DEC GKS GKS$
Binding Reference Manual

Order Number: AA–MJ31A–TE

April 1989

This manual is a reference to the DEC GKS GKS$ binding functions. It contains information about the DEC GKS GKS$ binding control, output, output attribute, transformation, segment, input, inquiry, metafile, and error-handling functions.

Revision/Update Information:	This is a new manual.
Operating System and Version:	VMS Version 4.7 or higher. ULTRIX Version 3.0 or higher. VAXstation requirement: VAXstation Windowing Software Versions 3.1 or higher, or DECwindows Version 1.0.
Software Version:	DEC GKS Version 4.0

digital equipment corporation
maynard, massachusetts

First Printing, March 1984
Revised, November 1984, May 1986, March 1987, and April 1989

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

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

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

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

The following are trademarks of Digital Equipment Corporation:

ALL–IN–1	EduSystem	RT
DEC	IAS	ULTRIX
DEC/CMS	MASSBUS	UNIBUS
DEC/MMS	PDP	VAX
DECnet	PDT	VAXcluster
DECmate	P/OS	VMS
DECsystem–10	Professional	VT
DECSYSTEM–20	Q–bus	Work Processor
DECUS	Rainbow	
DECwriter	RSTS	
DIBOL	RSX	digital™

ZK4633

Contents

Preface xi

Chapter 1 Introduction to DEC GKS

1.1 GKS Levels 1–1

1.2 Programming Considerations 1–2
1.2.1 Online Help 1–2
1.2.2 Capabilities of Supported Devices 1–2
1.2.3 Calling Sequences 1–3
1.2.4 Constants and Include Files 1–4
1.2.4.1 Including Definition Files 1–5

1.3 Function Syntax 1–5
1.3.1 Argument Descriptions 1–6

1.4 GKS$ Binding Data Types 1–6

1.5 GKS Function Organization 1–7

1.6 Using User-Defined Error-Handling Functions 1–7

1.7 Standard Escape/GDP Data Records 1–8

Chapter 2 Compiling, Linking, and Running DEC GKS Programs on VMS

2.1 **Compiling, Linking, and Running** 2–2

Chapter 3 Compiling, Linking, and Running DEC GKS Programs on ULTRIX

3.1 **ULTRIX Programming Considerations** 3–1
3.1.1 Supported Languages 3–2
3.1.2 Capabilities of Supported Devices 3–2
3.1.3 Calling Sequences 3–2
3.1.4 Constants and Include Files 3–4
3.1.4.1 Including Definition Files 3–5
3.1.5 Compiling and Linking GKS$ Programs 3–6
3.1.6 Environment Variables and DEC GKS Programming 3–6
3.1.6.1 Specifying Bit Masks as Workstation Type Values 3–7

Chapter 4 Control Functions

ACTIVATE WORKSTATION 4–2
CLEAR WORKSTATION 4–3
CLOSE GKS 4–4
CLOSE WORKSTATION 4–5
DEACTIVATE WORKSTATION 4–6
ESCAPE 4–7
MESSAGE 4–8
OPEN GKS 4–9
OPEN WORKSTATION 4–10
REDRAW ALL SEGMENTS ON WORKSTATION 4–11
SET DEFERRAL STATE 4–12
UPDATE WORKSTATION 4–13

Chapter 5 Output Functions

CELL ARRAY 5–2
FILL AREA 5–3
GENERALIZED DRAWING PRIMITIVE 5–4
POLYLINE 5–5
POLYMARKER 5–6
TEXT 5–7

Chapter 6 Output Attribute Functions

SET ASPECT SOURCE FLAGS 6–2
SET CHARACTER EXPANSION FACTOR 6–3
SET COLOR REPRESENTATION 6–4
SET FILL AREA COLOR INDEX 6–5
SET FILL AREA INDEX 6–6
SET FILL AREA INTERIOR STYLE 6–7
SET FILL AREA REPRESENTATION 6–8
SET FILL AREA STYLE INDEX 6–9
SET LINETYPE 6–10
SET LINEWIDTH SCALE FACTOR 6–11
SET MARKER SIZE SCALE FACTOR 6–12
SET MARKER TYPE 6–13
SET PATTERN REFERENCE POINT 6–14
SET PATTERN REPRESENTATION 6–15
SET PATTERN SIZE 6–16
SET PICK IDENTIFIER 6–17
SET POLYLINE COLOR INDEX 6–18
SET POLYLINE INDEX 6–19
SET POLYLINE REPRESENTATION 6–20
SET POLYMARKER COLOR INDEX 6–21
SET POLYMARKER INDEX 6–22
SET POLYMARKER REPRESENTATION 6–23
SET TEXT ALIGNMENT 6–25
SET TEXT COLOR INDEX 6–26
SET TEXT FONT AND PRECISION 6–27
SET TEXT HEIGHT 6–28
SET TEXT INDEX 6–29
SET TEXT PATH 6–30
SET TEXT REPRESENTATION 6–31
SET TEXT SPACING 6–32
SET TEXT UP VECTOR 6–33

Chapter 7 Transformation Functions

SELECT NORMALIZATION TRANSFORMATION 7–2
SET CLIPPING INDICATOR 7–3
SET VIEWPORT 7–4
SET VIEWPORT INPUT PRIORITY 7–5
SET WINDOW 7–6
SET WORKSTATION VIEWPORT 7–7
SET WORKSTATION WINDOW 7–8

Chapter 8 Input Functions

AWAIT EVENT 8–2
FLUSH DEVICE EVENTS 8–3
GET CHOICE 8–4
GET LOCATOR 8–5
GET PICK 8–6
GET STRING 8–7
GET STROKE 8–8
GET VALUATOR 8–9
INITIALIZE CHOICE 8–10
INITIALIZE LOCATOR 8–11
INITIALIZE PICK 8–12
INITIALIZE STRING 8–13
INITIALIZE STROKE 8–14
INITIALIZE VALUATOR 8–16
REQUEST CHOICE 8–17
REQUEST LOCATOR 8–18
REQUEST PICK 8–19
REQUEST STRING 8–20
REQUEST STROKE 8–21
REQUEST VALUATOR 8–23
SAMPLE CHOICE 8–24
SAMPLE LOCATOR 8–25
SAMPLE PICK 8–26
SAMPLE STRING 8–27
SAMPLE STROKE 8–28
SAMPLE VALUATOR 8–30
SET CHOICE MODE 8–31
SET LOCATOR MODE 8–32
SET PICK MODE 8–33
SET STRING MODE 8–34
SET STROKE MODE 8–35
SET VALUATOR MODE 8–36

