DEC GKS

GKS$ Binding Reference Manual, Part 2

Order Number: AA–PQP4A–TE

June 1992

This manual describes the GKS$ 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 postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK5677

This manual is available on CDROM.

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

F Differences in GKS Implementations

Index

Examples

Tables

Preface

This manual contains complete descriptions for the GKS$ 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 GKS$ 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 GKS3D$ Binding Reference Manual*—for programmers who need specific syntax and argument descriptions for the GKS3D$ 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
Return	The symbol RETURN 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.
Italic text	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.

11
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	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. 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.
Workstation description table	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. 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.
GKS state list	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. If you need to check the alterable DEC GKS values, you may need to inquire about the values contained in the GKS state list.

Table/List	Description
Workstation state list	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. 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.
Segment state list	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. If you need information concerning a particular segment, you may need to inquire about the values contained in the segment state list.

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:

```
gks$inq_level ( error_status, 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_status*. 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_status* argument. This function returns the level of the DEC GKS implementation with which you are working in the write-only argument *gks_level*.

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

```
gks$inq_locator_state ( ws_id, dev_num, value_type, error_status, op_mode,
echo_flag, trans_num, world_loc_x, world_loc_y,
pr_echo_type, echo_area, data_rec, rec_buf_len,
tot_rec_size )
```

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 *error_status* argument determines whether the values passed to the remaining write-only arguments are valid.

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

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

If a value is not presently available, the inquiry function returns a number, corresponding to an appropriate DEC GKS error message, in the *error_status* argument. If the value passed to the *error_status* 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.

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
GKS$K_VALUE_SET	If you specify this constant (or the value 0), the inquiry function returns the requested values exactly as specified in the application program. If you did not assign any values in the application program, the inquiry function returns the default values.
GKS$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

gks$inq_choice_state (ws_id, dev_num, error_status, op_mode, echo_flag, init_choice_status, init_choice_num, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_choice_ status	Integer (constant)	Write	Reference	Initial choice status.
init_choice_ num	Integer	Write	Reference	Initial choice measure.
pr_echo_type	Integer	Write	Reference	Prompt and echo type
echo_area	Array of reals	Write	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Choice input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the choice data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Constants

Defined Argument	Constant	Description
op_mode	GKS$K_INPUT_MODE_REQUEST	Request mode
	GKS$K_INPUT_MODE_SAMPLE	Sample mode

Defined Argument	Constant	Description
	GKS$K_INPUT_MODE_EVENT	Event mode
echo_flag	GKS$K_NOECHO	Echo disabled
	GKS$K_ECHO	Echo enabled
init_choice_status	GKS$K_STATUS_OK	Input obtained
	GKS$K_STATUS_NOCHOICE	No choice input obtained

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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 CLIPPING

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_clip (error_status, clip_ind, clip_rect)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
clip_ind	Integer (constant)	Write	Reference	Clipping indicator
clip_rect	Array of reals	Write	Reference	Four elements containing the NDC values of the clipping rectangle (XMIN, XMAX, YMIN, YMAX)

Constants

Defined Argument	Constant	Description
clip_ind	GKS$K_NOCLIP	Clipping disabled
	GKS$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 COLOUR FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_color_fac (ws_type, error_status, num_colors, color_flag, num_color_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
error_status	Integer	Write	Reference	Error status.
num_colors	Integer	Write	Reference	Number of colors. If the function returns 0, a continuous range of colors is available.
color_flag	Integer (constant)	Write	Reference	Color availability flag.
num_color_ind	Integer	Write	Reference	Number of predefined indexes.

Constants

Defined Argument	Constant	Description
color_flag	GKS$K_MONOCHROME	Monochrome display
	GKS$K_COLOR	Color display

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 REPRESENTATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_color_rep (ws_id, color_ind, value_type, error_status, red_inten, green_inten, blue_inten)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
color_ind	Integer	Read	Reference	Color index
value_type	Integer (constant)	Read	Reference	Type of output values
error_status	Integer	Write	Reference	Error status
red_inten	Real	Write	Reference	Red intensity
green_inten	Real	Write	Reference	Green intensity
blue_inten	Real	Write	Reference	Blue intensity

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Return the exact state list values.
	GKS$K_VALUE_REALIZED	Return the values approximated by the workstation.

Description

The INQUIRE COLOUR REPRESENTATION function returns the color triplet values, which are the color coordinates of the RGB 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 INDIVIDUAL ATTRIBUTE VALUES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_indiv_attb (error_status, pline_type, pline_width_sf, pline_color_ind, pmark_type, pmark_size, pmark_color_ind, text_font, text_prec, char_exp_factor, char_space, text_color_ind, int_style, style_ind, fill_color_ind, as_flags)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
pline_type	Integer (constant)	Write	Reference	Polyline type
pline_width_sf	Real	Write	Reference	Polyline width scale factor
pline_color_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_color_ind	Integer	Write	Reference	Polymarker color index
text_font	Integer	Write	Reference	Font number
text_prec	Integer (constant)	Write	Reference	Text precision
char_exp_factor	Real	Write	Reference	Character expansion factor
char_space	Real	Write	Reference	Character spacing
text_color_ind	Integer	Write	Reference	Text color index
int_style	Integer (constant)	Write	Reference	Fill area interior style
style_ind	Integer	Write	Reference	Fill area style index
fill_color_ind	Integer	Write	Reference	Fill area color index
as_flags	Array of integers	Write	Reference	13-element array containing aspect source flags for each of the nongeometric output attributes

Constants

Defined Argument	Constant	Description
pline_type	GKS$K_LINETYPE_SOLID	Line type solid.
	GKS$K_LINETYPE_DASHED	Line type dashed.
	GKS$K_LINETYPE_DOTTED	Line type dotted.
	GKS$K_LINETYPE_DASHED_ DOTTED	Line type dashed-dotted.
pmark_type	GKS$K_MARKERTYPE_DOT	Marker type dot.
	GKS$K_MARKERTYPE_ DIAGONAL_CROSS	Marker type diagonal cross.
	GKS$K_MARKERTYPE_PLUS	Marker type plus.
	GKS$K_MARKERTYPE_ ASTERISK	Marker type asterisk.
	GKS$K_MARKERTYPE_CIRCLE	Marker type circle.
text_prec	GKS$K_TEXT_PRECISION_ STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS$K_TEXT_PRECISION_CHAR	Character presision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GKS$K_TEXT_PRECISION_ STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.
int_style	GKS$K_INTSTYLE_HOLLOW	Interior style hollow.
	GKS$K_INTSTYLE_SOLID	Interior style solid.
	GKS$K_INTSTYLE_PATTERN	Interior style pattern.
	GKS$K_INTSTYLE_HATCH	Interior style hatch.
Aspect source flags	GKS$K_ASF_BUNDLED	Bundled attributes.
	GKS$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
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 NORMALIZATION TRANSFORMATION NUMBER

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_current_xformno (error_status, trans_num)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
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

gks$inq_pick_id (error_status, pick_id)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
pick_id	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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_prim_attb (error_status, bundle_ind, text_height, char_up_vec, char_width, text_base_vec, char_path, char_align, pattern_width, pattern_height, pattern_ref_point)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
bundle_ind	Array of integers	Write	Reference	4-element array containing bundle indexes (polyline index, polymarker index, text index, fill area index)
text_height	Real	Write	Reference	Character height specified by WC value
char_up_vec	Array of reals	Write	Reference	2-element array containing the current up vector (X, Y)
char_width	Real	Write	Reference	Character width specified by a WC value
text_base_vec	Array of reals	Write	Reference	2-element array containing text base vector (X, Y)
char_path	Integer (constant)	Write	Reference	Character path
char_align	Integer (constant)	Write	Reference	2-element array containing character alignment values
pattern_width	Array of reals	Write	Reference	2-element array containing pattern width vector (X, Y)
pattern_height	Array of reals	Write	Reference	2-element array containing pattern height vector (X, Y)
pattern_ref_point	Array of reals	Write	Reference	2-element array containing the X and Y WC values designating the pattern reference point (X, Y)

Constants

Defined Argument	Constant	Description
char_path	GKS$K_TEXT_PATH_RIGHT	Text string reads from left to right
	GKS$K_TEXT_PATH_LEFT	Text string reads from right to left
	GKS$K_TEXT_PATH_UP	Text string reads from bottom to top
	GKS$K_TEXT_PATH_DOWN	Text string reads from top to bottom

Defined Argument	Constant	Description
char_align	GKS$K_TEXT_VALIGN_NORMAL	Normal vertical alignment
	GKS$K_TEXT_VALIGN_TOP	Top vertical alignment
	GKS$K_TEXT_VALIGN_CAP	Cap vertical alignment
	GKS$K_TEXT_VALIGN_HALF	Half vertical alignment
	GKS$K_TEXT_VALIGN_BASE	Base vertical alignment
	GKS$K_TEXT_VALIGN_BOTTOM	Bottom vertical alignment

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 DEFAULT CHOICE DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_def_choice_data (ws_type, dev_num, error_status, max_choices, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
max_choices	Integer	Write	Reference	Maximum number of choices.
num_pr_ echo_types	Integer	Write	Reference	Number of choice prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Modify	Reference	Choice input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the choice data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Description

The INQUIRE DEFAULT CHOICE DEVICE DATA function returns the default values for the specified choice-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. 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 (gks*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 *rec_buf_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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_def_defer_state (ws_type, error_status, defer_mode, regen_flag)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
defer_mode	Integer (constant)	Write	Reference	Default deferral mode
regen_flag	Integer (constant)	Write	Reference	Default regeneration mode

Constants

Defined Argument	Constant	Description
defer_mode	GKS$K_ASAP	Generate images as soon as possible.
	GKS$K_BNIG	Generate images before the next interaction globally or before sample or event input occurs.
	GKS$K_BNIL	Generate images before the next interaction locally or before sample or event input occurs.
	GKS$K_ASTI	Generate images at some time. The exact time is determined by the workstation.
regen_flag	GKS$K_IRG_SUPPRESSED	Implicit regeneration is suppressed.
	GKS$K_IRG_ALLOWED	Implicit regeneration is allowed.

Description

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

See Also

SET DEFERRAL STATE

INQUIRE DEFAULT LOCATOR DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_def_locator_data (ws_type, dev_num, error_status, init_world_x, init_world_y, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
init_world_x	Real	Write	Reference	X coordinate of starting position of locator prompt expressed as a WC value.
init_world_y	Real	Write	Reference	Y coordinate of starting position of locator prompt expressed as a WC value.
num_pr_ echo_types	Integer	Write	Reference	Number of locator prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Locator input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the locator data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

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.

See Also

SIZEOF

INQUIRE DEFAULT PICK DEVICE DATA

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_def_pick_data (ws_type, dev_num, error_status, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
num_pr_ echo_types	Integer	Write	Reference	Number of pick prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Pick input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the pick data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

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 (gks*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 utility function to determine the size of the pick 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 *rec_buf_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

gks$inq_def_string_data (ws_type, dev_num, error_status, buffer_size, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
buffer_size	Integer	Write	Reference	Maximum string buffer size.
num_pr_ echo_types	Integer	Write	Reference	Number of string prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	String input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the string data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Description

The INQUIRE DEFAULT STRING DEVICE DATA function returns the default values for the specified string-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. 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 (gks*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 *rec_buf_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

gks$inq_def_stroke_data (ws_type, dev_num, error_status, buffer_size, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
buffer_size	Integer	Write	Reference	Maximum stroke buffer size in bytes.
num_pr_ echo_types	Integer	Write	Reference	Number of stroke prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Stroke input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the stroke data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Description

The INQUIRE DEFAULT STROKE DEVICE DATA function returns the default values for the specified stroke-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 Stroke Class section in the Input Functions chapter for information on the data record components. See the language-dependent header file (gks*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 *rec_buf_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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_def_valuator_data (ws_type, dev_num, error_status, init_value, num_pr_echo_types, pr_echo_types, echo_area, data_rec, num_ret_prompts, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
