2, 11-105

INQUIRE PIXEL ARRAY function, *Part 2*, 11-103

INQUIRE PIXEL function, *Part 2*, 11-102

INQUIRE POLYLINE COLOUR INDEX
 See INQUIRE CURRENT INDIVIDUAL
 ATTRIBUTE VALUES function

INQUIRE POLYLINE FACILITIES function, *Part*
 2, 11-106

INQUIRE POLYLINE INDEX
 See INQUIRE CURRENT INDIVIDUAL
 ATTRIBUTE VALUES function
 See INQUIRE CURRENT INDIVIDUAL
 ATTRIBUTE VALUES function
 See INQUIRE CURRENT PRIMITIVE
 ATTRIBUTE VALUES function

INQUIRE POLYLINE REPRESENTATION
 function, *Part 2*, 11-108

INQUIRE POLYMARKER FACILITIES function,
 Part 2, 11-110

INQUIRE POLYMARKER INDEX
 See INQUIRE CURRENT PRIMITIVE
 ATTRIBUTE VALUES function

INQUIRE POLYMARKER REPRESENTATION
 function, *Part 2*, 11-112

INQUIRE PREDEFINED COLOUR
 REPRESENTATION function, *Part 2*,
 11-114

INQUIRE PREDEFINED EDGE
 REPRESENTATION function, *Part 2*,
 11-115

INQUIRE PREDEFINED FILL AREA
 REPRESENTATION function, *Part 2*, 11-116

INQUIRE PREDEFINED PATTERN
 REPRESENTATION function, *Part 2*,
 11-117

INQUIRE PREDEFINED POLYLINE
 REPRESENTATION function, *Part 2*, 11-118

INQUIRE PREDEFINED POLYMARKER
 REPRESENTATION function, *Part 2*, 11-119

INQUIRE PREDEFINED TEXT
 REPRESENTATION function, *Part 2*,
 11-120

INQUIRE PREDEFINED VIEW
 REPRESENTATION function, *Part 2*,
 11-121

INQUIRE SEGMENT ATTRIBUTES 3 function,
 Part 2, 11-124

INQUIRE SEGMENT ATTRIBUTES function,
 Part 2, 11-122

INQUIRE SET OF ACTIVE WORKSTATIONS
 function, *Part 2*, 11-126

INQUIRE SET OF ASSOCIATED
 WORKSTATIONS function, *Part 2*,
 11-127

INQUIRE SET OF OPEN WORKSTATIONS
 function, *Part 2*, 11-128

INQUIRE SET OF SEGMENT NAMES IN USE
 function, *Part 2*, 11-129

INQUIRE SET OF SEGMENT NAMES ON
 WORKSTATION function, *Part 2*, 11-130

INQUIRE STRING DEVICE STATE 3 function,
 Part 2, 11-133

INQUIRE STRING DEVICE STATE function,
 Part 2, 11-131
 example, *Part 1*, 9-99

INQUIRE STROKE DEVICE STATE 3 function,
 Part 2, 11-137

INQUIRE STROKE DEVICE STATE function,
 Part 2, 11-135

INQUIRE TEXT ALIGNMENT
 See INQUIRE CURRENT PRIMITIVE
 ATTRIBUTE VALUES function

INQUIRE TEXT COLOUR INDEX
 See INQUIRE CURRENT INDIVIDUAL
 ATTRIBUTE VALUES function

INQUIRE TEXT EXTENT 3 function, *Part 2*,
 11-140

INQUIRE TEXT EXTENT function, *Part 2*,
 11-139

INQUIRE TEXT FACILITIES function, *Part 2*,
 11-141

INQUIRE TEXT FONT AND PRECISION
 See INQUIRE CURRENT INDIVIDUAL
 ATTRIBUTE VALUES function

INQUIRE TEXT INDEX

- See INQUIRE CURRENT PRIMITIVE
ATTRIBUTE VALUES function
- INQUIRE TEXT PATH
 - See INQUIRE CURRENT PRIMITIVE
ATTRIBUTE VALUES function
- INQUIRE TEXT REPRESENTATION function,
Part 2, 11-143
- INQUIRE VALUATOR DEVICE STATE 3 function,
Part 2, 11-147
- INQUIRE VALUATOR DEVICE STATE function,
Part 2, 11-145
 - example, *Part 1*, 9-103
- INQUIRE VIEW FACILITIES function, *Part 2*,
11-149