Chapter 9 Segment Functions

ASSOCIATE SEGMENT WITH WORKSTATION 9–2
CLOSE SEGMENT 9–3
COPY SEGMENT TO WORKSTATION 9–4
CREATE SEGMENT 9–5
DELETE SEGMENT 9–6
DELETE SEGMENT FROM WORKSTATION 9–7
INSERT SEGMENT 9–8

RENAME SEGMENT 9–9
SET DETECTABILITY 9–10
SET HIGHLIGHTING 9–11
SET SEGMENT PRIORITY 9–12
SET VISIBILITY 9–13
SET SEGMENT TRANSFORMATION 9–14

Chapter 10 Metafile, Error-Handling, and Utility Functions

GET ITEM TYPE FROM GKSM 10–2
INTERPRET ITEM 10–3
READ ITEM FROM GKSM 10–4
WRITE ITEM TO GKSM 10–5
EMERGENCY CLOSE GKS 10–6
ERROR HANDLING 10–7
ERROR LOGGING 10–8
ACCUMULATE TRANSFORMATION MATRIX 10–9
EVALUATE TRANSFORMATION MATRIX 10–11

Chapter 11 Inquiry Functions

INQUIRE LEVEL OF GKS 11–2
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER 11–3
INQUIRE WORKSTATION MAXIMUM NUMBERS 11–4
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES 11–5
INQUIRE COLOR FACILITIES 11–6
INQUIRE DEFAULT CHOICE DATA 11–7
INQUIRE DEFAULT DEFERRAL STATE VALUES 11–9
INQUIRE DEFAULT LOCATOR DEVICE DATA 11–10
INQUIRE DEFAULT PICK DEVICE DATA 11–12
INQUIRE DEFAULT STRING DEVICE DATA 11–14
INQUIRE DEFAULT STROKE DEVICE DATA 11–16
INQUIRE DEFAULT VALUATOR DEVICE DATA 11–18
INQUIRE DISPLAY SPACE SIZE 11–20
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES . . 11–21
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 11–23
INQUIRE FILL AREA FACILITIES 11–25
INQUIRE GENERALIZED DRAWING PRIMITIVE 11–27
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVE 11–28
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLE . . 11–29

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES 11–30
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED 11–31
INQUIRE PATTERN FACILITIES 11–32
INQUIRE POLYLINE FACILITIES 11–33
INQUIRE POLYMARKER FACILITIES 11–35
INQUIRE PREDEFINED COLOR REPRESENTATION 11–37
INQUIRE PREDEFINED FILL AREA REPRESENTATION 11–38
INQUIRE PREDEFINED PATTERN REPRESENTATION 11–39
INQUIRE PREDEFINED POLYLINE REPRESENTATION 11–41
INQUIRE PREDEFINED POLYMARKER REPRESENTATION 11–42
INQUIRE PREDEFINED TEXT REPRESENTATION 11–43
INQUIRE TEXT FACILITIES 11–44
INQUIRE WORKSTATION CATEGORY 11–46
INQUIRE WORKSTATION CLASSIFICATION 11–47
INQUIRE CLIPPING 11–48
INQUIRE INPUT QUEUE OVERFLOW 11–49
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES 11–50
INQUIRE MORE SIMULTANEOUS EVENTS 11–52
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER 11–53
INQUIRE CURRENT PICK IDENTIFIER 11–54
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 11–55
INQUIRE NAME OF OPEN SEGMENT 11–57
INQUIRE OPERATING STATE VALUE 11–58
INQUIRE NORMALIZATION TRANSFORMATION NUMBER 11–59
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS 11–60
INQUIRE SET OF ACTIVE WORKSTATIONS 11–61
INQUIRE SET OF OPEN WORKSTATIONS 11–62
INQUIRE SET OF SEGMENT NAMES IN USE 11–63
INQUIRE CHOICE DEVICE STATE 11–64
INQUIRE COLOR REPRESENTATION 11–66
INQUIRE FILL AREA REPRESENTATION 11–67
INQUIRE LIST OF COLOR INDEXES 11–68
INQUIRE LIST OF FILL AREA INDEXES 11–69
INQUIRE LIST OF PATTERN INDEXES 11–70
INQUIRE LIST OF POLYLINE INDEXES 11–71
INQUIRE LIST OF POLYMARKER INDEXES 11–72
INQUIRE LIST OF TEXT INDEXES 11–73
INQUIRE LOCATOR DEVICE STATE 11–74
INQUIRE PATTERN REPRESENTATION 11–76
INQUIRE PICK DEVICE STATE 11–78
INQUIRE POLYLINE REPRESENTATION 11–80
INQUIRE POLYMARKER REPRESENTATION 11–82

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION 11–84
INQUIRE STRING DEVICE STATE 11–85
INQUIRE STROKE DEVICE STATE 11–87
INQUIRE TEXT EXTENT 11–89
INQUIRE TEXT REPRESENTATION 11–91
INQUIRE VALUATOR DEVICE STATE 11–93
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES 11–95
INQUIRE WORKSTATION CONNECTION AND TYPE 11–97
INQUIRE WORKSTATION STATE 11–98
INQUIRE WORKSTATION TRANSFORMATION 11–99
INQUIRE SEGMENT ATTRIBUTES 11–101
INQUIRE LIST OF ASSOCIATED WORKSTATIONS 11–102
INQUIRE PIXEL 11–103
INQUIRE PIXEL ARRAY 11–104
INQUIRE PIXEL ARRAY DIMENSIONS 11–106

Appendix A DEC GKS Function Names and FORTRAN Binding Function Names

A.1 DEC GKS Function Names and FORTRAN Binding Function Names A–1

Appendix B DEC GKS Error Messages

B.1 Implementation-Specific Errors B–2

B.2 Operating State Errors B–16

B.3 Workstation Errors B–18

B.4 Transformation Function Errors B–23

B.5 Output Attribute Errors B–24

B.6 Output Function Errors B–31

B.7 Segment Function Errors B–32

B.8 Input Function Errors B–34

B.9 Metafile Function Errors B–37

B.10 Escape Function Errors B–39

B.11 Miscellaneous Errors B–39

B.12 System Errors B–40

B.13 FORTRAN Binding Errors B–41

Index

Tables

A–1 DEC GKS Function Names and Corresponding FORTRAN Binding Names A–1

Preface

Manual Objectives

This manual provides reference to the DEC GKS Graphical Kernel System (GKS) GKS$ binding functions. DEC GKS is a level 2c GKS implementation. For more information concerning GKS implementation levels, refer to Chapter 1, Introduction to DEC GKS.