init_value	Real	Write	Reference	Default initial value.
num_pr_ echo_types	Integer	Write	Reference	Number of valuator prompt and echo types.
pr_echo_ types	Array of integers	Write	Descriptor	List of prompt and echo types available on the specified workstation.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Valuator input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
num_ret_ prompts	Integer	Write	Reference	Actual number of prompt and echo types returned.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the valuator data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

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 (gks*defs.*) for the specific data types and structure for PET 1 for a valuator 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 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 *rec_buf_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

gks$inq_max_ds_size (ws_type, error_status, meters, dev_coords_x, dev_coords_y, raster_units_x, raster_units_y)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
meters	Integer (constant)	Write	Reference	Flag to determine the types of device coordinate units
dev_coords_x	Real	Write	Reference	Maximum X coordinate value in device coordinates
dev_coords_y	Real	Write	Reference	Maximum Y coordinate value in device coordinates
raster_units_x	Integer	Write	Reference	Maximum X value in raster units
raster_units_y	Integer	Write	Reference	Maximum Y value in raster units

Constants

Defined Argument	Constant	Description
meters	GKS$K_METERS	Meters
	GKS$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 DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_dyn_mod_seg_attb (ws_type, error_status, trans_change, vis_to_invis, invis_to_vis, highl_change, pri_change, add_prim, seg_del)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
trans_change	Integer (constant)	Write	Reference	Workstation capability for dynamic modification with a segment transformation change
vis_to_invis	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when visibility changes from ON to OFF
invis_to_vis	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when visibility changes from OFF to ON
highl_change	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when highlighting changes
pri_change	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the segment priority changes
add_prim	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when primitives are added to an open segment
seg_del	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when a segment is deleted

Constants

Defined Argument	Constant	Description
trans_change, vis_to_invis, invis_to_vis, highl_change, pri_change, add_prim, seg_del	GKS$K_IRG GKS$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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_dyn_mod_ws_attb (ws_type, error_status, pline_rep, pmark_rep, text_rep, fill_rep, pattern_rep, color_rep, ws_trans)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	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
pattern_rep	Integer (constant)	Write	Reference	Workstation capability for dynamic modification when the pattern representation changes
color_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, pattern_rep, color_rep, ws_trans	GKS$K_IRG	Implicit regeneration
	GKS$K_IMM	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
Example 7–4 for a program example using the INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES function

INQUIRE FILL AREA FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_fill_fac (ws_type, error_status, num_int_styles, int_style_list, num_hatch_styles, hatch_style_list, num_fill_ind, hatch_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_int_ styles	Integer	Write	Reference	Number of available interior styles
int_style_list	Array of integers (constants)	Write	Reference	List of available interior styles
num_hatch_ styles	Integer	Write	Reference	Number of available hatch styles
hatch_style_ list	Array of integers	Write	Descriptor	List of available hatch styles
num_fill_ind	Integer	Write	Reference	Number of predefined fill area index values available on specified workstation type
hatch_ret_ size	Integer	Write	Reference	Number of hatch styles returned to the list

Constants

Data Type	Constant	Description
Fill area types	GKS$K_INTSTYLE_HOLLOW	Interior style hollow
	GKS$K_INTSTYLE_SOLID	Interior style solid
	GKS$K_INTSTYLE_PATTERN	Interior style pattern
	GKS$K_INTSTYLE_HATCH	Interior style hatch

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

gks$inq_fill_rep (ws_id, fill_area_ind, value_type, error_status, int_style, style_ind, color_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 values to return
error_status	Integer	Write	Reference	Error status
int_style	Integer (constant)	Write	Reference	Interior style
style_ind	Integer	Write	Reference	Interior style index
color_ind	Integer	Write	Reference	Fill area color index

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use exact state list values.
	GKS$K_VALUE_REALIZED	Use the values approximated by the workstation.
int_style	GKS$K_INTSTYLE_HOLLOW	Interior style hollow.
	GKS$K_INTSTYLE_SOLID	Interior style solid.
	GKS$K_INTSTYLE_PATTERN	Interior style pattern.
	GKS$K_INTSTYLE_HATCH	Interior style hatch.

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_gdp (ws_type, gdp_id, error_status, num_attr_sets, attr_list, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
gdp_id	Integer	Read	Reference	GDP identifier
error_status	Integer	Write	Reference	Error status
num_attr_sets	Integer	Write	Reference	Number of attribute sets associated with the specified GDP
attr_list	Array of integers (constant)	Write	Descriptor	List of attribute sets associated with the specified GDP
num_ret	Integer	Write	Reference	Number of attribute sets returned in the list

Constants

Defined Argument	Constant	Description
attr_list	GKS$K_POLYLN_ATTRI	Polyline bundle attributes
	GKS$K_POLYMR_ATTRI	Polymarker bundle attributes
	GKS$K_TEXT_ATTRI	Text bundle attributes
	GKS$K_FILLAR_ATTRI	Fill area bundle attributes

Description

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

INQUIRE INPUT QUEUE OVERFLOW

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_input_queue_overflow (error_status, ws_id, input_class, dev_num)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
ws_id	Integer	Write	Reference	Workstation identifier
input_class	Integer	Write	Reference	Input class
dev_num	Integer	Write	Reference	Device number

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

gks$inq_level (error_status, gks_level)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
gks_level	Integer (constant)	Write	Reference	GKS level

Constants

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_avail_gdp (ws_type, error_status, num_gdps, gdp_list, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_gdps	Integer	Write	Reference	Number of available GDPs
gdp_list	Array of integers	Write	Descriptor	List of supported GDPs for the specified workstation
num_ret	Integer	Write	Reference	Actual number of GDPs returned to the array

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 WORKSTATION TYPES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_wstype_list (error_status, num_ws_types, ws_type_list, num_ret)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
num_ws_types	Integer	Write	Reference	Number of available workstation types
ws_type_list	Array of integers	Write	Descriptor	List of various supported workstation types
num_ret	Integer	Write	Reference	Actual number of workstation types returned to the array

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

gks$inq_color_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined color indexes
list_ind	Array of integers	Write	Descriptor	List of color index values
num_ret	Integer	Write	Reference	Number of indexes returned to the list

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 FILL AREA INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_fill_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined fill area indexes
list_ind	Array of integers	Write	Descriptor	List of fill area index values
num_ret	Integer	Write	Reference	Number of indexes returned in the list

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_xform_list (error_status, num_trans, list_trans, num_ret)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
num_trans	Integer	Write	Reference	Number of defined normalization transformations
list_trans	Array of integers	Write	Descriptor	List of all defined normalization transformations, in priority order
num_ret	Integer	Write	Reference	Number of normalization transformations passed in the list

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

gks$inq_pat_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined pattern indexes
list_ind	Array of integers	Write	Descriptor	List of pattern index values
num_ret	Integer	Write	Reference	Number of indexes returned in the list

Description

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

See Also

SET PATTERN REPRESENTATION

INQUIRE LIST OF POLYLINE INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pline_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined polyline indexes
list_ind	Array of integers	Write	Descriptor	List of polyline index values
num_ret	Integer	Write	Reference	Number of indexes returned to the list

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

gks$inq_pmark_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined polymarker indexes
list_ind	Array of integers	Write	Descriptor	List of polymarker index values
num_ret	Integer	Write	Reference	Number of indexes returned in the list

Description

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

See Also

SET POLYMARKER REPRESENTATION

INQUIRE LIST OF TEXT INDICES

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_text_indexes (ws_id, error_status, num_ind, list_ind, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
num_ind	Integer	Write	Reference	Number of defined text indexes
list_ind	Array of integers	Write	Descriptor	List of text indexes
num_ret	Integer	Write	Reference	Number of indexes returned in the list

Description

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

See Also

SET TEXT REPRESENTATION

INQUIRE LOCATOR DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_locator_state (ws_id, dev_num, value_type, error_status, op_mode, echo_flag, trans_num, world_loc_x, world_loc_y, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
value_type	Integer (constant)	Read	Reference	Type of values to return.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
trans_num	Integer	Write	Reference	Initial normalization transformation
world_loc_x	Real	Write	Reference	Initial X value of locator position, expressed as a WC value.
world_loc_y	Real	Write	Reference	Initial Y value of locator position, expressed as a WC value.
pr_echo_type	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of reals	Write	Reference	Array of four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Locator input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the locator data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	Use the values approximated by the device handler.
op_mode	GKS$K_INPUT_MODE_ REQUEST	Request mode.
	GKS$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS$K_NOECHO	Echo disabled.
	GKS$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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 MAXIMUM LENGTH OF WORKSTATION STATE TABLES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_max_ws_state_table (ws_type, error_status, max_pline, max_pmark, max_text, max_fill_area, max_pattern, max_color)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Read	Reference	Error status
max_pline	Integer	Write	Reference	Maximum number of polyline bundle entries
max_pmark	Integer	Write	Reference	Maximum number of polymarker bundle entries
max_text	Integer	Write	Reference	Maximum number of text bundle entries
max_fill_area	Integer	Write	Reference	Maximum number of fill area bundle entries
max_pattern	Integer	Write	Reference	Maximum number of pattern indexes
max_color	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 NORMALIZATION TRANSFORMATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_max_xform (error_status, max_trans)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
max_trans	Integer	Write	Reference	Maximum normalization transformation number supported

Description

The INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER function returns the maximum normalization transformation number supported by the GKS implementation being used. The maximum number for the DEC GKS software is 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

gks$inq_more_simul_events (error_status, more_events_flag)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
more_ events_flag	Integer (constant)	Write	Reference	More simultaneously generated events flag

Constants

Defined Argument	Constant	Description
more_events_flag	GKS$K_NOMORE_EVENTS	No more events
	GKS$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

Operating States

SGOP

Syntax

gks$inq_name_open_seg (error_status, seg_name)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
seg_name	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

gks$inq_xform (trans_num, error_status, wind_bound, view_bound)

Argument	Data Type	Access	Passed by	Description
trans_num	Integer	Read	Reference	Specified normalization transformation number
error_status	Integer	Write	Reference	Error status
wind_bound	Array of reals	Write	Reference	Four WC values comprising the normalization window (XMIN, XMAX, YMIN, YMAX)
view_bound	Array of reals	Write	Reference	Four NDC values comprising the normalization viewport (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 NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_input_dev (ws_type, error_status, num_loc_dev, num_stroke_dev, num_val_dev, num_choice_dev, num_pick_dev, num_string_dev)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_loc_dev	Integer	Write	Reference	Number of locator devices
num_stroke_dev	Integer	Write	Reference	Number of stroke devices
num_val_dev	Integer	Write	Reference	Number of valuator devices
num_choice_dev	Integer	Write	Reference	Number of choice devices
num_pick_dev	Integer	Write	Reference	Number of pick devices
num_string_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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_seg_priority (ws_type, error_status, num_pri)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_pri	Integer	Write	Reference	Number of supported segment 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

gks$inq_operating_state (op_state)

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

Constants

Defined Argument	Constant	Description
op_state	GKS$K_GKCL	GKS closed
	GKS$K_GKOP	GKS open
	GKS$K_WSOP	At least one workstation open
	GKS$K_WSAC	At least one workstation active
	GKS$K_SGOP	At least one segment open

Description

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

See Also

OPEN GKS

INQUIRE PATTERN FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_pat_fac (ws_type, error_status, num_pattern_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_ pattern_ ind	Integer	Write	Reference	Number of predefined pattern indexes

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

gks$inq_pat_rep (ws_id, pattern_ind, value_type, error_status, pattern_width, pattern_height, list_color_ind, color_col_ret_size, color_rows_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