- INQUIRE VIEW REPRESENTATION 3 function,
Part 2, 11-150
- INQUIRE WORKSTATION CATEGORY function,
Part 2, 11-152
- INQUIRE WORKSTATION CLASSIFICATION
function, *Part 2*, 11-153
- INQUIRE WORKSTATION CONNECTION AND
TYPE function, *Part 2*, 11-154
 - example, *Part 1*, 4-29
- INQUIRE WORKSTATION DEFERRAL AND
UPDATE STATES function, *Part 2*, 11-155
- INQUIRE WORKSTATION MAXIMUM
NUMBERS function, *Part 2*, 11-157
- INQUIRE WORKSTATION STATE function, *Part*
2, 11-158
- INQUIRE WORKSTATION TRANSFORMATION
3 function, *Part 2*, 11-160
- INQUIRE WORKSTATION TRANSFORMATION
function, *Part 2*, 11-159
- Inquiry functions, *Part 2*, 11-1 to 11-160
 - 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
- Interface
 - prompt and echo types, *Part 1*, 9-5 to 9-14
- Interior styles
 - See also Attributes
 - See also Hatches
 - See also Patterns
 - of fill areas, *Part 1*, 6-27
- 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

- Languages
 - BASIC, *Part 2*, D-2
 - C, *Part 2*, D-2
 - COBOL, *Part 2*, D-2
 - declaring external functions, *Part 2*, D-1
 - Pascal, *Part 2*, D-5
 - programming information, *Part 2*, D-1 to D-6
- 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
- Line cap styles
 - list of, *Part 2*, B-10
- Line join styles
 - list of, *Part 2*, B-10
- Lines
 - See also Attributes
 - See also Output
 - generating, *Part 1*, 5-17, 5-18
 - types, *Part 1*, 1-3, 6-34
 - width, *Part 1*, 6-35
- Line types
 - DEC GKS implementation-specific
 - list of, *Part 2*, B-10
 - standard
 - list of, *Part 2*, B-10
- Linking, *Part 1*, 2-2, 3-2
 - reducing time, *Part 1*, 3-7
 - RISC processors, *Part 1*, 3-2
- Lists
 - See also GKS
 - See also Input
 - See also Workstations
 - GKS state, *Part 2*, 11-1
 - segment state, *Part 2*, 11-1
 - viewport input priority, *Part 1*, 7-6, 9-19
 - workstation state, *Part 2*, 11-1
- Locator
 - input class, *Part 1*, 9-4
 - viewport input priority, *Part 1*, 7-6, 9-19
- 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*, 2–4
- GKS\$ASF, *Part 1*, 2–4
- GKS\$CONID, *Part 1*, 2–4
- GKS\$DEF_MODE, *Part 1*, 2–4
- 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*, 2–5
- GKS\$WSTYPE, *Part 1*, 2–3

Logical types

- list of, *Part 2*, B–11

M

Mapping

- See also Transformations
- color indexes, *Part 1*, 5–4, 5–6
- device transformations, *Part 1*, 7–7

Markers, *Part 1*, 5–19, 5–20

- See also Attributes
- See also Output
- size, *Part 1*, 6–36
- types, *Part 1*, 6–37

Marker types

- DEC GKS implementation-specific
 - list of, *Part 2*, B–11
- standard
 - list of, *Part 2*, B–11

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*, 4–19

- example, *Part 1*, 9–87

Messages

- See also Errors
- error, *Part 2*, A–1 to A–37
- produced by error handler, *Part 2*, 12–2
- sent to workstations, *Part 1*, 4–19