NOTE

Before reading this manual, you should review the DEC GKS release notes by typing the following:

```
$ HELP  GKS  RELEASE_NOTES RETURN
```

Intended Audience

This manual is intended for experienced application programmers who need to reference information (binding syntax and and brief argument descriptions) concerning the DEC GKS GKS$ binding functions. Readers should be familiar with the DIGITAL Command Language (DCL). (For more information concerning DCL, refer to the *VAX/VMS DCL Dictionary*.) about GKS.

This manual is not tutorial in nature. New users who need tutorial information and moderately experienced users needing programming suggestions should refer to the *DEC GKS User Manual*.

Document Structure

This manual is contained in one volume, with the following information contained in its chapters:

- Chapter 1, Introduction to DEC GKS, provides an introduction to the DEC GKS GKS$ binding, including information about function syntax, data types, passing mechanisms, and manual organization.
- Chapter 2, Compiling, Linking, and Running DEC GKS Programs on VMS, provides VMS specific information to compile, link, and edit your programs.
- Chapter 3, Compiling, Linking, and Running DEC GKS Programs on ULTRIX, provides ULTRIX specific information to compile, link, and edit your programs.
- Chapter 4, Control Functions, provides information concerning the establishment of the DEC GKS and workstation environments.
- Chapter 5, Output Functions, provides information concerning the generation of output primitives.
- Chapter 6, Output Attribute Functions, provides information concerning the generation of output attributes.
- Chapter 7, Transformation Functions, provides information concerning the normalization and workstation transformations.
- Chapter 8, Input Functions, provides information concerning input.
- Chapter 9, Segment Functions, provides information concerning the storage of output primitives in segments.
- Chapter 10, Metafile, Error-Handling, and Utility Functions, provides information concerning long-term storage of graphical images, error-handling, and utility functions.
- Chapter 11, Inquiry Functions, provides information concerning the acquisition of DEC GKS and workstation status information.
- Appendix A, DEC GKS Function Names and FORTRAN Binding Function Names, provides information concerning all DEC GKS$ and function name cross references.
- Appendix B, DEC GKS Error Messages, provides information concerning all DEC GKS error messages.

Associated Documents

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

- *DEC GKS User Manual*—For programmers who need tutorial information or guides to programming technique.
- *DEC GKS Reference Manual*—For programmers who need encyclopedic information about GKS functions.
- *DEC GKS FORTRAN Binding Reference Manual*—For programmers who need specific syntax and argument descriptions for the FORTRAN Binding.
- *DEC GKS GKS$ Binding Reference Manual*—For programmers who need specific syntax and argument descriptions for the GKS$ Binding.
- *DEC GKS C Binding Reference Manual*—For programmers who need specific syntax and argument descriptions for the C Binding.
- *Building a DEC GKS Workstation Handler System*—For programmers who need to build DEC GKS workstation graphics handler.
- *Building a DEC GKS Device Handler System*—For programmers who need to provide support for a device unsupported by the DEC GKS graphics handlers.
- *DEC GKS Device Specifics Reference Manual*—For programmers who need device-specific information.
- *DEC GKS Installation Guide*—For system managers who install the DEC GKS software, including the Run-Time installation, on either VMS or ULTRIX operating systems.

Conventions

Convention	Meaning
RETURN	The symbol RETURN represents a single stroke of the RETURN key on a terminal.
$ RUN GKSPROG RETURN	In interactive examples, the user's response to a prompt is printed in red; system prompts are printed in black.

Convention	Meaning
INTEGER X . . . X = 5	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.
option, . . .	A horizontal ellipsis indicates that additional arguments, options, or values can be entered. A comma that precedes the ellipsis indicates that successive items must be separated by commas.
[output-source, . . .]	Square brackets, in function synopses and a few other contexts, indicate that a syntactic element is optional.

Chapter 1

Introduction to DEC GKS

The Graphical Kernel System (GKS) is a set of graphics functions that can be used by numerous types of graphics applications to produce two-dimensional pictures on graphics output devices. GKS is defined by the ANSI X3.124-1985 and the ISO 7942-1985 standards. DEC GKS adheres to both standards. When this manual refers to the GKS standard, the reference applies to both standards.

The GKS standard provides a functional standard, and syntactical standards called *language bindings*. The functional standard determines the effects produced by a particular GKS function, but does not specify the function name or the number of function parameters. Therefore, a given function in two different GKS implementations can produce the same effects, but may have a different function name or a different number of parameters.

DEC GKS implements the functional standard using function names beginning with the prefix GKS$. These functions should be used when programming with the VMS or ULTRIX implementation of DEC GKS. If you use the GKS$ functions, you will have to edit your program if you want to transport the program across systems or across GKS implementations.

1.1 GKS Levels

The GKS standard defines levels of a GKS implementation that address the most common classes of graphic devices and application needs. The levels are determined primarily by input and output capabilities. The output level values are represented by the characters m, 0, 1, and 2. The input level values are represented by the characters a, b, and c.

The DEC GKS software is a level 2c implementation, incorporating all of the GKS output capabilities (level 2) and all of the input capabilities (level c). This manual uses the term DEC GKS when describing the 2c level DEC GKS product.

1.2 Programming Considerations

The specific method for using DEC GKS software depends on the features and conventions of each VAX language. This section discusses general issues that must be considered when using any VAX language with DEC GKS.