pattern_ind	Integer	Read	Reference	Pattern index
value_type	Integer (constant)	Read	Reference	Type of values to return
error_status	Integer	Write	Reference	Error status
pattern_ width	Integer	Write	Reference	Number of columns within the color array describing the pattern
pattern_ height	Integer	Write	Reference	Number of rows within the color array describing the pattern
list_color_ind	2D array of integers	Write	Descriptor	Color array describing the pattern
color_col_ret_ size	Integer	Write	Reference	X dimension of the list array
color_rows_ ret_size	Integer	Write	Reference	Y dimension of the color array

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	Use the values approximated by the workstation.

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pick_state (ws_id, dev_num, value_type, error_status, op_mode, echo_flag, init_pick_status, init_seg, init_pick_id, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
value_type	Integer (constant)	Read	Reference	Type of values to return.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
init_pick_ status	Integer (constant)	Write	Reference	Initial input status.
init_seg	Integer	Write	Reference	Initial pick segment.
init_pick_id	Integer	Write	Reference	Initial pick identifier.
pr_echo_type	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Pick input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the pick data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of the data record.

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values
	GKS$K_VALUE_REALIZED	Use the values approximated by the workstation.
op_mode	GKS$K_INPUT_MODE_ REQUEST	Request mode.
	GKS$K_INPUT_MODE_ SAMPLE	Sample mode.
	GKS$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS$K_NOECHO	Echo disabled.
	GKS$K_ECHO	Echo enabled.
init_pick_status	GKS$K_STATUS_OK	Input obtained.
	GKS$K_STATUS_NOPICK	No pick hit detected.

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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 PIXEL

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pixel (ws_id, world_x, world_y, error_status, color_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
world_x	Real	Read	Reference	X coordinate WC value of pixel
world_y	Real	Read	Reference	Y coordinate WC value of pixel
error_status	Integer	Write	Reference	Error status
color_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

gks$inq_pixel_array (ws_id, column_num, row_num, max_columns, max_rows, world_x, world_y, error_status, inval_ind_flag, color_index_array)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
column_num	Integer	Read	Reference	Starting column in color index array
row_num	Integer	Read	Reference	Starting row in color index array
max_ columns	Integer	Read	Reference	Number of columns of pixels
max_rows	Integer	Read	Reference	Number of rows of pixels
world_x	Real	Read	Reference	X coordinate WC value of upper left corner of the pixel array
world_y	Real	Read	Reference	Y coordinate WC value of upper left corner of the pixel array
error_status	Integer	Write	Reference	Error status
inval_ind_ flag	Integer (constant)	Write	Reference	Invalid color index value flag
color_index_ array	2D array of integers	Write	Descriptor	Color index array

Constants

Defined Argument	Constant	Description
inval_ind_flag	GKS$K_INVALID_ABSENT	Color array contains no invalid indexes.
	GKS$K_INVALID_PRESENT	Color array contains invalid indexes.

Description

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

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

See Also

SET COLOUR REPRESENTATION

INQUIRE PIXEL ARRAY DIMENSIONS

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pixel_array_dim (ws_id, start_point_x, start_point_y, diag_point_x, diag_point_y, error_status, dim_x, dim_y)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
start_point_x	Real	Read	Reference	X coordinate of starting point, WC value
start_point_y	Real	Read	Reference	Y coordinate of starting point, WC value
diag_point_x	Real	Read	Reference	X coordinate of point diagonal to the starting point, WC value
diag_point_y	Real	Read	Reference	Y coordinate of point diagonal to the starting point, WC value
error_status	Integer	Write	Reference	Error status
dim_x	Integer	Write	Reference	Pixel array columns
dim_y	Integer	Write	Reference	Pixel array rows

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_pline_fac (ws_type, error_status, num_line_types, line_types, num_line_widths, nom_line_width, line_width_min, line_width_max, num_ind, line_type_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_line_types	Integer	Write	Reference	Number of line types available
line_types	Array of integers (constant)	Write	Descriptor	List of line types available on specified workstation
num_line_widths	Integer	Write	Reference	Number of line widths
nom_line_width	Real	Write	Reference	Nominal default size line supported by the workstation
line_width_min	Real	Write	Reference	Smallest line width possible
line_width_max	Real	Write	Reference	Largest line width possible
num_ind	Integer	Write	Reference	Number of predefined polyline index values
line_type_ret_size	Integer	Write	Reference	Number of line types returned in the line type list

Constant

Defined Argument	Constant	Description
line_types	GKS$K_LINETYPE_SOLID	Line type solid
	GKS$K_LINETYPE_DASHED	Line type dashed
	GKS$K_LINETYPE_DOTTED	Line type dotted
	GKS$K_LINETYPE_DASHED_DOTTED	Line type dash-dotted

INQUIRE POLYLINE FACILITIES

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pline_rep (ws_id, pline_index, value_type, error_status, line_type, pline_width_sf, color_ind)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
pline_index	Integer	Read	Reference	Polyline index
value_type	Integer (constant)	Read	Reference	Type of values to return
error_status	Integer	Write	Reference	Error status
line_type	Integer (constant)	Write	Reference	Line type associated with the specified index
pline_width_sf	Integer	Write	Reference	Line width scale factor associated with the specified index
color_ind	Integer	Write	Reference	Color index associated with the specified index

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	Use the values aproximated by the workstation.
line_type	GKS$K_LINETYPE_SOLID	Line type solid.
	GKS$K_LINETYPE_DASHED	Line type dashed.
	GKS$K_LINETYPE_DOTTED	Line type dotted.
	GKS$K_LINETYPE_DASHED_DOTTED	Line type dashed-dotted.

Description

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

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

See Also

SET POLYLINE REPRESENTATION

INQUIRE POLYMARKER FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_pmark_fac (ws_type, error_status, num_marker_types, marker_types, num_marker_sizes, nom_marker_size, marker_size_min, marker_size_max, marker_indexes, marker_type_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_ marker_ types	Integer	Write	Reference	Number of marker types available
marker_ types	Array of integers (constant)	Write	Descriptor	List of marker types available on specified workstation
num_ marker_ sizes	Integer	Write	Reference	Number of marker sizes
nom_ marker_ size	Real	Write	Reference	Nominal default size marker supported by the workstation
marker_size_ min	Real	Write	Reference	Smallest marker size possible
marker_size_ max	Real	Write	Reference	Largest marker size possible
marker_ indexes	Integer	Write	Reference	Number of predefined polymarker index values
marker_ type_ret_size	Integer	Write	Reference	Number of marker types returned in the marker type list

Constants

Defined Argument	Constant	Description
marker_types	GKS$K_MARKERTYPE_DOT	Marker type dot (.)
	GKS$K_MARKERTYPE_PLUS	Marker type plus (+)
	GKS$K_MARKERTYPE_ ASTERISK	Marker type asterisk (*)
	GKS$K_MARKERTYPE_CIRCLE	Marker type circle (O)

Defined Argument	Constant	Description
	GKS$K_MARKERTYPE_ DIAGONAL_CROSS	Marker type diagonal cross (X)

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_pmark_rep (ws_id, pmark_ind, value_type, error_status, marker_type, marker_size_sf, color_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 values to return
error_status	Integer	Write	Reference	Error status
marker_type	Integer (constant)	Write	Reference	Polymarker type associated with the specified index
marker_size_sf	Real	Write	Reference	Polymarker size scale factor associated with the specified index
color_ind	Integer	Write	Reference	Color index associated with the specified index

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	Use the values approximated by the workstation.
marker_type	GKS$K_MARKERTYPE_DOT	Marker type dot (.).
	GKS$K_MARKERTYPE_PLUS	Marker type plus (+).
	GKS$K_MARKERTYPE_ASTERISK	Marker type asterisk (+).
	GKS$K_MARKERTYPE_CIRCLE	Marker type circle (O).
	GKS$K_MARKERTYPE_DIAGONAL_CROSS	Marker type diagonal cross (X).

Description

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

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

See Also

SET POLYMARKER REPRESENTATION

INQUIRE PREDEFINED COLOUR REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_color_rep (ws_type, color_ind, error_status, red_inten, green_inten, blue_inten)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
color_ind	Integer	Read	Reference	Color index
error_status	Integer	Write	Reference	Error status
red_inten	Real	Write	Reference	Red intensity
green_inten	Real	Write	Reference	Green intensity
blue_inten	Real	Write	Reference	Blue intensity

Description

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

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_fill_rep (ws_type, fill_index, error_status, int_style, style_ind, color_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
fill_index	Integer	Read	Reference	Fill area index
error_status	Integer	Write	Reference	Error status
int_style	Integer (constant)	Write	Reference	Fill area interior style
style_ind	Integer	Write	Reference	Fill area style index
color_ind	Integer	Write	Reference	Fill area color index

Constants

Defined Argument	Constant	Description
int_style	GKS$K_INTSTYLE_HOLLOW	Interior style hollow
	GKS$K_INTSTYLE_SOLID	Interior style solid
	GKS$K_INTSTYLE_PATTERN	Interior style pattern
	GKS$K_INTSTYLE_HATCH	Interior style hatch

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_pat_rep (ws_type, pattern_ind, error_status, height, width, color_indexes, color_col_ret_size, color_rows_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
pattern_ind	Integer	Read	Reference	Pattern index
error_status	Integer	Write	Reference	Error status
height	Integer	Write	Reference	Number of rows in the color index array describing the pattern
width	Integer	Write	Reference	Number of columns in the color index describing the pattern
color_indexes	2D array of integers	Write	Descriptor	Color index array describing the pattern
color_col_ret_size	Integer	Write	Reference	Number of columns returned in the color index array
color_rows_ret_size	Integer	Write	Reference	Number of rows returned in the color index array

Description

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

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_pline_rep (ws_type, pline_index, error_status, line_type, color_ind, line_width)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
pline_index	Integer	Read	Reference	Polyline index
error_status	Integer	Write	Reference	Error status
line_type	Integer (constant)	Write	Reference	Line type
color_ind	Integer	Write	Reference	Polyline color index
line_width	Real	Write	Reference	Line width scale factor

Constants

Defined Argument	Constant	Description
line_type	GKS$K_LINETYPE_SOLID	Line type solid
	GKS$K_LINETYPE_DASHED	Line type dashed
	GKS$K_LINETYPE_DOTTED	Line type dotted
	GKS$K_LINETYPE_DASHED_DOTTED	Line type dashed-dotted

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

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_pmark_rep (ws_type, pmark_ind, error_status, marker_type, color_ind, marker_size_sf)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
pmark_ind	Integer	Read	Reference	Polymarker index
error_status	Integer	Write	Reference	Error status
marker_type	Integer (constant)	Write	Reference	Marker type
color_ind	Integer	Write	Reference	Polymarker color index
marker_size_ sf	Real	Write	Reference	Marker size scale factor

Constants

Defined Argument	Constant	Description
marker_type	GKS$K_MARKERTYPE_DOT	Marker type dot
	GKS$K_MARKERTYPE_PLUS	Marker type plus
	GKS$K_MARKERTYPE_ ASTERISK	Marker type asterisk
	GKS$K_MARKERTYPE_CIRCLE	Marker type circle
	GKS$K_MARKERTYPE_ DIAGONAL_CROSS	Marker type diagonal cross

Description

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

INQUIRE PREDEFINED TEXT REPRESENTATION

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_predef_text_rep (ws_type, text_index, error_status, font_num, precision, char_exp_factor, char_space, color_ind)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
text_index	Integer	Read	Reference	Text index
error_status	Integer	Write	Reference	Error status
font_num	Integer	Write	Reference	Text font
precision	Integer (constant)	Write	Reference	Text precision
char_exp_ factor	Real	Write	Reference	Text expansion factor
char_space	Real	Write	Reference	Text spacing
color_ind	Integer	Write	Reference	Text color index

Constants

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_seg_attb (seg_name, error_status, trans_matrix, visibility, highlighting, priority, detectability)

Argument	Data Type	Access	Passed by	Description
seg_name	Integer	Read	Reference	Segment name
error_status	Integer	Write	Reference	Error status
trans_matrix	Array of reals	Write	Reference	6-element array containing the segment transformation matrix
visibility	Integer (constant)	Write	Reference	Visibility flag
highlighting	Integer (constant)	Write	Reference	Highlighting flag
priority	Real	Write	Reference	Priority
detectability	Integer (constant)	Write	Reference	Detectability flag

Constants

Defined Argument	Constant	Description
visibility	GKS$K_INVISIBLE	The segment is not visible on the workstation surface.
	GKS$K_VISIBLE	The segment is visible on the workstation surface. This is the default value.
highlighting	GKS$K_NORMAL	The segment is not highlighted. This is the default value.
	GKS$K_HIGHLIGHTED	The segment is highlighted.
detectability	GKS$K_UNDETECTABLE	The segment cannot be picked. This is the default value.
	GKS$K_DETECTABLE	The 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 SET OF ACTIVE WORKSTATIONS

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_active_ws (error_status, num_ws, ws_list, num_ret)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
num_ws	Integer	Write	Reference	Number of active workstations
ws_list	Array of integers	Write	Descriptor	Identifiers associated with the active workstations
num_ret	Integer	Write	Reference	Number of workstation identifiers returned in the list

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_set_assoc_ws (seg_name, error_status, num_ws, list_ws, num_ret)

Argument	Data Type	Access	Passed by	Description
seg_name	Integer	Read	Reference	Segment name
error_status	Integer	Write	Reference	Error status
num_ws	Integer	Write	Reference	Number of associated workstations
list_ws	Array of integers	Write	Descriptor	List of workstation identifiers
num_ret	Integer	Write	Reference	Number of workstation identifiers returned in the list

Description

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

See Also

ASSOCIATE SEGMENT WITH WORKSTATION
CREATE SEGMENT

INQUIRE SET OF OPEN WORKSTATIONS

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_open_ws (error_status, num_open_ws, ws_list, num_ret)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
num_open_ws	Integer	Write	Reference	Number of open workstations
ws_list	Array of integers	Write	Descriptor	List of open workstations
num_ret	Integer	Write	Reference	Number of workstation identifiers returned in the list