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

- 2D cell arrays, *Part 1*, 5–4
- 3D cell arrays, *Part 1*, 5–6

Mode

- See also Input
- control
 - SET CHOICE MODE, *Part 1*, 9–80
 - SET LOCATOR MODE, *Part 1*, 9–81
 - SET PICK MODE, *Part 1*, 9–82
 - SET STRING MODE, *Part 1*, 9–83
 - SET STROKE MODE, *Part 1*, 9–84
 - SET VALUATOR MODE, *Part 1*, 9–85
- event, *Part 1*, 9–2, 9–3, 9–17
 - AWAIT EVENT, *Part 1*, 9–22
 - FLUSH DEVICE EVENTS, *Part 1*, 9–24
 - GET CHOICE, *Part 1*, 9–25
 - GET LOCATOR, *Part 1*, 9–26
 - GET LOCATOR 3, *Part 1*, 9–27
 - GET PICK, *Part 1*, 9–28
 - GET STRING, *Part 1*, 9–29
 - GET STROKE, *Part 1*, 9–30
 - GET STROKE 3, *Part 1*, 9–32
 - GET VALUATOR, *Part 1*, 9–34
- input operating, *Part 1*, 9–2, 9–3, 9–14
- request, *Part 1*, 9–2, 9–3, 9–15
 - REQUEST CHOICE, *Part 1*, 9–59
 - REQUEST LOCATOR, *Part 1*, 9–60
 - REQUEST LOCATOR 3, *Part 1*, 9–61

Mode

request (cont'd)

- REQUEST PICK, *Part 1*, 9-62
- REQUEST STRING, *Part 1*, 9-63
- REQUEST STROKE, *Part 1*, 9-65
- REQUEST STROKE 3, *Part 1*, 9-67
- REQUEST VALUATOR, *Part 1*, 9-69

sample, *Part 1*, 9-2, 9-3, 9-16

- SAMPLE CHOICE, *Part 1*, 9-70
- SAMPLE LOCATOR, *Part 1*, 9-71
- SAMPLE LOCATOR 3, *Part 1*, 9-72
- SAMPLE PICK, *Part 1*, 9-73
- SAMPLE STRING, *Part 1*, 9-74
- SAMPLE STROKE, *Part 1*, 9-75
- SAMPLE STROKE 3, *Part 1*, 9-77
- SAMPLE VALUATOR, *Part 1*, 9-79

Model

- color, *Part 1*, 6-15

Multiple transformations

See also Segments

See also Transformations

N

Names

- error messages, *Part 2*, 12-2
- 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

New frame action necessary states

- list of, *Part 2*, B-11

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
 - 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

error messages

Numbers

error messages (cont'd)

- handling, *Part 2*, 12-2

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*, 4-20

- example, *Part 1*, 4-27

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, 3-3

OPEN WORKSTATION function, *Part 1*, 4-21

- example, *Part 1*, 4-27

Operating modes

- input, *Part 1*, 9-2, 9-3, 9-14 to 9-19

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

deferral, *Part 1*, 4-6, 5-3

DECwindows, *Part 1*, 1-7

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*, 9–18

Overlapping

See also Transformations

segments, *Part 1*, 8–6

viewports, *Part 1*, 7–6, 9–19

P

Parallel projection, *Part 1*, 7–7

Pascal programming information, *Part 2*, D–5

Passing by descriptor, *Part 2*, D–2

problems, *Part 2*, D–1

Passing mechanisms

arguments, *Part 1*, 1–6

Pasteboard

See also Transformations

normalization viewport, *Part 1*, 7–5

Path

See also Text

text, *Part 1*, 6–57

Patterns, *Part 1*, 6–27

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*, 9–16

visibility, *Part 1*, 8–10

Pick status types

list of, *Part 2*, B–12

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

using GKS3D\$POLYLINE3 function, *Part 1*, 5–18

using GKS3D\$POLYLINE function, *Part 1*, 5–17

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

bundles, *Part 1*, 6–44

line type, *Part 1*, 1–3, 6–34

representation, *Part 1*, 6–45

POLYLINE 3 function, *Part 1*, 5–18

POLYLINE function, *Part 1*, 5–17

example, *Part 1*, 6–66

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*, 5–20

POLYMARKER function, *Part 1*, 5–19

example, *Part 1*, 6–69

Polymarkers
 initial attributes, *Part 2*, E-1 to E-2

Positioning
 primitives, *Part 1*, 7-5

Precision
 text, *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
 metatables, *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, 9-19