NOTE

Some of the VAX languages have language-specific requirements for using DEC GKS. For a complete discussion, you should refer to Appendix F, Language-Specific Programming Information, in the *DEC GKS Reference Manual*, before coding your programs. For a discussion of the capabilities of each of the DEC GKS supported physical devices, refer to the appropriate device-specific chapter in the *DEC GKS Device Specifics Reference Manual*.

1.2.1 Online Help

DEC GKS provides an online HELP library. To access this information, type the following:

```
$ HELP GKS RETURN
```

Before using the DEC GKS software, you should review the release notes for information pertinent to the current release. To review the release notes, type the following:

```
$ HELP GKS RELEASE_NOTES RETURN
```

1.2.2 Capabilities of Supported Devices

In many applications, you may wish to write completely device-independent programs. In this way, you can run your programs using different devices without having to rewrite your programs. The *DEC GKS User Manual* outlines the procedure for device-independent programming using DEC GKS.

However, you may wish to review the range of capabilities of the DEC GKS supported devices, or you may wish to write device-dependent subroutines within your application. In any instance, it is helpful to review the *DEC GKS Device Specifics Reference Manual* before you begin coding your application. This manual contains information concerning predefined bundle index representations, color capabilities, initial input values, bit masks as workstation type values, supported escape functions for that particular device, and so forth.

1.2.3 Calling Sequences

Each DEC GKS function requires a specific calling sequence. The calling sequence indicates the elements included in the language statement that calls the function, and the order of those elements. The three elements are the following:

- **Call Type**
  High-level VAX languages call DEC GKS functions with CALL statements or function references. For example, when using FORTRAN, you can use a CALL statement to call DEC GKS functions.
- **Function Identifier**
  All DEC GKS function names begin with the prefix GKS$.
  If writing programs to be transported across systems or across GKS implementations, you should use the appropriate language binding functions. Refer to the *DEC GKS FORTRAN Binding Reference Manual* and the *DEC GKS C Binding Reference Manual* for information concerning the FORTRAN and C binding function names.
- **Argument List**
  Arguments that are passed to DEC GKS functions must be listed in the order shown in the syntax descriptions contained in this manual. See Section 1.2.4.1 for more information concerning the function description format used in this manual. The various language binding functions may have an argument list that is different from the corresponding GKS$ function.

The specific requirements for writing calls and passing arguments to DEC GKS functions vary from one language to another. Whatever the language of the calling program when using the GKS$ binding, DEC GKS functions expect the following:

- Integer arguments to be 32-bit longwords passed by reference.
- Real numbers to be in single-precision, floating-point format passed by reference.

- Character strings to be passed by string descriptors.
- Arrays to be passed either by reference or by descriptor, depending on the particular DEC GKS function.

Each language may have specific requirements concerning the language-specific calling sequence. For a discussion of language-specific programming concerns, refer to Appendix F, Language-Specific Programming Information, in the *DEC GKS Reference Manual*.

1.2.4 Constants and Include Files

DEC GKS constants are symbolic names that are syntactically equivalent to literal integer constants. These constants are used in the following ways:

- As arguments to DEC GKS functions.
- As literal values to which you can compare a returned value from an inquiry function (for example, you can compare the return value, from a call to the function GKS$INQ_WS_TYPE, to the constant GKS$K_VT125).
- As literal completion status codes to which you can compare a function return value.

Many DEC GKS functions use constants as arguments, as shown by the following function call:

```
GKS$CLEAR_WS( 1, GKS$K_CLEAR_ALWAYS)
```

You can compare one of the completion status codes to a function return value, as follows:

```
   .
   .
   .
IF ( GKS$_SUCCESS = GKS$ACTIVATE_WS( 1 ) )
   .
   .
   .
```

Most DEC GKS constants begin with the prefix GKS$K and are defined in a definition file. All DEC GKS completion status code constants begin with the prefix GKS$_ERROR_ or DECGKS$_ERROR_NEG_ and are defined in a separate definition file. All DEC GKS bit mask constants begin with the prefix GKS$M_.

You can either specify a literal value as an argument to a DEC GKS function, or you can include the language definition files and use a defined constant name instead. The use of constants adds to program legibility and program documentation.

To review the list of DEC GKS constants, refer to Appendix B, DEC GKS Constants, in the *DEC GKS Reference Manual.* To review the list of DEC GKS completion status code constants, refer to Appendix B, DEC GKS Error Messages.

1.2.4.1 Including Definition Files

You use DEC GKS software primarily by placing calls to DEC GKS functions in your program. However, when using DEC GKS, you need statements in your program other than calls to GKS functions. The necessary statements depend on the VAX language you use. (For more information, refer to Appendix F, Language-Specific Programming Information, in the *DEC GKS Reference Manual.*)

DEC GKS constants and their values must be made available to all programs using DEC GKS regardless of the VAX language you use. All VAX high-level languages that use DEC GKS have a method for inserting an external file into the program source code stream at compile time. Incorporating an external file is the method for making DEC GKS constants available.

Your installation kit has been supplied with several files that contain DEC GKS constants and separate files that contain DEC GKS completion status code constants. You incorporate these files into your program with a statement that is appropriate to the language you are using.

1.3 Function Syntax

The syntax section of the function description lists the syntax of a call to the DEC GKS function. Each argument in the function is listed and separated by a comma.

All of the DEC GKS functions always return a longword condition status value. For a description of the longword status value, refer to Appendix B, DEC GKS Error Messages. For information concerning DEC GKS error handling, refer to Chapter 10, Metafile, Error-Handling, and Utility Functions, in the *DEC GKS Reference Manual.*

1.3.1 Argument Descriptions

The arguments passed to DEC GKS functions must be of specific data types and they must be passed by specific mechanisms. In the function descriptions, these data types are described in the table following each of the argument names.

For each argument, the listed values include:

- The data type of the argument
- The type of access made by the function (Input/Output)
- The argument-passing mechanism and form
- Brief argument description

Most of the passing mechanisms required by DEC GKS functions are the default mechanisms of VAX FORTRAN. (This manual clearly documents those functions requiring different passing mechanisms in the section labeled Arguments within each function description.)