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_seg_names (error_status, num_seg, list_seg, num_ret)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
num_seg	Integer	Write	Reference	Number of existing segments
list_seg	Array of integers	Write	Descriptor	List of existing segments
num_ret	Integer	Write	Reference	Actual number of segment names returned in the list

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_seg_names_on_ws (ws_id, error_status, num_seg_names, list_seg_names, num_ret)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_seg_ names	Integer	Write	Reference	Number of segment names for specified workstation
list_seg_ names	Array of integers	Write	Descriptor	List of segment names
num_ret	Integer	Write	Reference	Number of segment names returned in the list

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

gks$inq_string_state (ws_id, dev_num, error_status, op_mode, echo_flag, default_string, string_ret_size, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
default_ string	Character string	Write	Descriptor	Default input string.
string_ret_ size	Integer	Write	Reference	Length of returned string, in bytes.
pr_echo_type	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	String input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the string data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Constants

Defined Argument	Constant	Description
op_mode	GKS$K_INPUT_MODE_REQUEST	Request mode
	GKS$K_INPUT_MODE_SAMPLE	Sample mode
	GKS$K_INPUT_MODE_EVENT	Event mode
echo_flag	GKS$K_NOECHO	Echo disabled
	GKS$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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 STROKE DEVICE STATE

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_stroke_state (ws_id, dev_num, value_type, num_elem, error_status, op_mode, echo_flag, trans_num, total_points, world_x_pts, world_y_pts, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
value_type	Integer (constant)	Read	Reference	Type of output values.
num_elem	Integer	Modify	Reference	Number of elements in the declared *world_x_pts* and *world_y_pts* array buffers.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
trans_num	Integer	Write	Reference	Normalization transformation.
total_points	Integer	Write	Reference	Number of points in initial stroke.
world_x_pts	Array of reals	Write	Reference	X coordinate of the initial point of the stroke, WC value.
world_y_pts	Array of reals	Write	Reference	Y coordinate of the initial point of the stroke, WC value.
pr_echo_type	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX).
data_rec	Record	Write	Reference	Stroke input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the stroke data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size, in bytes, of data record.

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	use the values approximated by the workstation device.
op_mode	GKS$K_INPUT_MODE_REQUEST	Request mode.
	GKS$K_INPUT_MODE_SAMPLE	Sample mode.
	GKS$K_INPUT_MODE_EVENT	Event mode.
echo_flag	GKS$K_NOECHO	Echo disabled.
	GKS$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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 TEXT EXTENT

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_text_extent (ws_id, string_pos_x, string_pos_y, string, error_status, concat_x, concat_y, ext_rect_x, ext_rect_y)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
string_pos_x	Real	Read	Reference	X WC value of the text string position
string_pos_y	Real	Read	Reference	Y WC value of the text string position
string	Character string	Read	Descriptor	The character string
error_status	Integer	Write	Reference	Error status
concat_x	Real	Write	Reference	X WC value of the concatenation point
concat_y	Real	Write	Reference	Y WC value of the concatenation point
ext_rect_x	Array of reals	Write	Reference	Array of four elements containing the X WC values of the extent rectangle
ext_rect_y	Array of reals	Write	Reference	Array of four elements containing the Y WC values of the extent rectangle

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 FACILITIES

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_text_fac (ws_type, error_status, num_fonts, font_list, prec_list, num_heights, height_min, height_max, num_char_exp, char_exp_min, char_exp_max, num_ind, prec_ret_size, font_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
num_fonts	Integer	Write	Reference	Number of fonts available on specified workstation type
font_list	Array of integers	Write	Descriptor	List of available font numbers
prec_list	Array of integers (constant)	Write	Descriptor	List of available text precisions
num_heights	Integer	Write	Reference	Number of available character heights
height_min	Real	Write	Reference	Minimum character height in device coordinates
height_max	Real	Write	Reference	Maximum character height in device coordinates
num_char_exp	Integer	Write	Reference	Number of available character expansions
char_exp_min	Real	Write	Reference	Minimum character expansion
char_exp_max	Real	Write	Reference	Maximum character expansion
num_ind	Integer	Write	Reference	Number of predefined text indexes
prec_ret_size	Integer	Write	Reference	Number of elements returned in precision array
font_ret_size	Integer	Write	Reference	Number of elements returned in font array

Constants

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

gks$inq_text_rep (ws_id, text_index, value_type, error_status, text_font, text_prec, char_exp_factor, char_space, color_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 output values to return
error_status	Integer	Write	Reference	Error status
text_font	Integer	Write	Reference	Text font
text_prec	Integer (constant)	Write	Reference	Text precision
char_exp_ factor	Real	Write	Reference	Character expansion factor
char_space	Real	Write	Reference	Character spacing
color_ind	Integer	Write	Reference	Text color index

Constants

Defined Argument	Constant	Description
value_type	GKS$K_VALUE_SET	Use the exact state list values.
	GKS$K_VALUE_REALIZED	use the values approximated by the workstation.
text_prec	GKS$K_TEXT_PRECISION_ STRING	String precision. DEC GKS evaluates character height and width attributes only.
	GKS$K_TEXT_PRECISION_CHAR	Character precision. DEC GKS evaluates each character for compliance with all other specified text attributes.
	GKS$K_TEXT_PRECISION_ STROKE	Stroke precision. DEC GKS looks for exact compliance with all specified text attributes.

INQUIRE TEXT REPRESENTATION

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

gks$inq_valuator_state (ws_id, dev_num, error_status, op_mode, echo_flag, default_value, pr_echo_type, echo_area, data_rec, rec_buf_len, tot_rec_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier.
dev_num	Integer	Read	Reference	Device number.
error_status	Integer	Write	Reference	Error status.
op_mode	Integer (constant)	Write	Reference	Operating input mode.
echo_flag	Integer (constant)	Write	Reference	Echo flag.
default_value	Real	Write	Reference	Default value of valuator input device.
pr_echo_type	Integer	Write	Reference	Prompt and echo type.
echo_area	Array of reals	Write	Reference	Four device coordinates specifying an echo area.
data_rec	Record	Write	Reference	Valuator input data record. See the appropriate language-dependent header file (gks*defs.*) for the specific data type and data record structure.
rec_buf_len	Integer	Modify	Reference	On input, the maximum number of bytes to write to the valuator data record. On output, the length of the returned data record, in bytes. Compare this value to *tot_rec_size* to see if the entire data record was returned.
tot_rec_size	Integer	Write	Reference	Total size of the data record, in bytes.

Constants

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

Defined Argument	Constant	Description
echo_flag	GKS$K_NOECHO	Echo disabled
	GKS$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 *pr_echo_type*. 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 (gks*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 *rec_buf_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 WORKSTATION CATEGORY

Operating States

GKOP, WSOP, WSAC, SGOP

Syntax

gks$inq_ws_category (ws_type, error_status, ws_cat)

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

Constants

Defined Argument	Constant	Description
ws_cat	GKS$K_WSCAT_OUTPUT	Output workstation
	GKS$K_WSCAT_INPUT	Input workstation
	GKS$K_WSCAT_OUTIN	Output/input workstation
	GKS$K_WSCAT_WISS	Workstation independent segment storage
	GKS$K_WSCAT_MO	Metafile output
	GKS$K_WSCAT_MI	Metafile input

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

gks$inq_ws_classification (ws_type, error_status, ws_class)

Argument	Data Type	Access	Passed by	Description
ws_type	Integer	Read	Reference	Workstation type
error_status	Integer	Write	Reference	Error status
ws_class	Integer (constant)	Write	Reference	Workstation classification

Constants

Defined Argument	Constant	Description
ws_class	GKS$K_WSCLASS_VECTOR	Vector workstation
	GKS$K_WSCLASS_RASTER	Raster workstation
	GKS$K_WSCLASS_OTHERD	Other device

Description

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

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

INQUIRE WORKSTATION CONNECTION AND TYPE

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_ws_type (ws_id, error_status, conn_id, ws_type, log_ret_size)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
conn_id	Character string	Write	Descriptor	Connection identifier
ws_type	Integer	Write	Reference	Workstation type
log_ret_size	Integer	Write	Reference	Length of the connection identifier string returned

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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_ws_defer_and_update (ws_id, error_status, defer_mode, regen_mode, surf_empty_flag, new_frame_flag)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
defer_mode	Integer (constant)	Write	Reference	Deferral mode
regen_mode	Integer (constant)	Write	Reference	Implicit regeneration mode
surf_empty_ flag	Integer (constant)	Write	Reference	Display surface empty flag
new_frame_ flag	Integer (constant)	Write	Reference	New frame necessary at update flag

Constants

Defined Argument	Constant	Description
defer_mode	GKS$K_ASAP	Generate images as soon as possible.
	GKS$K_BNIG	Generate images before the next interaction globally, or before sample or event input occurs.
	GKS$K_BNIL	Generate images before next interaction locally, or before sample or event input occurs.
	GKS$K_ASTI	Generate images at some time. The exact time is determined by the workstation.
regen_mode	GKS$K_IRG_SUPPRESSED	Implicit regeneration suppressed.
	GKS$K_IRG_ALLOWED	Implicit regeneration allowed.
surf_empty_flag	GKS$K_NOTEMPTY	Display surface not empty.
	GKS$K_EMPTY	Display surface empty.

Defined Argument	Constant	Description
new_frame_flag	GKS$K_NEWFRAME_ NOTNECESSARY	New frame action not necessary on update.
	GKS$K_NEWFRAME_ NECESSARY	New frame action necessary 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

gks$inq_ws_max_num (error_status, max_open_ws, max_active_ws, max_ws_seg)

Argument	Data Type	Access	Passed by	Description
error_status	Integer	Write	Reference	Error status
max_open_ws	Integer	Write	Reference	Maximum number of simultaneously open workstations
max_active_ws	Integer	Write	Reference	Maximum number of simultaneously active workstations
max_ws_seg	Integer	Write	Reference	Maximum number of workstations that can be associated with a 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

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_ws_state (ws_id, error_status, ws_state)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
ws_state	Integer (constant)	Write	Reference	Workstation state

Constants

Defined Argument	Constant	Description
ws_state	GKS$K_WS_INACTIVE	Workstation inactive
	GKS$K_WS_ACTIVE	Workstation active

Description

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

See Also

ACTIVATE WORKSTATION

INQUIRE WORKSTATION TRANSFORMATION

Operating States

WSOP, WSAC, SGOP

Syntax

gks$inq_ws_xform (ws_id, error_status, trans_pending, req_window, cur_window, req_viewport, cur_viewport)

Argument	Data Type	Access	Passed by	Description
ws_id	Integer	Read	Reference	Workstation identifier
error_status	Integer	Write	Reference	Error status
trans_ pending	Integer (constant)	Write	Reference	Transformation update state
req_window	Array of reals	Write	Reference	Four NDC points defining the requested workstation window (XMIN, XMAX, YMIN, YMAX)
cur_window	Array of reals	Write	Reference	Four NDC points defining the current workstation window (XMIN, XMAX, YMIN, YMAX)
req_viewport	Array of reals	Write	Reference	Four device coordinates defining the requested workstation viewport (XMIN, XMAX, YMIN, YMAX)
cur_viewport	Array of reals	Write	Reference	Four device coordinates defining the current workstation viewport (XMIN, XMAX, YMIN, YMAX)

Constants

Defined Argument	Constant	Description
trans_pending	GKS$K_NOTPENDING	Action not pending
	GKS$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 in 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

Error-Handling Functions

Insert tabbed divider here. Then discard this sheet.

12
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.

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

gks$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

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

gks$error_handler (error_num, func_name, error_file)

Argument	Data Type	Access	Passed by	Description
error_num	Integer	Read	Reference	Error number
func_name	Character string	Read	Descriptor	Function identifier
error_file	Character string	Read	Descriptor	Name of error logging file

Description

The ERROR HANDLING function calls the ERROR LOGGING function and allows continued program execution. By default, DEC GKS calls this function when it encounters an error condition. 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

ERROR LOGGING

ERROR LOGGING

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

gks$log_error (error_num, func_name, error_file)

Argument	Data Type	Access	Passed by	Description
error_num	Integer	Read	Reference	Error number
func_name	Character string	Read	Descriptor	Function identifier
error_file	Character string	Read	Descriptor	Error file

Description

The ERROR LOGGING function writes the standard DEC GKS error message, which includes the number of the error and the text of the message, to the error file and returns to the procedure or function from which it was called.

The error handler function supplied by DEC GKS (ERROR HANDLING) automatically calls the ERROR LOGGING function. An application-supplied error handler can call this function if necessary.

See Also

ERROR HANDLING
Example 12–1 for a program example using the ERROR LOGGING function

SET ERROR HANDLER

Operating States

GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

gks$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 Examples

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 <gksdefs.h>              /* GKS$ binding definition file */

static int error_handler ();

main ()
{
   int     default_conid;
   int     default_wstype;
   struct  dsc$descriptor_s   errorfile_dsc;
   char    *errorfile_name = "error.dat";
   struct  {  int (* error_hand1)();
              int (* error_hand2)(); } new_error_handler;
   struct  {  int (* error_hand1)();
              int (* error_hand2)(); } old_error_handler;
   float   start_x;
   float   start_y;
   struct  dsc$descriptor_s   string_dsc;
   char    *string_name     = "Test";
   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  = GKS$K_CONID_DEFAULT;
   default_wstype = GKS$K_WSTYPE_DEFAULT;

   gks$open_gks (&errorfile_dsc, 0);
   gks$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;

   gks$set_error_handler ( &new_error_handler, &old_error_handler);
```