Program examples, *Part 2*, C-1 to C-3

Programming
 See also GKS
 BASIC, *Part 2*, D-2
 C, *Part 2*, D-2
 COBOL, *Part 2*, D-2
 device-independent input, *Part 1*, 9-20
 error handling, *Part 2*, 12-1
 language-specific information, *Part 2*, D-1
 Pascal, *Part 2*, D-5

Programs
 execution of, *Part 1*, 2-2, 3-2
 pausing, *Part 1*, 1-7

Projections
 parallel, *Part 1*, 7-7
 perspective, *Part 1*, 7-7

Projection types
 list of, *Part 2*, B-12

Prompt and echo types, *Part 1*, 9-2, 9-5 to 9-14
 See also Input
 standard data records, *Part 1*, 9-8

Proportionate
 See also Transformations

Q

Queue
 event input, *Part 1*, 9-17

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
 escape data, *Part 1*, 4-16
 input, *Part 1*, 9-8
 prompt and echo types, *Part 1*, 9-5 to 9-14
 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*, 4-22
 example, *Part 1*, 8-28

Regeneration flag states
 list of, *Part 2*, B-12

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*, 9-17

Representation
 See also Attributes
 bundle table entries, *Part 1*, 6-3
 color, *Part 1*, 6-16
 edge, *Part 1*, 6-21
 fill area, *Part 1*, 6-29

Representation (cont'd)
 functions, *Part 1*, 6–4
 implicit regenerations, *Part 1*, 6–4
 pattern, *Part 1*, 6–40
 polyline, *Part 1*, 6–45
 polymarker, *Part 1*, 6–49
 text, *Part 1*, 6–59

Reproducing
 metafiles, *Part 1*, 10–1

REQUEST CHOICE function, *Part 1*, 9–59
 REQUEST functions, *Part 1*, 9–2, 9–3, 9–15
 REQUEST LOCATOR 3 function, *Part 1*, 9–61
 REQUEST LOCATOR function, *Part 1*, 9–60
 Request mode, *Part 1*, 9–15 to 9–16
 See also Input
 breaking, *Part 1*, 9–15
 REQUEST PICK function, *Part 1*, 9–62
 REQUEST STRING function, *Part 1*, 9–63
 example, *Part 1*, 9–99
 REQUEST STROKE 3 function, *Part 1*, 9–67
 REQUEST STROKE function, *Part 1*, 9–65
 REQUEST VALUATOR function, *Part 1*, 9–69

Returned type values
 list of, *Part 2*, B–12

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*, 9–70
 SAMPLE functions, *Part 1*, 9–16
 SAMPLE LOCATOR 3 function, *Part 1*, 9–72
 SAMPLE LOCATOR function, *Part 1*, 9–71
 Sample mode, *Part 1*, 9–16
 SAMPLE PICK function, *Part 1*, 9–73
 example, *Part 1*, 9–92
 SAMPLE STRING function, *Part 1*, 9–74
 SAMPLE STROKE 3 function, *Part 1*, 9–77
 SAMPLE STROKE function, *Part 1*, 9–75
 SAMPLE VALUATOR function, *Part 1*, 9–79
 example, *Part 1*, 9–103