1.4 GKS$ Binding Data Types

Some of the descriptions of data types in this manual are not worded in exactly the same manner as in the VMS documentation. For instance, when this manual says that an argument is of the data type "real," the corresponding VMS data type is "F_floating_point." The following list presents the notation used in this manual and the corresponding VMS notation:

GKS Type/Mechanism	Corresponding VMS Type/Mechanism
Integer	Longword integer (signed)
Real	F_floating point
String	Character-coded text string
Record (reference)	Longword integer (signed) This is an address of a data record.
Type: Array (integer) Mechanism: by reference	Type: Longword integer (signed) Mechanism: by reference, array reference

For a complete discussion of the argument-passing mechanisms, refer to the *VAX/VMS Run-Time Library Routines Reference Manual*. For information

concerning language-specific passing extensions, refer to the appropriate VAX high-level language manual.

1.5 GKS Function Organization

The GKS$ binding functions are divided into these functional categories that correspond to the following chapters:

- Control functions
- Output functions
- Attribute functions
- Transformation functions
- Input functions
- Segment functions
- Metafile functions
- Error-handling functions
- Inquiry functions
  - GKS description table functions
  - Workstation description table functions
  - GKS state list functions
  - Workstation state list functions
  - Segment functions
  - Pixel functions

1.6 Using User-Defined Error-Handling Functions

If you choose to handle errors in a different manner from the way in which the GKS$ binding handles errors, you can replace the function GKS$ERROR_HANDLER with a function of your own. In this way, you can control data storage, message generation, and execution of an error-handler in the manner most suited to your application.

To implement an application-supplied error-handling function, you must define a function with three parameters corresponding to the following values: the DEC GKS error number, the name of the DEC GKS function that generated the error, and the name of the error file. Then, you must pass the address of your error-handling function to GKS$SET_ERROR_HANDLER.

When working with the error-handling functions, you may need to specify the name of the GKS$ binding function that caused the error. The error-handling functions specify that the function identifiers must be integers.

The GKS$ binding defines a series of identifiers, which are defined to be an integer value, that correspond to each of the GKS$ binding function names.

For example, to specify the function GKS$OPEN_WS (OPEN WORKSTATION) as the function that generated the error, pass the function identifier EOPWK as an argument, as follows:

```
      .
      .
      .
C     Log an error generated by the function GKS$OPEN_WS.
      CALL GKS$LOG_ERROR( error_number, EOPWK, error_file )
      .
      .
      .
```

1.7 Standard Escape/GDP Data Records

When calling the functions GKS$ESCAPE or GKS$GDP (generalized drawing primitive), you may need to pass a data record. DEC GKS has a standard escape/GDP data record that contains up to three integer components and four array addresses.

To use an escape or GDP data record, you need to perform the following tasks:

1. Look up the escape or GDP description in Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual.*
2. Determine the size and contents of the required data record (if one is required).
3. Declare the data record as determined by your particular programming language. Each of the seven components of the data record is an integer value. The record is read only, passed by reference.
4. Pass to GKS$ESCAPE or GKS$GDP only the data record components required by the escape or GDP. For instance, if an escape or GDP only requires 5 data record components, omit values from components 6 and 7.

5. Pass to GKS$ESCAPE or GKS$GDP the exact size of the valid portion of the data record, as specified in Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual*. For instance, if an escape or GDP requires 5 valid components to the data record, then pass the value 20 as the data record size (each component being a longword in length).

The DEC GKS standard escape/GDP data record is as follows:

Position	Data Type	Description
1	Integer	Number of integer values passed in the data record.
2	Integer	Number of real values passed in the data record.
3	Integer	Number of string addresses passed in the data record.
4	Integer (address)	Address of array of integers with exactly as many elements as the number specified in component number 1.
5	Integer (address)	Address of array of real numbers with exactly as many elements as the number specified in component number 2.
6	Integer (address)	Address of array of string lengths with exactly as many elements as the number specified in component number 3.
7	Integer (address)	Address of array of string addresses with exactly as many elements as the number specified in component number 3.

After performing its task, some escape functions pass information back to you by use of an output data record. This output data record is identical in format to the input data record, except that the output record's components are modifiable. You pass the buffer sizes in the first three components and the addresses of your buffers in the last four components. DEC GKS modifies the first three components to contain the number of elements DEC GKS actually used to write output data to each of the corresponding buffers.

If you are using an escape function and you need to determine the size required by the entire output data record buffer, you can pass the value 0 to the output record buffer size (documented as the argument record_buffer_length in the GKS$ESCAPE function description, described in Chapter 4, Control Functions, in the *DEC GKS Reference Manual*). When you pass the value 0 as this argument, GKS$ESCAPE does *not* perform the escape, but instead returns the size of the output data record to argument record_size. In this manner, you can be sure that you declared an output data record buffer that is large enough to hold the entire data record.

To place array addresses in the fourth, fifth, sixth, and seventh components of the data record, you need to use a technique specific to your programming language. For instance, using VAX FORTRAN, you can use the %LOC built-in function. For more information concerning addresses and pointers, refer to the documentation set for your programming language. For more information concerning the use of %LOC and data records, refer to the choice input examples in Chapter 8, Input Functions, in the *DEC GKS Reference Manual*.

Each of the device-specific appendixes in this manual describes its level of support for the various escapes and GDPs provided by DEC GKS. For more information, refer to Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual* or to the appropriate device-specific appendix.

NOTE

Remember that the DEC GKS input data records have a format that is completely different from the DEC GKS standard escape/GDP data record format. To review the GKS standard input data records, refer to Chapter 8, Input Functions, in the *DEC GKS Reference Manual*. To review the actual data records required by the DEC GKS graphics handlers, refer to Appendix J, DEC GKS Specific Input Values, in the *DEC GKS Reference Manual*.

Chapter 2

Compiling, Linking, and Running DEC GKS Programs on VMS

The DEC GKS functions that begin with the prefix GKS$ are designed to be used on one of the VMS systems. Those functions meet the *functional* GKS standard. In other words, they perform the necessary tasks as designated by the GKS standard.

However, these functions are in no way meant to meet a *syntactical* standard. For instance, the DEC GKS function GKS$CELL_ARRAY might have a different number of arguments than the cell array function in another GKS implementation. As a result, programs written using the GKS$ interface are not easily transportable; you have to edit the function names, and quite possibly the number and order of function arguments.