(continued on next page)

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.
 */

   start_x = 0.5;
   start_y = 0.5;

   gks$text (&start_x, &start_y, &string_dsc);

/* Release the GKS and workstation environments. */

   gks$close_ws (&ws_id);
   gks$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. */

   gks$log_error ( error_number, function_name, error_file_name);

/* If ERROR_5, abort. */

   if (*error_number == ws_not_active)
       {
       gks$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. Consider the following example:

```
          .
          .
          .
C     Include the error symbol definitions file.
      INCLUDE 'SYS$LIBRARY:GKSMSGS.FOR'

C     Check for success.
      IF ( NO_ERROR = GKS$OPEN_WS( WS_ID, CON_ID, WS_TYPE ) ) THEN
          .
          .
          .
C     Check for an invalid workstation identifier.
      IF ( GKS$_ERROR_20 = GKS$ACTIVATE_WS( WS_ID ) ) THEN
          .
          .
          .
```

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 GKS$K_WSTYPE_WISS as an argument.

(continued on next page)

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)

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] × [0,1,0] × [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] × [0,1,0] × [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)

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 line types for a given workstation.
71	Marker size scale factor is less than zero in routine ****
	User Action: Either change the scale factor, or check the order and the number of the specified arguments.

(continued on next page)

Table A–4 (Cont.) Attribute Function Errors

Error	Error Message
72	Text index is invalid in routine **** User Action: Make sure the arguments are specified in the correct order and the index is an integer.
73	A representation for the specified text index has not been defined on this workstation in routine **** User Action: Use SET TEXT REPRESENTATION to define a representation for the index value, or use another predefined index value.
74	A representation for the specified text index has not been predefined on this workstation in routine **** User Action: Use SET TEXT REPRESENTATION to define a representation for the index value, or use another predefined index value.
75	Text font is equal to zero in routine **** 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.
76	Requested text font is not supported for the specified precision on this workstation in routine **** User Action: Lower the precision or change the font number.
77	Character expansion factor is less than or equal to zero in routine **** 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.
78	Character height is less than or equal to zero in routine **** 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.
79	Length of character up vector is zero in routine **** 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.
80	Fill area index is invalid in routine **** User Action: Make sure the arguments are specified in the correct order and the index is an integer.

(continued on next page)

Table A–4 (Cont.) Attribute Function Errors

Error	Error Message
81	A representation for the specified fill area index has not been defined on this workstation in routine ****
	User Action: Use SET FILL AREA REPRESENTATION to define a representation for the given index value, or pass another predefined index value.
82	A representation for the specified fill area index has not been predefined on this workstation in routine ****
	User Action: Use SET FILL AREA REPRESENTATION to define a representation for the given index value, or pass another predefined index value.
83	Specified fill area interior style is not supported on this workstation in routine ****
	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.
84	Style (pattern or hatch) index is equal to zero in routine ****
	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.
85	Specified pattern index is invalid in routine ****
	User Action: Make sure the arguments are specified in the correct order and the index is an integer.
86	Specified hatch style is not supported on this workstation in routine ****
	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.
87	Pattern size value is not positive in routine ****
	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.
88	A representation for the specified pattern index has not been defined on this workstation in routine ****
	User Action: Use SET PATTERN REPRESENTATION to define a representation for the pattern index, or pass another predefined index to the function.
89	A representation for the specified pattern index has not been predefined on this workstation in routine ****
	User Action: Use SET PATTERN REPRESENTATION to define a representation for the pattern index, or pass another predefined index to the function.

(continued on next page)

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	Number of points is invalid in routine ****
	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.
101	Invalid code in string in routine ****
	User Action: Your text string contained characters that cannot be printed. Remove the characters.
102	Generalized drawing primitive identifier is invalid in routine ****
	User Action: Specify another identifier or check to see if the identifier is an integer value.
103	Content of generalized drawing primitive data record is invalid in routine ****
	User Action: Make sure you passed a correct data record size.
104	At least one active workstation is not able to generate the specified generalized drawing primitive in routine ****
	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.
105	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformation and clipping rectangle in routine ****
	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.

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	Specified segment name is invalid in routine ****
	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.
121	Specified segment name is already in use in routine ****
	User Action: Either remove the call to CREATE SEGMENT or check to make sure you specified the correct segment name.

(continued on next page)

Table A–6 (Cont.) Segment Function Errors

Error	Error Message
122	Specified segment does not exist in routine ****
	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.
123	Specified segment does not exist on specified workstation in routine ****
	User Action: Either remove the function call, or if the segment exists in WISS, associate the segment with the appropriate workstation.
124	Specified segment does not exist on Workstation Independent Segment Storage in routine ****
	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.
125	Specified segment is open in routine ****
	User Action: Either remove the call to CREATE SEGMENT or specify another segment name.
126	Segment priority is outside the range [0,1] in routine ****
	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.

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	Specified input device is not present on workstation in routine ****
	User Action: Make sure you specified the function that applies to the correct logical input device and the correct workstation identifier.
141	Input device is not in REQUEST mode in routine ****
	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.
142	Input device is not in SAMPLE mode in routine ****
	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.

(continued on next page)

Table A–7 (Cont.) Input Function Errors

Error	Error Message
143	EVENT and SAMPLE input mode are not available at this level of GKS in routine **** User Action: This error code is required by the standard, but DEC GKS will never generate this error.
144	Specified prompt and echo type is not supported on this workstation in routine **** 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.
145	Echo area is outside display space in routine **** 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.
146	Contents of input data record are invalid in routine **** 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.
147	Input queue has overflowed in routine **** User Action: Check the input queue with greater frequency or flush the input queue.
148	Input queue has not overflowed since GKS was opened or the last invocation of INQUIRE INPUT QUEUE OVERFLOW in routine **** 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.
149	Input queue has overflowed, but associated workstation has been closed in routine **** 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.
150	No input value of the correct class is in the current event report in routine **** User Action: Make sure you check the input class argument passed to AWAIT EVENT before you try to call the appropriate GET function.

(continued on next page)

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.

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)

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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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)

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

Error	Error Message
–4	Invalid bit mask in workstation type in routine **** 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.
–5	Bad string addresses found writing choice data record in routine **** 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.
–6	Echo area is too narrow for data in routine **** 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.
–7	Maximum number of representable choices exceeded in routine **** 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.
–8	Echo area is too short for data in routine **** 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.
–9	Binary format and integer number representation not supported in this implementation of GKS in routine **** User Action: You opened a metafile of an incompatible type. Check the metafile type.
–10	Invalid value specified for ASF in routine **** User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GKS$K_ASF_BUNDLED (0) or GKS$K_ASF_INDIVIDUAL (1).
–11	Invalid value specified for fill area interior style in routine **** User Action: Make sure you passed one of the values GKS$K_INTSTYLE_HOLLOW (0), GKS$K_INTSTYLE_SOLID (1), GKS$K_INTSTYLE_PATTERN(2), or GKS$K_INTSTYLE_HATCH (3).
–12	Invalid value specified for horizontal component of text alignment in routine ****

(continued on next page)

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

Error	Error Message
	User Action: Make sure you passed one of the values GKS$K_TEXT_HALIGN_NORMAL (0), GKS$K_TEXT_HALIGN_LEFT (1), GKS$K_TEXT_HALIGN_CENTER (2), or GKS$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 GKS$K_TEXT_VALIGN_NORMAL (0), GKS$K_TEXT_VALIGN_TOP (1), GKS$K_TEXT_VALIGN_CAP (2), GKS$K_TEXT_VALIGN_HALF (3), GKS$K_TEXT_VALIGN_BASE (4), or GKS$K_TEXT_VALIGN_BOTTOM (5).
–14	Invalid value specified for text precision in routine ****
	User Action: Make sure you passed one of the values GKS$K_TEXT_PRECISION_STRING (0), GKS$K_TEXT_PRECISION_CHAR (1), or GKS$K_TEXT_PRECISION_STROKE (2).
–15	Invalid value specified for text path in routine ****
	User Action: Make sure you passed one of the values GKS$K_TEXT_PATH_RIGHT (0), GKS$K_TEXT_PATH_LEFT (1), GKS$K_TEXT_PATH_UP (2), or GKS$K_TEXT_PATH_DOWN (3).
–16	Echo switch is invalid in routine ****
	User Action: Make sure you passed the one of the values GKS$K_NOECHO (0) or GKS$K_ECHO (1). Also, if you used an inquiry function to obtain the echo switch, check to see that the arguments to the inquiry function are specified in the correct order.
–17	Inquired device values not set or realized in routine ****
	User Action: Check the value type argument to make sure it is either GKS$K_VALUE_SET(0) or GKS$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 ****
	User Action: Remove the call to the unsupported function.

(continued on next page)

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

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

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 *Building a Device Handler System for DEC GKS and DEC PHIGS*.
–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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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)

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)

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 GKS$K_ESC_SET_EDGE_INDEX is valid.
–151	Edge width scale factor is less than zero **** User Action: Make sure the edge width scale factor passed in the escape Set Edge Index (GKS$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)

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 (GKS$K_ESC_SET_SPEED).
–156	Loudness is outside range [0,1] in routine **** User Action: Pass a valid value to the escape Beep (GKS$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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*). 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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*). 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 GKS$K_NORMAL (0) or GKS$K_HIGHLIGHTED (1).

(continued on next page)

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

Error	Error Message
–301	Invalid value specified for visibility in routine **** User Action: Make sure you specify either GKS$K_INVISIBLE (0) or GKS$K_VISIBLE (1).
–302	Invalid value specified for detectability in routine **** User Action: Make sure you specify either GKS$K_UNDETECTABLE (0) or GKS$K_ DETECTABLE (1).
–303	Input device cannot be activated due to conflict with another input device that is currently active in routine **** 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.
–304	Cannot set input device echo on due to conflict with other input devices active in the same echo area in routine **** 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.
–305	Error parsing GKS$[wstype]_INPUT_DEVICES logical name due to syntax error in routine **** User Action: Check the definition of the logical name with the syntax description in the Tektronix 41xx chapter in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.
–306	The definition of GKS$HPGL_THRESHOLD is invalid (contains nonnumeric values in routine) **** User Action: Check the definition of GKS$HPGL_THRESHOLD and redefine to range 0 to 1023.
–450	Valuator device could not be used–dials are not available on the PCM device in routine **** 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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.
–451	Choice device could not be used–buttons are not available on the PCM device in routine ****

(continued on next page)

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

Error	Error Message
	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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.
–452	PCM initialization failed–hardware dials and buttons not available in routine ****
	User Action: The PCM server is not running. It has either not been started or the PCM dial and button box is not connected. Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware buttons will not be used. See the chapter on PCM button and dials in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.
–453	PCM configuration failed–valuator or choice device not initialized in routine ****
	User Action: Either hook up the dial and button box correctly, or change the value of the environment option "use dials and buttons" so the hardware buttons will not be used. See the chapter on PCM button and dials in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.
–1517	Error trying to open UID database
	User Action: Verify that the UID files gfx_decw.uid and gfx_decw_*xx*.uid (where *xx* is the language code) exist in either the specified system area or user path.
–1518	Error trying to fetch from UID database
	User Action: Verify that the name used in the DwtFetchWidget or DwtFetchWidgetOverride command matches that in the diagrams (in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*), and has not been altered while editing the UIL files.
–2000	Illegal call to routine **** while GKS is in an error state
	User Action: Make sure GKS is not called while it is already in an error state. This may happen if you call GKS functions from the GKS error handler provided by the application.
–2001	Invalid value specified for ASF in routine ****
	User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GKS$K_ASF_BUNDLED (0) or GKS$K_ASF_INDIVIDUAL (1).
–2002	Invalid clipping indicator in routine ****
	User Action: Make sure you passed either the value GKS$K_NOCLIP (0) or GKS$K_CLIP (1).
–2003	Invalid normalization transformation priority in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the normalization priority value is either GKS$K_INPUT_PRIORITY_HIGHER (0) or GKS$K_ INPUT_PRIORITY_LOWER (1).

(continued on next page)

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

Error	Error Message
–2004	Invalid view index specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified view index is in the permitted range (greater than or equal to 0 and less than the maximum number of view representations).
–2005	Invalid metafile format found in routine **** User Action: Make sure the input file you are trying to interpret is a metafile that was generated by DEC GKS.
–2006	Null text direction vector found in routine **** User Action: Make sure the arguments are specified in the correct order and that the text direction vector is not {0.0, 0.0, 0.0} (null vector).
–2007	Invalid text path specified in routine **** User Action: Make sure you passed one of the values GKS$K_TEXT_PATH_RIGHT (0), GKS$K_TEXT_PATH_LEFT (1), GKS$K_TEXT_PATH_UP (2), or GKS$K_TEXT_PATH_DOWN (3).
–2008	Invalid text character alignment specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the text alignment specified is a permitted value (GKS$K_TEXT_HALIGN_NORMAL (0), GKS$K_TEXT_HALIGN_LEFT (1), GKS$K_TEXT_HALIGN_CENTER (2), or GKS$K_TEXT_HALIGN_RIGHT (3) for the horizontal component, and GKS$K_TEXT_VALIGN_NORMAL (0), GKS$K_TEXT_VALIGN_TOP (1), GKS$K_TEXT_VALIGN_CAP (2), GKS$K_TEXT_VALIGN_HALF (3), GKS$K_TEXT_VALIGN_BASE (4), or GKS$K_TEXT_VALIGN_BOTTOM (5) for the vertical component).
–2009	Invalid fill area interior style specified in routine **** User Action: Make sure you passed one of the values GKS$K_INTSTYLE_HOLLOW (0), GKS$K_INTSTYLE_SOLID (1), GKS$K_INTSTYLE_PATTERN(2), or GKS$K_INTSTYLE_HATCH (3).
–2010	Invalid edge flag specified in routine **** User Action: Make sure the arguments are specified in the correct order and that the specified edge flag is a permitted value (GKS$K_EDGE_FLAG_OFF (0) or GKS$K_EDGE_FLAG_ON (1)).
–2011	Invalid aspect source flag specified in routine **** User Action: Check the array to make sure it has 13 elements and that its elements only contain the value GKS$K_ASF_BUNDLED (0) or GKS$K_ASF_INDIVIDUAL (1).
–2012	Null pattern reference vector found in routine ****