Scale
 See also Segments
 edge width factor, *Part 1*, 6–24
 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, *Part 1*, 8–22
 SET HIGHLIGHTING, *Part 1*, 8–23
 SET SEGMENT PRIORITY, *Part 1*, 8–24
 SET VISIBILITY, *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, *Part 1*, 7–10
 ACCUMULATE TRANSFORMATION
 MATRIX 3, *Part 1*, 7–12
 EVALUATE TRANSFORMATION MATRIX,
Part 1, 7–14
 EVALUATE TRANSFORMATION MATRIX
 3, *Part 1*, 7–16
 EVALUATE VIEW MAPPING MATRIX 3,
Part 1, 7–18
 EVALUATE VIEW ORIENTATION
 MATRIX 3, *Part 1*, 7–20
 SET SEGMENT TRANSFORMATION,
Part 1, 8–25
 SET SEGMENT TRANSFORMATION 3,
Part 1, 8–26
 transformation matrix, *Part 1*, 8–8
 translating, *Part 1*, 8–7
 view mapping matrix 3, *Part 1*, 7–18
 view orientation matrix 3, *Part 1*, 7–20
 visibility, *Part 1*, 8–10
 WDSS, *Part 1*, 8–2

Segments (cont'd)

- WISS, *Part 1*, 8–3
- SELECT NORMALIZATION TRANSFORMATION function, *Part 1*, 7–22
 - example, *Part 1*, 7–45
- SET ASPECT SOURCE FLAGS 3 function, *Part 1*, 6–9
- SET ASPECT SOURCE FLAGS function, *Part 1*, 6–7
 - example, *Part 1*, 6–63
- SET CHARACTER EXPANSION FACTOR function, *Part 1*, 6–11
- SET CHARACTER HEIGHT function, *Part 1*, 6–12
 - example, *Part 1*, 6–69
- SET CHARACTER SPACING function, *Part 1*, 6–13
- SET CHARACTER UP VECTOR function, *Part 1*, 6–14
- SET CHOICE MODE function, *Part 1*, 9–80
- SET CLIPPING INDICATOR function, *Part 1*, 7–23
 - example, *Part 1*, 7–45
- SET COLOUR MODEL function, *Part 1*, 6–15
- SET COLOUR REPRESENTATION function, *Part 1*, 6–16
 - example, *Part 1*, 6–61
- SET DEFERRAL STATE function, *Part 1*, 4–23
 - example, *Part 1*, 5–23
- SET DETECTABILITY function, *Part 1*, 8–22
 - example, *Part 1*, 9–92
- SET EDGE COLOUR INDEX function, *Part 1*, 6–18
- SET EDGE FLAG function, *Part 1*, 6–19
- SET EDGE INDEX function, *Part 1*, 6–20
- SET EDGE REPRESENTATION function, *Part 1*, 6–21
- SET EDGETYPE function, *Part 1*, 6–23
- SET EDGEWIDTH SCALE FACTOR function, *Part 1*, 6–24
- SET ERROR HANDLER function, *Part 2*, 12–6
- SET FILL AREA COLOUR INDEX function, *Part 1*, 6–25
 - example, *Part 1*, 6–61
- SET FILL AREA INDEX function, *Part 1*, 6–26
 - example, *Part 1*, 6–63
- SET FILL AREA INTERIOR STYLE function, *Part 1*, 6–27
 - example, *Part 1*, 6–61
- SET FILL AREA REPRESENTATION function, *Part 1*, 6–29
 - example, *Part 1*, 6–63
- 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*, 6–66
- SET LINEWIDTH SCALE FACTOR function, *Part 1*, 6–35
- SET LOCATOR MODE function, *Part 1*, 9–81
 - example, *Part 1*, 9–87
- 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*, 6–69
- SET MODE functions, *Part 1*, 9–2, 9–3, 9–14, 9–15, 9–16
- 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*, 9–92
- SET PICK MODE function, *Part 1*, 9–82
 - example, *Part 1*, 9–92
- 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*, 6–69
- 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*, 7–36
 SET STRING MODE function, *Part 1*, 9–83
 SET STROKE MODE function, *Part 1*, 9–84
 SET TEXT ALIGNMENT function, *Part 1*, 6–51
 example, *Part 1*, 6–69
 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*, 6–69
 SET TEXT REPRESENTATION function, *Part 1*,
 6–59
 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*, 9–85
 example, *Part 1*, 9–103
 Set values, *Part 2*, 11–4
 SET VIEW INDEX function, *Part 1*, 7–24
 SET VIEWPORT 3 function, *Part 1*, 7–28
 SET VIEWPORT function, *Part 1*, 7–27
 example, *Part 1*, 7–45
 SET VIEWPORT INPUT PRIORITY function,
 Part 1, 7–29
 SET VIEW REPRESENTATION 3 function, *Part*
 1, 7–18, 7–25
 SET VIEW TRANSFORMATION INPUT
 PRIORITY function, *Part 1*, 7–26
 SET VISIBILITY function, *Part 1*, 8–27
 SET WINDOW 3 function, *Part 1*, 7–31
 SET WINDOW function, *Part 1*, 7–30
 example, *Part 1*, 7–45
 SET WORKSTATION VIEWPORT 3 function,
 Part 1, 7–33
 SET WORKSTATION VIEWPORT function, *Part*
 1, 7–32
 example, *Part 1*, 7–49
 SET WORKSTATION WINDOW 3 function, *Part*
 1, 7–35
 SET WORKSTATION WINDOW function, *Part 1*,
 7–34
 Shift segments, *Part 1*, 8–7
 Shrink segments, *Part 1*, 8–7
 Simultaneous events flags
 list of, *Part 2*, B–12
 SIZEOF function, *Part 1*, 9–86
 Sizes
 input data record, *Part 1*, 9–20
 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*, 6–27
 fill areas, *Part 1*, 5–8, 5–9
 Spacing
 text, *Part 1*, 6–13
 Standards
 See also ANSI
 See also GKS
 DEC GKS escape data records, *Part 1*, 4–16
 metafiles, *Part 1*, 10–1
 State lists
 GKS, *Part 1*, 4–3, 4–20, 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; *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*, 9–19
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*, 5–19, 5–20