To provide a syntactical standard so that programs can be transportable between GKS implementations, the GKS standard defines the FORTRAN binding for users of the FORTRAN language, and the C binding for users of the C language. These bindings standardize the GKS function names, and the number and order of function parameters. Consequently, you can use a program written using the DEC GKS FORTRAN or C bindings on any GKS implementation.

Note that all of the names begin with an uppercase G. If you are writing a GKS program that uses the FORTRAN binding, avoid using an uppercase G as the first letter of the name of any of your own subroutines, functions, variables, or the name of your main program.

2.1 Compiling, Linking, and Running

To use the GKS$ binding, you compile and execute programs using the GKS$ binding functions just as you would any other program. Use the compile command that is appropriate for your chosen programming language, and use the RUN command to execute the image.

The symbols in the DEC GKS image have been inserted in the system image library. Therefore, to link your programs that use the GKS$ binding, you only need to specify the name of your program's object file on the command line, as follows:

```
$ LINK  MYPROG.OBJ RETURN
```

If you choose to make reference to the GKS$ binding constant values, you need to include the appropriate definitions file into your source program. There is a separate definitions file for the GKS$ constants. For specific information concerning the inclusion of definition files, refer to Chapter 1, Introduction to DEC GKS. To review the list of GKS$ binding constants, refer to Appendix B, DEC GKS Constants, in the *DEC GKS Reference Manual*.

Chapter 3

Compiling, Linking, and Running DEC GKS Programs on ULTRIX

The DEC GKS functions that begin with the prefix GKS$ are designed to be used on a DIGITAL system. Those functions meet the *functional* GKS standard. In other words, they perform the necessary tasks as designated by the GKS standard.

However, these functions are in no way meant to meet a *syntactical* standard. For instance, the DEC GKS function GKS$CELL_ARRAY might have a different number of arguments than the cell array function in another GKS implementation. As a result, programs written using the GKS$ interface are not easily transportable; you have to edit the function names, and quite possibly the number and order of function arguments.

Use the FORTRAN binding, and approved ISO and ANSI standard, for tranportability.

3.1 ULTRIX Programming Considerations

The specific method for using DEC GKS software depends on the features and conventions of each VAX language. This section discusses general issues that must be considered when using any VAX language with DEC GKS. For a discussion of the capabilities of each of the DEC GKS supported physical devices, refer to the appropriate device-specific chapter in the *DEC GKS Device Specifics Reference Manual.*

3.1.1 Supported Languages

DEC GKS supports the following languages:

- VAX FORTRAN
- VAX C
- CC (Portable C)

3.1.2 Capabilities of Supported Devices

In many applications, you may wish to write completely device-independent programs. In this way, you can run your programs using different devices without having to rewrite your programs. The *DEC GKS User Manual* outlines the procedure for device-independent programming using DEC GKS.

However, you may wish to review the range of capabilities of the DEC GKS supported devices, or you may wish to write device-dependent subroutines within your application. In any instance, it is helpful to review the device-specific appendixes in this manual before you begin coding your application. The device-dependent appendixes contain information concerning predefined bundle index representations, color capabilities, initial input values, bit masks as workstation type values, supported escape functions for that particular device, and similar information.

3.1.3 Calling Sequences

Each DEC GKS function requires a specific calling sequence. The calling sequence indicates the elements included in the language statement that calls the function, and the order of those elements. The three elements are the following:

- **Call Type**
  High-level VAX languages call DEC GKS functions with CALL statements or function references. For example, when using FORTRAN, you can use a CALL statement to call DEC GKS functions.
- **Function Identifier**
  All DEC GKS function names begin with the prefix GKS$. FORTRAN binding names begin with the uppercase letter G, and C binding names begin with a lowercase g. The remainder of the name indicates the operation performed by the function.

If writing programs to be transported across systems or across GKS implementations, you should use the appropriate language binding functions. Refer to *DEC GKS FORTRAN Binding Reference Manual* and *DEC GKS C Binding Reference Manual* for information concerning the FORTRAN and C binding function names.

- **Argument List**

  Arguments that are passed to DEC GKS functions must be listed in the order shown in the syntax descriptions contained in this manual. See Section 3.1.4.1 for more information concerning the function description format used in this manual. The various language binding functions may have an argument list that is different from the corresponding GKS$ function.

The specific requirements for writing calls and passing arguments to DEC GKS functions vary from one language to another. Whatever the language of the calling program, DEC GKS$ binding functions expect the following:

- Integer arguments to be 32-bit longwords passed by reference.
- Real numbers to be in single-precision, floating-point format passed by reference.
- Character strings to be passed by string descriptors.
- Arrays to be passed either by reference or by descriptor, depending on the particular DEC GKS function.

Each language may have specific requirements concerning the language-specific calling sequence. In VAX C, for example, strings are passed by a null terminator. For a discussion of language-specific programming concerns, refer to Appendix F, Language-Specific Programming Information, in the *DEC GKS Reference Manual*.

NOTE

All languages that need to declare DEC GKS functions as external functions should type the appropriate language definition file, to determine the actual function parameter identifiers specified in the DEC GKS code.

3.1.4 Constants and Include Files

DEC GKS constants are symbolic names that are syntactically equivalent to literal integer constants. These constants are used in the following ways:

- As arguments to DEC GKS functions.
- As literal values to which you can compare a returned value from an inquiry function (for example, you can compare the return value, from a call to the function GKS$INQ_WS_TYPE, to the constant GKS$K_VT125).
- As literal completion status codes to which you can compare a function return value.

NOTE

Constants (defines) for the bindings are in the binding specific include files.

Many DEC GKS functions use constants as arguments, as shown by the following function call:

```
GKS$CLEAR_WS( 1, GKS$K_CLEAR_ALWAYS)
```

You can compare one of the completion status codes to a function return value, as follows:

```
   .
   .
   .
IF ( GKS$_SUCCESS = GKS$ACTIVATE_WS( 1 ) )
   .
   .
   .
```

Most DEC GKS constants begin with the prefix GKS$K_ and are defined in a definition file. All DEC GKS completion status code constants begin with the prefix GKS$_ERROR_ or DECGKS$_ERROR_NEG_ and are defined in a separate definition file. All DEC GKS bit mask constants begin with the prefix GKS$M_.