(continued on next page)

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
–2015	Invalid update control flag specified in routine ****
	User Action: Make sure you passed one of the values GKS$K_ASAP (0), GKS$K_BNIG (1), GKS$K_BNIL (2), or GKS$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 (GKS$K_IRG_SUPPRESSED (0) or GKS$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 (GKS$K_IRG_SUPPRESSED (0) or GKS$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 (GKS$K_INPUT_PRIORITY_HIGHER (0) or GKS$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 (GKS$K_TEXT_PRECISION_STRING (0), GKS$K_TEXT_PRECISION_CHAR (1), or GKS$K_TEXT_PRECISION_STROKE (2)).
–2020	Invalid coordinate switch argument in routine ****
	User Action: Make sure the arguments are in the correct order and that the coordinate switch argument is a permitted value (GKS$K_COORDINATES_WC (0) or GKS$K_COORDINATES_NDC (1)).
–2021	Invalid projection type argument in routine ****

(continued on next page)

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 GKS$K_PARALLEL_PROJECTION (0) or GKS$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: GKS$K_INPUT_MODE_REQUEST (0), GKS$K_INPUT_MODE_SAMPLE (1), or GKS$K_INPUT_MODE_EVENT (2).
–2025	Invalid input device echo state specified in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the input device echo state is either GKS$K_NOECHO (0) or GKS$K_ECHO (1).
–2026	Invalid polyline/fill area control flag in data record in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the polyline/fill area control flag is a permitted value (GKS$K_ACF_POLYLINE (0) or GKS$K_ACF_FILL_AREA (1)).
–2027	Invalid attribute control flag in data record in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the attribute control flag in the data record is a permitted value (GKS$K_ACF_CURRENT (0) or GKS$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)

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 input device class is one of the following values: GKS$K_INPUT_CLASS_NONE (0), GKS$K_INPUT_CLASS_LOCATOR (1), GKS$K_INPUT_CLASS_STROKE (2), GKS$K_INPUT_CLASS_VALUATOR (3), GKS$K_INPUT_CLASS_CHOICE (4), GKS$K_INPUT_CLASS_PICK (5), GKS$K_INPUT_CLASS_STRING (6), or GKS$K_INPUT_CLASS_VIEWPORT (7).
–2032	Invalid visibility flag specified in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the specified visibility flag is either GKS$K_INVISIBLE (0) or GKS$K_VISIBLE (1).
–2033	Invalid highlighting flag specified in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the specified highlighting flag is either GKS$K_NORMAL (0) or GKS$K_HIGHLIGHTED (1).
–2034	Invalid detectability flag specified in routine ****
	User Action: Make sure the arguments are specified in the correct order and that the specified detectability flag is either GKS$K_UNDETECTABLE (0) or GKS$K_DETECTABLE (1).
–2036	Invalid matrix specified, matrix has a zero fourth row in routine ****
	User Action: Make sure the specified matrix does not have a zero fourth row.
–2037	Warning, the specified view matrix is not invertible in routine ****
	User Action: Specify a matrix that is invertible.
–2038	Warning, a 3D item cannot be written into GKSM (2D metafile output) in routine ****
	User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
–2039	Warning, the normalization transformation is 3D and cannot be used with GKSM (2D metafile output) in routine ****
	User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
–2040	Warning, the segment transformation is 3D and cannot be used with GKSM (2D metafile output) in routine ****
	User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.
–2041	Warning, 3D inquiries are invalid with GKSM (2D metafile output) in routine ****
	User Action: Avoid using three-dimensional functions when writing to a GKSM two-dimensional metafile.

(continued on next page)

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

Error	Error Message
–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 *Building a Device Handler System for DEC GKS and DEC PHIGS*.
–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 *Building a Device Handler System for DEC GKS and DEC PHIGS*.

(continued on next page)

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)

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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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)

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 GKS$ 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 for VMS systems:

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

The language definitions files located in /usr/include/GKS are as follows for ULTRIX systems:

- /usr/include/GKS/gksdefs.h
- /usr/include/GKS/gksdefs.ada

Table B–1 lists the GKS$ constants. It contains the defined constant exactly as it is used in programming, its associated value as defined by DEC GKS, and a brief description of the constant. The constants are listed by category and are numerically ordered by value within the category. The constant for the language are identical to the listed constants except they do not have the '$' character.

Note

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

Table B–1 GKS$ Constants

Constant	Value	Description
Arc Types:		
GKS$K_ARC_TYPE_OPEN	1	Arc type open
GKS$K_ARC_TYPE_PIE	2	Arc type pie
GKS$K_ARC_TYPE_CHORD	3	Arc type chord
Aspect Source Flags:		
GKS$K_ASF_BUNDLED	0	Bundled
GKS$K_ASF_INDIVIDUAL	1	Individual
Attribute Control Flags:		
GKS$K_ACF_CURRENT	0	Input data record current values
GKS$K_ACF_SPECIFIED	1	Input data record specified values
Choice Input Prompt Flags:		
GKS$K_CHOICE_PROMPT_OFF	0	Choice data record prompt off
GKS$K_CHOICE_PROMPT_ON	1	Choice date record prompt on
Choice Status Types:		
GKS$K_STATUS_NONE	0	No input obtained
GKS$K_STATUS_OK	1	Input obtained
GKS$K_STATUS_NOCHOICE	2	Input is NOCHOICE
Clear Screen States:		
GKS$K_CLEAR_CONDITIONALLY	0	Clear conditionally
GKS$K_CLEAR_ALWAYS	1	Clear always
Clipping Flags:		
GKS$K_NOCLIP	0	Clipping off
GKS$K_CLIP	1	Clipping on
Color Validity States:		
GKS$K_INVALID_ABSENT	0	Invalid values absent
GKS$K_INVALID_PRESENT	1	Invalid values present
Coordinate Switch:		
GKS$K_COORDINATES_WC	0	World coordinates
GKS$K_COORDINATES_NDC	1	Normalized device coordinates

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Data Type Identifiers:		
GKS$K_TYP_INTEGER	0	Integer
GKS$K_TYP_CHOICE_REC	1	Choice input data record
GKS$K_TYP_LOC_REC	2	Locator input data record
GKS$K_TYP_PICK_REC	3	Pick input data record
GKS$K_TYP_STRING_REC	4	String input data record
GKS$K_TYP_STROKE_REC	5	Stroke input data record
GKS$K_TYP_VAL_REC	6	Valuator input data record
GKS$K_TYP_ESCAPE_REC	7	Escape data record
GKS$K_TYP_GDP_REC	8	GDP data record
GKS$K_TYP_REAL	9	Real
GKS$K_TYP_ADDRESS	10	Address
Default Connection Identifier:		
GKS$K_CONID_DEFAULT	0	Default connection identifier
Deferral Modes:		
GKS$K_ASAP	0	As soon as possible
GKS$K_BNIG	1	Before the next global interaction
GKS$K_BNIL	2	Before the next local interaction
GKS$K_ASTI	3	At some time
Detectability Flags:		
GKS$K_UNDETECTABLE	0	Set to undetectable
GKS$K_DETECTABLE	1	Set to detectable
Device Coordinate Units:		
GKS$K_METRES	0	Meters
GKS$K_OTHER_UNITS	1	Other units
Display Surface States:		
GKS$K_NOTEMPTY	0	Display surface not empty
GKS$K_EMPTY	1	Display surface empty
Dynamic Modification States:		
GKS$K_IRG	0	Implicit regeneration necessary
GKS$K_IMM	1	Immediate

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Echo States:		
GKS$K_NOECHO	0	Echo disabled
GKS$K_ECHO	1	Echo enabled
Edge Flags:		
GKS$K_NOEDGE	0	No edge displayed
GKS$K_EDGE	1	Edge displayed
Edge Types:		
GKS$K_EDGE_SOLID	1	Edge type solid
GKS$K_EDGE_DASHED	2	Edge type dashed
GKS$K_EDGE_DOTTED	3	Edge type dotted
GKS$K_EDGE_DASHED_DOTTED	4	Edge type dash-dotted
GKS$K_EDGE_DASH_2_DOT	–1	Edge type double-dashed dotted
GKS$K_EDGE_DASH_3_DOT	–2	Edge type triple-dashed dotted
GKS$K_EDGE_LONG_DASH	–3	Edge type long-dashed
GKS$K_EDGE_LONG_SHORT_DASH	–4	Edge type long-short-dashed
GKS$K_EDGE_SPACED_DASH	–5	Edge type spaced-dashed
GKS$K_EDGE_SPACED_DOT	–6	Edge type spaced-dotted
GKS$K_EDGE_DOUBLE_DOT	–7	Edge type double dotted
GKS$K_EDGE_TRIPLE_DOT	–8	Edge type triple dotted
Error Handling Modes:		
GKS$K_ERROR_OFF	0	No error handling
GKS$K_ERROR_ON	1	Error handling
Escape Function Identifiers:		
GKS$K_ESC_SET_SPEED	–100	Set display speed
GKS$K_ESC_PRINT	–101	Generate hardcopy of workstation surface
GKS$K_ESC_BEEP	–103	Beep
GKS$K_ESC_POP_WORKSTATION	–106	Pop workstation
GKS$K_ESC_PUSH_WORKSTATION	–107	Push workstation
GKS$K_ESC_SET_ERROR_HANDLING_MODE	–108	Set error handling mode
GKS$K_ESC_SET_VIEWPORT_EVENT	–109	Set viewport event
GKS$K_ESC_ASSOC_WSTYPE_CONID	–110	Associated workstation type and connection ID
GKS$K_ESC_SET_SOFT_CLIP	–111	Software clipping

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Escape Function Identifiers:		
GKS$K_ESC_SET_WRITING_MODE	–150	Set writing mode
GKS$K_ESC_SET_LINE_CAP	–151	Set line cap style
GKS$K_ESC_SET_LINE_JOIN	–152	Set Line Join style
GKS$K_ESC_SET_EDGE_CTL	–153	Set edge control flag
GKS$K_ESC_SET_EDGE_TYPE	–154	Set edge type
GKS$K_ESC_SET_EDGE_WIDTH	–155	Set edge width scale factor
GKS$K_ESC_SET_EDGE_COLOR_INDEX	–156	Set edge color index
GKS$K_ESC_SET_EDGE_INDEX	–157	Set edge index
GKS$K_ESC_SET_EDGE_ASF	–158	Set edge aspect source flag (ASF)
GKS$K_ESC_BEGIN_TRANS_BLOCK	–160	Begin transformation block
GKS$K_ESC_END_TRANS_BLOCK	–161	End transformation block
GKS$K_ESC_SET_SEG_HIGH_METHOD	–162	Set segment highlighting method
GKS$K_ESC_SET_HIGH_METHOD	–163	Set highlighting method
GKS$K_ESC_SET_EDGE_REP	–200	Set edge representation
GKS$K_ESC_SET_WINDOW_TITLE	–202	Set window title
GKS$K_ESC_SET_RESET_STRING	–203	Set reset string
GKS$K_ESC_SET_CANCEL_STRING	–204	Set cancel string
GKS$K_ESC_SET_ENTER_STRING	–205	Set enter string
GKS$K_ESC_SET_ICON_BITMAPS	–206	Set icon bitmaps
GKS$K_ESC_INQ_WRITING_MODE	–251	Inquire current writing mode
GKS$K_ESC_INQ_LINE_CAP	–252	Inquire current line cap style
GKS$K_ESC_INQ_LINE_JOIN	–253	Inquire current line join style
GKS$K_ESC_INQ_EDGE_ATTR	–254	Inquire current edge attributes
GKS$K_ESC_INQ_VIEWPORT_DATA	–255	Inquire viewport data
GKS$K_ESC_INQ_SPEED	–300	Inquire current display speed
GKS$K_ESC_INQ_LIST_EDGE_INDEXES	–302	Inquire list of edge indexes
GKS$K_ESC_INQ_SEGMENT_EXTENT	–303	Inquire segment extent
GKS$K_ESC_INQ_WINDOW_IDS	–304	Inquire window identifiers
GKS$K_ESC_INQ_SEG_HIGH_METHOD	–305	Inquire segment highlighting method
GKS$K_ESC_INQ_HIGH_METHOD	–306	Inquire highlighting method
GKS$K_ESC_INQ_PASTEBOARD_ID	–307	Inquire pasteboard identifier
GKS$K_ESC_INQ_MENU_BAR_ID	–308	Inquire menu bar identifier
GKS$K_ESC_INQ_SHELL_ID	–309	Inquire shell identifier
GKS$K_ESC_INQ_LIST_ESC	–350	Inquire list of available escapes
GKS$K_ESC_INQ_DEF_SPEED	–351	Inquire default display speed
GKS$K_ESC_INQ_LINE_CAP_JOIN_FAC	–352	Inquire line cap and join facilities
GKS$K_ESC_INQ_EDGE_FAC	–354	Inquire edge facilities