Synchronous input, *Part 1*, 9–14
See also Input

Syntax
format, *Part 1*, 1–5

System defaults file, *Part 1*, 3–7

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
edge bundle index, *Part 1*, 6–21
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*, 6–59
workstation description, *Part 2*, 11–1

Terminate
GKS environment, *Part 1*, 4–11
workstation environment, *Part 1*, 4–12

Terminating
error handling, *Part 2*, 12–1
request input, *Part 1*, 9–15

Text, *Part 1*, 5–21
See also Attributes
See also TEXT function
alignment, *Part 1*, 6–51
attributes
SET CHARACTER EXPANSION FACTOR,
Part 1, 6–11

Text
attributes (cont'd)
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*, 6–59
spacing, *Part 1*, 6–13
up-vector, *Part 1*, 6–14
TEXT 3 function, *Part 1*, 5–22
TEXT function, *Part 1*, 5–21
example, *Part 1*, 6–69
Text horizontal alignment types
list of, *Part 2*, B–12
Text path types
list of, *Part 2*, B–12
Text precision types
list of, *Part 2*, B–13
Text vertical alignment types
list of, *Part 2*, B–13
Toggling
logical input device control, *Part 1*, 9–14
Transformation functions, *Part 1*, 7–1 to 7–53
introduction to, *Part 1*, 7–1 to 7–8
transformation matrix, *Part 1*, 7–10, 7–14
transformation matrix 3, *Part 1*, 7–12, 7–16
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
text height, *Part 1*, 6–12
normalization viewports, *Part 1*, 7–4
normalization windows, *Part 1*, 7–3
overlapping viewports, *Part 1*, 9–19

Transformations (cont'd)