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Escape Function Identifiers:		
GKS$K_ESC_INQ_PREDEF_EDGE_REP	–355	Inquire predefined edge representation
GKS$K_ESC_INQ_MAX_EDGE_BUNDLE	–356	Inquire maximum number of edge bundles
GKS$K_ESC_INQ_LIST_HIGH	–358	Inquire list of highlighting methods
GKS$K_ESC_INQ_EDGE_REP	–359	Inquire edge representation
GKS$K_ESC_MAP_NDC_OF_WC	–400	Evaluate NDC mapping of a WC point
GKS$K_ESC_MAP_DC_OF_NDC	–401	Evaluate DC mapping of an NDC point
GKS$K_ESC_MAP_WC_OF_NDC	–402	Evaluate WC mapping of an NDC point
GKS$K_ESC_MAP_NDC_OF_DC	–403	Evaluate NDC mapping of a DC point
GKS$K_ESC_INQ_GDP_EXTENT	–404	Inquire extent of a GDP
GKS$K_ESC_DOUBLE_BUFFER	–500	Set double buffering
GKS$K_ESC_SET_BCKGRND_PIXMAP	–501	Set background pixmap
GKS$K_ESC_INQ_DBUFFER_PIXMAP	–502	Inquire double buffer pixmap
GKS$K_ESC_INQ_BCKGRND_PIXMAP	–503	Inquire background pixmap
Fill Area Control Flags:		
GKS$K_ACF_POLYLINE	0	Prompt rectangle is drawn as a polyline
GKS$K_ACF_FILL_AREA	1	Prompt rectangle is drawn as a fill area
Fill Area Interior Styles:		
GKS$K_INTSTYLE_HOLLOW	0	Interior style hollow
GKS$K_INTSTYLE_SOLID	1	Interior style solid
GKS$K_INTSTYLE_PATTERN	2	Interior style pattern
GKS$K_INTSTYLE_HATCH	3	Interior style hatched
GDP Bundle Types:		
GKS$K_POLYLN_ATTRI	0	Polyline bundle attributes
GKS$K_POLYMR_ATTRI	1	Polymarker bundle attributes
GKS$K_TEXT_ATTRI	2	Text bundle attributes
GKS$K_FILLAR_ATTRI	3	Fill area bundle attributes
GDPs:		
GKS$K_GDP_DISJOINT_PLINE	–100	Disjoint polyline
GKS$K_GDP_CIRCLE_CTR_PT	–101	Circle: center and point on circumference
GKS$K_GDP_CIRCLE_3PT	–102	Circle: three points on circumference
GKS$K_GDP_CIRCLE_CTR_RAD	–103	Circle: center and radius
GKS$K_GDP_CIRCLE_2PT_RAD	–104	Circle: two points on circumference, and radius

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
GDPs:		
GKS$K_GDP_ARC_CTR_2PT	–106	Arc: center and two points on arc
GKS$K_GDP_ARC_3PT	–107	Arc: three points on circumference
GKS$K_GDP_ARC_CTR_2VEC_RAD	–108	Arc: center, two vectors, and a radius
GKS$K_GDP_ARC_2PT_RAD	–109	Arc: two points on arc and radius
GKS$K_GDP_ARC_CTR_PT_ANG	–110	Arc: center, starting point, and angle
GKS$K_GDP_ELLIPSE_CTR_AXES	–111	Ellipse: center, and two axis vectors
GKS$K_GDP_ELLIPSE_FOCII_PT	–113	Ellipse: focal points and point on circumference
GKS$K_GDP_ELIARC_CTR_AXES_2VEC	–114	Elliptic arc: center, two axis vectors, and two vectors
GKS$K_GDP_ELIARC_FOCII_2PT	–116	Elliptic arc: focal points and two points on circumference
GKS$K_GDP_RECT_2PT	–125	Rectangle: two corners
GKS$K_GDP_FILL_AREA_SET	–332	Fill area set
GKS$K_GDP_FCIRCLE_CTR_PT	–333	Filled circle: center and point on circumference
GKS$K_GDP_FCIRCLE_3PT	–334	Filled circle: three points on circumference
GKS$K_GDP_FCIRCLE_CTR_RAD	–335	Filled circle: center and radius
GKS$K_GDP_FCIRCLE_2PT_RAD	–336	Filled circle: two points on circumference, and radius
GKS$K_GDP_FARC_CTR_2PT	–338	Filled arc: center and two points on arc
GKS$K_GDP_FARC_3PT	–339	Filled arc: three points on circumference
GKS$K_GDP_FARC_CTR_2VEC_RAD	–340	Filled arc: center, two vectors, and a radius
GKS$K_GDP_FARC_2PT_RAD	–341	Filled arc: two points on arc, and radius
GKS$K_GDP_FARC_CTR_PT_ANG	–342	Filled arc: center, starting point, and angle
GKS$K_GDP_FELLIPSE_CTR_AXES	–343	Filled ellipse: center, and two axis vectors
GKS$K_GDP_FELLIPSE_FOCII_PT	–345	Filled ellipse: focal points and point on circumference
GKS$K_GDP_FELIARC_CTR_AXES_2VEC	–346	Filled elliptic arc: center, two axis vectors, and two vectors
GKS$K_GDP_FELIARC_FOCII_2PT	–348	Filled elliptic arc: focal points and two points on circumference
GKS$K_GDP_FRECT_2PT	–349	Filled rectangle: two corners
GKS$K_GDP_IMAGE_ARRAY	–400	Packed cell array

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
GKS Level Types:		
GKS$K_LEVEL_MA	–3	Minimal output, no input
GKS$K_LEVEL_MP	–2	Minimal output, request input
GKS$K_LEVEL_MC	–1	Minimal output, full input
GKS$K_LEVEL_0A	0	All primitives and attributes, no input
GKS$K_LEVEL_0B	1	All primitives and attributes, request input
GKS$K_LEVEL_0C	2	All primitives and attributes, full input
GKS$K_LEVEL_1A	3	Basic segmentation with full output, no input
GKS$K_LEVEL_1B	4	Basic segmentation with full output, request input
GKS$K_LEVEL_1C	5	Basic segmentation with full output, full input
GKS$K_LEVEL_2A	6	Workstation-independent and segment storage, no input
GKS$K_LEVEL_2B	7	Workstation-independent and segment storage, request input
GKS$K_LEVEL_2C	8	Workstation-independent and segment storage, full input
GKS Operating States:		
GKS$K_GKCL	0	GKS closed
GKS$K_GKOP	1	GKS open
GKS$K_WSOP	2	At least one workstation open
GKS$K_WSAC	3	At least one workstation active
GKS$K_SGOP	4	At least one segment open
Highlighting Flags:		
GKS$K_NORMAL	0	Primitives not highlighted
GKS$K_HIGHLIGHTED	1	Primitives highlighted
Highlighting Methods:		
GKS$K_HIGH_METHOD_DEFAULT	0	Default highlighting
GKS$K_HIGH_METHOD_COMP	1	Highlight with complement mode
GKS$K_HIGH_METHOD_COLOR	2	Highlight with color
GKS$K_HIGH_METHOD_LINE	3	Highlight with extent line box
GKS$K_HIGH_METHOD_FILL	4	Highlight with extent fill area
GKS$K_HIGH_METHOD_DUAL	5	Highlight with extent line box and fill area

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Implicit Regeneration States:		
GKS$K_IRG_SUPPRESSED	0	Implicit regeneration suppressed
GKS$K_IRG_ALLOWED	1	Implicit regeneration allowed
Input Classes:		
GKS$K_INPUT_CLASS_NONE	0	No input class
GKS$K_INPUT_CLASS_LOCATOR	1	Locator input class
GKS$K_INPUT_CLASS_STROKE	2	Stroke input class
GKS$K_INPUT_CLASS_VALUATOR	3	Valuator input class
GKS$K_INPUT_CLASS_CHOICE	4	Choice input class
GKS$K_INPUT_CLASS_PICK	5	Pick input class
GKS$K_INPUT_CLASS_STRING	6	String input class
GKS$K_INPUT_CLASS_VIEWPORT	7	Viewport input class
Input Device Type:		
GKS$K_INPUT_DEV_DEFAULT	0	Default input device
Input Mode Types:		
GKS$K_INPUT_MODE_REQUEST	0	Request mode
GKS$K_INPUT_MODE_SAMPLE	1	Sample mode
GKS$K_INPUT_MODE_EVENT	2	Event mode
Input Priority States:		
GKS$K_INPUT_PRIORITY_HIGHER	0	Relative input priority higher
GKS$K_INPUT_PRIORITY_LOWER	1	Relative input priority lower
Line Cap Types:		
GKS$K_LINE_CAP_DEFAULT	1	Default line cap
GKS$K_LINE_CAP_BUTT	2	Line cap type butted
GKS$K_LINE_CAP_ROUND	3	Line cap type rounded
GKS$K_LINE_CAP_SQUARE	4	Line cap type square
Line Join Types:		
GKS$K_LINE_JOIN_DEFAULT	2	Default line join
GKS$K_LINE_JOIN_MITRE	3	Line join type miter
GKS$K_LINE_JOIN_ROUND	4	Line join type round
GKS$K_LINE_JOIN_BEVEL	5	Line join type bevel

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Line Types (Standard):		
GKS$K_LINETYPE_SOLID	1	Line type solid
GKS$K_LINETYPE_DASHED	2	Line type dashed
GKS$K_LINETYPE_DOTTED	3	Line type dotted
GKS$K_LINETYPE_DASHED_DOTTED	4	Line type dash-dotted
Line Types (DEC GKS Implementation-Specific):		
GKS$K_LINETYPE_DASH_2_DOT	–1	Line type double-dashed dotted
GKS$K_LINETYPE_DASH_3_DOT	–2	Line type triple-dashed dotted
GKS$K_LINETYPE_LONG_DASH	–3	Line type long-dashed
GKS$K_LINETYPE_LONG_SHORT_DASH	–4	Line type long-short-dashed
GKS$K_LINETYPE_SPACED_DASH	–5	Line type spaced-dashed
GKS$K_LINETYPE_SPACED_DOT	–6	Line type spaced-dotted
GKS$K_LINETYPE_DOUBLE_DOT	–7	Line type double dotted
GKS$K_LINETYPE_TRIPLE_DOT	–8	Line type triple dotted
GKS$K_LINETYPE_CENTER	–9	Line type long-dash-long
GKS$K_LINETYPE_PHANTOM	–10	Line type long-dash-dash-long
Logical Types:		
GKS$K_FALSE	0	Logical FALSE
GKS$K_TRUE	1	Logical TRUE
Marker Types (Standard):		
GKS$K_MARKERTYPE_DOT	1	Marker type dot (.)
GKS$K_MARKERTYPE_PLUS	2	Marker type plus (+)
GKS$K_MARKERTYPE_ASTERISK	3	Marker type asterisk (*)
GKS$K_MARKERTYPE_CIRCLE	4	Marker type circle (O)
GKS$K_MARKERTYPE_DIAGONAL_CROSS	5	Marker type diagonal cross (X)
Marker Types (DEC GKS Implementation-Specific):		
GKS$K_MARKERTYPE_SOLID_CIRCLE	–1	Marker type solid circle
GKS$K_MARKERTYPE_TRIANGLE_UP	–2	Marker type hollow up triangle
GKS$K_MARKERTYPE_SOLID_TRI_UP	–3	Marker type solid up triangle
GKS$K_MARKERTYPE_TRIANGLE_DOWN	–4	Marker type hollow down triangle
GKS$K_MARKERTYPE_SOLID_TRI_DOWN	–5	Marker type solid down triangle
GKS$K_MARKERTYPE_SQUARE	–6	Marker type hollow square
GKS$K_MARKERTYPE_SOLID_SQUARE	–7	Marker type solid square
GKS$K_MARKERTYPE_BOWTIE	–8	Marker type hollow bow tie

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Marker Types (DEC GKS Implementation-Specific):		
GKS$K_MARKERTYPE_SOLID_BOWTIE	–9	Marker type solid bow tie
GKS$K_MARKERTYPE_HOURGLASS	–10	Marker type hollow hourglass
GKS$K_MARKERTYPE_SOLID_HGLASS	–11	Marker type solid hourglass
GKS$K_MARKERTYPE_DIAMOND	–12	Marker type hollow diamond
GKS$K_MARKERTYPE_SOLID_DIAMOND	–13	Marker type solid diamond
New Frame Action States:		
GKS$K_NEWFRAME_NOTNECESSARY	0	No new frame action on update
GKS$K_NEWFRAME_NECESSARY	1	New frame action on update
Pick Status Types:		
GKS$K_STATUS_NONE	0	No input obtained
GKS$K_STATUS_OK	1	Input obtained
GKS$K_STATUS_NOPICK	2	Input is NOPICK
Regeneration Flag States:		
GKS$K_POSTPONE_FLAG	0	Implicit regeneration postponed
GKS$K_PERFORM_FLAG	1	Implicit regeneration performed
Returned Type Values:		
GKS$K_VALUE_SET	0	Type of returned value set
GKS$K_VALUE_REALIZED	1	Type of returned value realized
Simultaneous Events Flags:		
GKS$K_NOMORE_EVENTS	0	No more simultaneously generated events
GKS$K_MORE_EVENTS	1	More simultaneously generated events
Text Horizontal Alignment Types:		
GKS$K_TEXT_HALIGN_NORMAL	0	Horizontal align normal
GKS$K_TEXT_HALIGN_LEFT	1	Horizontal align left
GKS$K_TEXT_HALIGN_CENTRE	2	Horizontal align center
GKS$K_TEXT_HALIGN_RIGHT	3	Horizontal align right

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Text Path Types:		
GKS$K_TEXT_PATH_RIGHT	0	Path right
GKS$K_TEXT_PATH_LEFT	1	Path left
GKS$K_TEXT_PATH_UP	2	Path up
GKS$K_TEXT_PATH_DOWN	3	Path down
Text Precision Types:		
GKS$K_TEXT_PRECISION_STRING	0	Text precision string
GKS$K_TEXT_PRECISION_CHAR	1	Text precision character
GKS$K_TEXT_PRECISION_STROKE	2	Text precision stroke
Text Vertical Alignment Types:		
GKS$K_TEXT_VALIGN_NORMAL	0	Vertical align normal
GKS$K_TEXT_VALIGN_TOP	1	Vertical align top
GKS$K_TEXT_VALIGN_CAP	2	Vertical align cap
GKS$K_TEXT_VALIGN_HALF	3	Vertical align half
GKS$K_TEXT_VALIGN_BASE	4	Vertical align base
GKS$K_TEXT_VALIGN_BOTTOM	5	Vertical align bottom
Update States:		
GKS$K_NOTPENDING	0	Not pending
GKS$K_PENDING	1	Pending
Visibility Flags:		
GKS$K_INVISIBLE	0	Set to invisible
GKS$K_VISIBLE	1	Set to visible
Workstation Category Types:		
GKS$K_WSCAT_OUTPUT	0	Output
GKS$K_WSCAT_INPUT	1	Input
GKS$K_WSCAT_OUTIN	2	Output/input
GKS$K_WSCAT_WISS	3	Workstation independent segment storage
GKS$K_WSCAT_MO	4	Metafile output
GKS$K_WSCAT_MI	5	Metafile input

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Workstation Class Types:		
GKS$K_WSCLASS_VECTOR	0	Vector
GKS$K_WSCLASS_RASTER	1	Raster
GKS$K_WSCLASS_OTHERD	2	Other device
Workstation Color Availability States:		
GKS$K_MONOCHROME	0	Monochrome
GKS$K_COLOR	1	Color
Workstation States:		
GKS$K_WS_INACTIVE	0	Inactive
GKS$K_WS_ACTIVE	1	Active
Workstation Types:		
GKS$K_WSTYPE_DEFAULT	0	Default workstation type
GKS$K_GKSM_OUTPUT	2	GKS output metafile
GKS$K_GKSM_INPUT	3	GKS input metafile
GKS$K_WSTYPE_WISS	5	GKS workstation independent segment storage
GKS$K_CGM_OUTPUT	7	CGM output metafile
GKS$K_VT_OUTPUT	10	Digital VT125 (output only)
GKS$K_VT125BW_OUTPUT	10	Monochrome Digital VT125
GKS$K_VT125	11	Digital VT125 with color option
GKS$K_VT125BW	12	Monochrome Digital VT125
GKS$K_VT240	13	Digital VT240 with color option
GKS$K_VT240BW	14	Monochrome Digital VT240
GKS$K_LCP01	15	Digital LCP01 printer
GKS$K_LCG01	15	Digital LCG01 printer
GKS$K_VT330	16	Digital VT330 (monochrome)
GKS$K_VT340	17	Digital VT340 (color)
GKS$K_LA34	31	Digital LA34 with graphics option
GKS$K_LA100	31	Digital LA100
GKS$K_LA50	32	Digital LA50 with 2:1 aspect ratio
GKS$K_LA210	34	Digital LA210
GKS$K_LA75	35	Digital LA75
GKS$K_LNO3_PLUS	38	Digital LN03 PLUS
GKS$K_VSII	41	Digital VAXstation II
GKS$K_VSII_GPX	41	Digital VAXstation II/GPX

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Workstation Types:		
GKS$K_VS2000	41	Digital VAXstation 2000
GKS$K_VS3200	41	Digital VAXstation 3200
GKS$K_VS3500	41	Digital VAXstation 3500
GKS$K_LVP16A	51	Digital LVP16 color plotter (8.5×11 inches)
GKS$K_HP7475	51	Hewlett-Packard HP7475
GKS$K_LVP16B	52	Digital LVP16 color plotter (11×17 inches)
GKS$K_HP7550	53	Hewlett-Packard HP7550 pen plotter
GKS$K_HP7580	54	Hewlett-Packard HP7580 pen plotter
GKS$K_LG_MPS2000	55	LaserGraphics MPS–2000 film recorder
GKS$K_HP7585	56	Hewlett-Packard HP7585 pen plotter
GKS$K_POSTSCRIPT	61	PostScript graphics handler
GKS$K_COLOR_POSTSCRIPT	62	Color PostScript graphics handler
GKS$K_EPSF	65	Encapsulated PostScript
GKS$K_COLOR_EPSF	66	Color Encapsulated PostScript
GKS$K_TEK4014_OUTPUT	70	Tektronix–4014 (output only)
GKS$K_TEK4014	72	Tektronix–4014
GKS$K_TEK4107_OUTPUT	80	Tektronix–4107 (output only)
GKS$K_TEK4107	82	Tektronix–4107
GKS$K_TEK4207_OUTPUT	83	Tektronix 4207 (output only)
GKS$K_TEK4207	84	Tektronix 4207
GKS$K_TEK4128_OUTPUT	85	Tektronix 4128 (output only)
GKS$K_TEK4128	86	Tektronix 4128
GKS$K_TEK4129_OUTPUT	87	Tektronix 4129 (output only)
GKS$K_VS500_OUTPUT	87	VS500 (output only)
GKS$K_TEK4129	88	Tektronix 4129
GKS$K_VS500	88	VS500 interactive
GKS$K_LJ250	91	Digital LJ250 90 DPI
GKS$K_LJ250_180DPI	92	Digital LJ250 180 DPI
GKS$K_LA324	93	Digital LJ250 ink jet and LA324 printer
GKS$K_DECWINDOWS_OUTPUT	210	DECwindows output
GKS$K_DECWINDOWS	211	DECwindows
GKS$K_DECWINDOWS_DRAWABLE	212	DECwindows drawable
GKS$K_DECWINDOWS_WIDGET	213	DECwindows widget
GKS$K_PEX_OUTPUT	220	PEX output
GKS$K_PEX	221	PEX output and input

(continued on next page)

Table B–1 (Cont.) GKS$ Constants

Constant	Value	Description
Workstation Types:		
GKS$K_PEX_DRAWABLE	222	PEX drawable
GKS$K_PEX_WIDGET	223	PEX widget
GKS$K_MOTIF_OUTPUT	230	Motif output
GKS$K_MOTIF	231	Motif
GKS$K_MOTIF_DRAWABLE	232	Motif drawable
GKS$K_MOTIF_WIDGET	233	Motif widget
GKS$K_PMOTIF_OUTPUT	240	Motif PEX output
GKS$K_PMOTIF	241	Motif PEX input/output
GKS$K_PMOTIF_DRAWABLE	242	Motif PEX drawable (output only)
GKS$K_PMOTIF_WIDGET	243	Motif PEX widget (input/output)
GKS$K_DDIF	250	DDIF
Writing Modes:		
GKS$K_WRT_MODE_DEFAULT	1	Default writing mode
GKS$K_WRT_MODE_COMPLEMENT	2	Complement writing mode
GKS$K_WRT_MODE_ERASE	3	Erase writing mode
GKS$K_WRT_MODE_OVERLAY	4	Overlay writing mode

Program Example

Insert tabbed divider here. Then discard this sheet.

C
Program Example

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 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
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	Example 7–4

(continued on next page)

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

Function	Example
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
SET TEXT PATH	Example 6–4
SET VALUATOR MODE	Example 9–4

(continued on next page)

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

Function	Example
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.

D

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
- ERROR LOGGING
- ERROR HANDLING
- GET STRING
- GET STROKE
- INITIALIZE STRING
- INQUIRE DEFAULT CHOICE DEVICE DATA
- INQUIRE DEFAULT LOCATOR DEVICE DATA
- INQUIRE DEFAULT PICK DEVICE DATA
- INQUIRE DEFAULT STRING DEVICE DATA
- INQUIRE DEFAULT STROKE DEVICE DATA
- INQUIRE DEFAULT VALUATOR DEVICE DATA

- INQUIRE FILL AREA FACILITIES
- INQUIRE GENERALIZED DRAWING PRIMITIVE
- INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES
- INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
- INQUIRE POLYLINE FACILITIES
- INQUIRE POLYMARKER FACILITIES
- INQUIRE PREDEFINED PATTERN REPRESENTATION
- INQUIRE LIST OF COLOUR INDICES
- INQUIRE LIST OF FILL AREA INDICES
- INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS
- INQUIRE LIST OF PATTERN INDICES
- INQUIRE LIST OF POLYLINE FACILITIES
- INQUIRE LIST OF POLYMARKER FACILITIES
- INQUIRE LIST OF TEXT FACILITIES
- INQUIRE PATTERN REPRESENTATION
- INQUIRE PIXEL ARRAY
- INQUIRE SET OF ACTIVE WORKSTATIONS
- INQUIRE SET OF ASSOCIATED WORKSTATIONS
- INQUIRE SET OF OPEN WORKSTATIONS
- INQUIRE SET OF SEGMENT NAMES IN USE
- INQUIRE SET OF SEGMENT NAMES ON WORKSTATION
- INQUIRE STRING DEVICE STATE
- INQUIRE TEXT EXTENT
- INQUIRE TEXT FACILITIES
- INQUIRE WORKSTATION CONNECTION AND TYPE
- MESSAGE
- OPEN GKS
- OPEN WORKSTATION
- REQUEST STRING
- REQUEST STROKE
- SAMPLE STRING
- SAMPLE STROKE
- SET PATTERN REPRESENTATION
- TEXT

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 BUILDESC 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 BUILDESC, 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.

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
;
```

(continued on next page)

Example D–1 (Cont.) Macro Subroutine Used to Build Array Descriptors

```
      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):

```
      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).

         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.
```

(continued on next page)

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

```
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 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.

E

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

E.2 Initial Polymarker Attributes

This section lists the initial values for the polymarker attributes.

Attribute	Initial Value	Description
Polymarker index	1	Polymarker bundle number 1
Marker type	GKS$K_ MARKERTYPE_ ASTERISK	Asterisk for marker
Marker size	1.0	Nominal size
Color index	1	Workstation dependent value
Marker type ASF	GKS$K_ASF_ INDIVIDUAL	Use current marker type
Marker size ASF	GKS$K_ASF_ INDIVIDUAL	Use current marker size
Marker color index ASF	GKS$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 GKS$K_TEXT_PRECISION_ STRING	Hardware font 1, string precision
Character expansion factor	1.0	Width-to-height ratio from font file
Character spacing	0.0	Adjacent character bodies
Color index	1	Workstation dependent value
Text font and precision ASF	GKS$K_ASF_INDIVIDUAL	Use current font and precision
Character expansion factor ASF	GKS$K_ASF_INDIVIDUAL	Use current width and height ratio
Character spacing ASF	GKS$K_ASF_INDIVIDUAL	Use current character space
Text color index ASF	GKS$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

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	GKS$K_INTSTYLE_ HOLLOW	Boundary of polygonal area
Style index	1	Workstation-dependent pattern or hatch style
Color index	1	Workstation-dependent value
Interior style ASF	GKS$K_ASF_ INDIVIDUAL	Use current interior style
Style index ASF	GKS$K_ASF_ INDIVIDUAL	Use current pattern or hatch style
Color index ASF	GKS$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)	Pattern starting point in WC units

E.5 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	GKS$K_VISIBLE	The segment is visible.
Highlighting	GKS$K_NORMAL	The segment is not highlighted.
Segment priority	0.0	The segment has the lowest priority.
Detectability	GKS$K_ UNDETECTABLE	The segment is undetectable.

The default segment transformation is called the identity transformation. The identity transformation uses a 2×3 matrix as follows:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}$$

E.6 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 rectangular, begin with a lower left corner point of (0.0, 0.0), and extend to the value 1.0 on both the X and Y axes.

Initially, clipping is enabled (GKS$K_CLIP) at the normalization viewport limit.

Differences in GKS

Implementations

Insert tabbed divider here. Then discard this sheet.

F

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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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)

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 *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* for information on fonts.
Number of GDPs	DEC GKS has 31 GDPs. See the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* for more information on the supported GDPs.
Number of escapes	DEC GKS has 61 escape functions. See the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* 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 *DEC GKS User's Guide* for a detailed description of the metafile format.
Font definitions	See the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* for complete information on fonts supported by DEC GKS.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W