- 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*, 9-19
 - workstation, *Part 1*, 1-3
- ## 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
- ## Truncation
- metafile data record, *Part 1*, 10-8
- ## Types
- edge type, *Part 1*, 6-23
 - inquiry value type argument, *Part 2*, 11-4
 - lines, *Part 1*, 6-34
 - markers, *Part 1*, 6-37
 - prompt and echo, *Part 1*, 9-5 to 9-14
 - workstation
 - metafile, *Part 1*, 10-1
 - workstations, *Part 1*, 4-2

U

ULTRIX linking

- RISC processors, *Part 1*, 3-2
- ULTRIX operating system, *Part 1*, 3-1 to 3-8
- 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 states

- list of, *Part 2*, B-13
- UPDATE WORKSTATION function, *Part 1*, 4-25
 - example, *Part 1*, 4-27

Up-vector

- text, *Part 1*, 6-14
- User defaults file, *Part 1*, 3-7
- 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-17
- VAX languages, *Part 2*, D-1

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

Vertical alignment types

- list of, *Part 2*, B-13

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
- Viewport priority states
 - list of, *Part 2*, B-13

Viewports

- See also Transformations
- input priority, *Part 1*, 7-6, 9-19
- normalization, *Part 1*, 7-4
 - initial value, *Part 2*, E-4
- overlapping, *Part 1*, 7-6, 9-19
- workstation, *Part 1*, 7-7
- View reference coordinates, *Part 1*, 7-1
 - VRC, *Part 1*, 7-7

View table, *Part 1*, 7-7

Visibility flags

- list of, *Part 2*, B-13
- Visibility segments, *Part 1*, 8-10

Visual interface

- See also Input
- input prompt and echo types, *Part 1*, 9-5 to 9-14

VMS logical names

- GKS\$CONID, *Part 1*, 2-2
- GKS\$ERRFILE, *Part 1*, 2-5
- GKS\$WSTYPE, *Part 1*, 2-3

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

lines, *Part 1*, 6–35

Windows

See also Transformations

normalization

initial value, *Part 2*, E–4

workstation, *Part 1*, 7–7

WISS, *Part 1*, 4–2, 8–3

Workstation category types

list of, *Part 2*, B–13

Workstation class types

list of, *Part 2*, B–14

Workstation color availability states

list of, *Part 2*, B–14

Workstation identifier, *Part 1*, 9–1

Workstation MI types

list of, *Part 2*, B–14

Workstation MO types

list of, *Part 2*, B–14

Workstations

activating, *Part 1*, 4–5

attributes, *Part 1*, 6–1

clearing the surface, *Part 1*, 4–10

closing, *Part 1*, 4–5

deactivating, *Part 1*, 4–5

definition of, *Part 1*, 4–2

description tables, *Part 1*, 4–1

device coordinates, *Part 1*, 7–1

device manipulation

GKS3D\$ESCAPE, *Part 1*, 4–14

device number, *Part 1*, 9–1

environment, *Part 1*, 4–1

initializing, *Part 1*, 4–21

terminating, *Part 1*, 4–12

foreground and background colors, *Part 1*, 6–5

identifiers

input, *Part 1*, 9–1

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

sending messages to, *Part 1*, 4–19

state list

attributes, *Part 1*, 6–3

color table, *Part 1*, 6–15, 6–16

edge bundle table, *Part 1*, 6–21

fill area bundle table, *Part 1*, 6–29

pattern style bundle table, *Part 1*, 6–40

Workstations

state list (cont'd)

polyline bundle table, *Part 1*, 6–45

polymarker bundle table, *Part 1*, 6–49

text bundle table, *Part 1*, 6–59

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

Workstation states

list of, *Part 2*, B–14

Workstation type

default, *Part 1*, 2–3, 3–3

defined, *Part 1*, 2–3, 3–3

specifying on ULTRIX, *Part 1*, 3–3

specifying on VMS, *Part 1*, 2–3

Workstation types

list of, *Part 2*, B–14

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 modes

list of, *Part 2*, B–16

Writing to metafiles, *Part 1*, 10–2

WSTYPE/CONID association modes

list of, *Part 2*, B–17