You can either specify a literal value as an argument to a DEC GKS function, or you can include the language definition files and use a defined constant name instead. The use of constants adds to program legibility and program documentation.

To review the list of DEC GKS constants, refer to Appendix B, DEC GKS Constants, in the *DEC GKS Reference Manual*. To review the list of DEC GKS completion status code constants, refer to Appendix D, DEC GKS Error Messages, in the *DEC GKS Reference Manual*.

3.1.4.1 Including Definition Files

You use DEC GKS software primarily by placing calls to DEC GKS functions in your program. However, when using DEC GKS, you need statements in your program other than calls to GKS functions. The specific statements that are needed depend on the VAX language you use. (For more information, refer to Appendix F, Language-Specific Programming Information, in the *DEC GKS Reference Manual.*)

DEC GKS constants and their values must be made available to all programs using DEC GKS regardless of the language you use. All high-level languages that use DEC GKS have a method for inserting an external file into the program source code stream at compile time. Incorporating an external file is the method for making DEC GKS constants available.

Your installation kit has been supplied with files that contain DEC GKS constants and separate files that contain DEC GKS completion status code constants. You incorporate these files into your program with a statement that is appropriate to the language you are using.

For example, the C programming language provides the #INCLUDE statement for inserting an external file into a program. Therefore, any C program that uses the C binding should contain the following statement:

```
#INCLUDE <GKS/gks.h>
```

Any FORTRAN program that uses the FORTRAN binding functions should contain the following statement:

```
INCLUDE '/usr/include/GKS/gksdefs.bnd'
```

The language definition files located in /usr/include/GKS are as follows:

- gksdefs.h for VAX C and CC (GKS$ binding)
- gks.h for VAX C and CC (C binding)
- gksdefs.bnd for VAX FORTRAN using the FORTRAN binding functions

The completion status code definition files located in usr/include/GKS are as follows:

- gksmsgs.h for VAX C

Each file includes comments that describe the exact method for using a given definition file.

3.1.5 Compiling and Linking GKS$ Programs

A program that uses DEC GKS function calls should be compiled and executed as any other program. Use the compile command that is appropriate to the language you are using. To run an executable program, type the executable file name that you specified.

To compile and link a DEC GKS GKS$ program, use the following syntax:

```
vcc -o application application.c\ RETURN
   -lGKS -ddif -dwt -lcursesX -lc -lX11 -lm -lc
```

NOTE

The \ RETURN convention indicates that you type the backslash character \, press Return, and then type text on the next line of the screen.

3.1.6 Environment Variables and DEC GKS Programming

In many DEC GKS programs, the execution of your application appears as follows:

```
❶       CALL GKS$OPEN_GKS( stderr )

❷       CALL GKS$OPEN_WS( 1, GKS$K_CONID_DEFAULT,
       * GKS$K_WSTYPE_DEFAULT )

        CALL GKS$ACTIVATE_WS( 1 )
           .
           .
           .
C       Release the DEC GKS and workstation environments.
        CALL GKS$DEACTIVATE_WS( 1 )
        CALL GKS$CLOSE_WS( 1 )
        CALL GKS$CLOSE_GKS()
```

The following numbers correspond to the numbers in the previous example:

❶ In this call to GKS$OPEN_GKS, the name stderr is the only argument to the function. This argument tells DEC GKS where to write generated error messages.

If you pass the name stderr (or the value 0), DEC GKS writes the error messages to the specified location. By default, stderr goes to the device /dev/tty, which translates to your process' default device connection (error messages appear on your terminal's display surface).

If you choose, you can specify a path name as an argument to GKS$OPEN_GKS. In this way, you have a permanent record of generated error messages for use during program debugging.

❷ The constant GKS$K_CONID_DEFAULT (or the value 0) tells DEC GKS to evaluate the environment variable GKSconid in order to determine the name of the device connection. evaluate the environment variable GKSwstype in order to determine the name of the workstation type.

Consequently, you can use the setenv command to your shell to define the environment variables to be the connection and type with which you are working, as follows:

```
csh> setenv  GKSconid  /dev/tty RETURN
csh> setenv  GKSwstype  13   # VT241 Color RETURN
csh> application RETURN
   .
   .
   .
csh> setenv  GKSconid  /dev/tt00 RETURN
csh> setenv  GKSwstype  12   # VT125 Black and White RETURN
csh> application RETURN
   .
   .
   .
```

There may be times when you do not wish to define the DEC GKS environment variables. In this case, or if you define an invalid value, DEC GKS translates several environment variables in the following order:

1. If the environment variable GKSconid is undefined, DEC GKS uses /dev/tty for output.
2. If the environment variable GKSwstype is undefined, then DEC GKS sets the device type to be GKS$K_VT240BW (the value 14, a black and white VT240).

The ability to define GKSconid and GKSwstype provides device independency. For more information concerning device-independent DEC GKS programs, refer to the *DEC GKS User Manual.*

3.1.6.1 Specifying Bit Masks as Workstation Type Values

You have the option of specifying the workstation type value in either a hexadecimal, octal, or decimal longword value. In most cases, it is sufficient to specify the type value in decimal.

However, some of the DEC GKS supported devices allow you to pass a *bit mask* in the first word of the longword workstation type value. For example, the following workstation type specifies default values for the DIGITAL LVP16 plotter:

```
csh> setenv  GKSwstype   51 RETURN
```

The following decimal workstation type specifies to DEC GKS to use the LVP16 plotter in landscape mode, with a paper size of 11 x 17:

```
csh> setenv  GKSwstype   %x131123 RETURN
```

For a complete list of all of the available bit masks for a given device, refer to the *DEC GKS Device Specifics Reference Manual.*

Chapter 4

Control Functions

The control functions establish the DEC GKS and workstation environments, and control the workstation surface in a variety of ways. The following list presents the control functions by category:

Category	GKS Functions
GKS Environment	GKS$OPEN_GKS, GKS$CLOSE_GKS
Workstation Environment	GKS$OPEN_WS, GKS$ACTIVATE_WS, GKS$DEACTIVATE_WS, GKS$CLOSE_WS
Display Surface Control	GKS$CLEAR_WS, GKS$REDRAW_SEG_ON_WS, GKSSET_DEFER_STATE, GKSUPDATE_WS
Additional Control	GKS$ESCAPE, GKS$MESSAGE

ACTIVATE WORKSTATION

Operating States: WSOP, WSAC

Syntax

GKS$ACTIVATE_WS *(workstation_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier

CLEAR WORKSTATION

Operating States: WSOP, WSAC

Syntax

GKS$CLEAR_WS *(workstation_id, flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
flag	integer	I	Reference	Control flag (GKS$K_CLEAR_CONDITIONALLY, GKS$K_CLEAR_ALWAYS)

CLOSE GKS

Operating States: GKOP

Syntax

GKS$CLOSE_GKS *()*

CLOSE WORKSTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$CLOSE_WS *(workstation_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier

DEACTIVATE WORKSTATION

Operating States: WSAC

Syntax

GKS$DEACTIVATE_WS *(workstation_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier

ESCAPE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$ESCAPE *(function_id, in_data_record, in_record_size, out_data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
function_id	integer	I	Reference	Function identifier.
in_data_record	record	I	Reference	Pointer to input data record buffer.
in_record_size	integer	I	Reference	Size of data record in bytes.
out_data_record	record	M	Reference	Pointer to output data record buffer.
record_buffer_length	integer	M	Reference	On input, size of output record in bytes. On output, the amount of the buffer actually containing the output data record.
record_size	integer	O	Reference	Total size of the output data record in bytes.

NOTE

If you pass the value 0 as the argument record_buffer_length, the function GKS$ESCAPE checks for errors, returns the size of the output data record in record_size, but does *not* perform the escape. In this way, you can check the actual size of the data record to compare it to your buffer space.

MESSAGE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$MESSAGE *(workstation_id, message)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
message	string	I	Descriptor	Text of the message

OPEN GKS

Operating States: GKCL

Syntax

GKS$OPEN_GKS *(error_file [, memory])*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_file	string	I	Descriptor	Either logical name or physical name of a device or a file that points to the error log file.
memory	integer	I	Reference	Amount of memory units available to DEC GKS (optional argument). Ignored by DEC GKS.

OPEN WORKSTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$OPEN_WS *(workstation_id, device_connection_id, workstation_type)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_connection_id	string	I	Descriptor	Connection identifier
workstation_type	integer	I	Reference	Workstation type

REDRAW ALL SEGMENTS ON WORKSTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REDRAW_SEG_ON_WS *(workstation_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier

SET DEFERRAL STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_DEFER_STATE *(workstation_id, deferral_mode, regeneration_mode)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
deferral_mode	integer	I	Reference	Deferral mode (GKSK_ASAP, GKSK_BNIG, GKSK_BNIL, GKSK_ASTI)
regeneration_mode	integer	I	Reference	Implicit regeneration mode (GKS$K_IRG_SUPPRESSED, GKS$K_IRG_ALLOWED)

UPDATE WORKSTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$UPDATE_WS *(workstation_id, flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
flag	integer	I	Reference	Implicit regeneration flag (GKS$K_POSTPONE_FLAG, GKS$K_PERFORM_FLAG)

Chapter 5

Output Functions

The DEC GKS output functions generate the basic components, or **primitives**, of all graphical pictures. The output functions are divided into the following categories:

Category	GKS Functions
Draw connected lines.	GKS$POLYLINE
Mark locations with symbols.	GKS$POLYMARKER
Draw text.	GKS$TEXT
Fill a polygon.	GKS$FILL_AREA
Color cells of a rectangle.	GKS$CELL_ARRAY
Draw generalized drawing primitive.	GKS$GDP

CELL ARRAY

Operating States: WSAC, SGOP

Syntax

GKS$CELL_ARRAY *(starting_point_x, starting_point_y, diagonal_point_x, diagonal_point_y, offset_column_number, offset_row_number, number_of_columns, number_of_rows, color_index_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
starting_point_x	real	I	Reference	Starting point X value in world coordinates
starting_point_y	real	I	Reference	Starting point Y value in world coordinates
diagonal_point_x	real	I	Reference	Diagonal point X value in world coordinates
diagonal_point_y	real	I	Reference	Diagonal point Y value in world coordinates
offset_column_number	integer	I	Reference	Column number offset into the color index array
offset_row_number	integer	I	Reference	Row number offset into the color index array
number_of_columns	integer	I	Reference	Number of columns in cell array rectangle
number_of_rows	integer	I	Reference	Number of rows in cell array rectangle
color_index_value	2-D array (integer)	I	Descriptor	Two-dimensional array containing color index values

FILL AREA

Operating States: WSAC, SGOP

Syntax

GKS$FILL_AREA *(number_of_points, x_coordinates, y_coordinates)*

Arguments

Argument	Data Type	I/O	Passed by	Description
number_of_points	integer	I	Reference	Number of points in the polygon
x_coordinates	array (real)	I	Reference	X world coordinate points
y_coordinates	array (real)	I	Reference	Y world coordinate points

GENERALIZED DRAWING PRIMITIVE

Operating States: WSAC, SGOP

Syntax

GKS$GDP *(number_of_points, x_coordinates, y_coordinates, gdp_id, data_record, data_record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
number_of_points	integer	I	Reference	Number of points in the GDP
x_coordinates	array (real)	I	Reference	X world coordinate points
y_coordinates	array (real)	I	Reference	Y world coordinate points
gdp_id	integer	I	Reference	GDP identifier
data_record	record	I	Reference	Address of the GDP data record
data_record_size	integer	I	Reference	GDP data record size in bytes

POLYLINE

Operating States: WSAC, SGOP

Syntax

GKS$POLYLINE *(number_of_points, x_coordinates, y_coordinates)*

Arguments

Argument	Data Type	I/O	Passed by	Description
number_of_points	integer	I	Reference	Number of points
x_coordinates	array (real)	I	Reference	X world coordinate points
y_coordinates	array (real)	I	Reference	Y world coordinate points

POLYMARKER

Operating States: WSAC, SGOP

Syntax

GKS$POLYMARKER *(number_of_points, x_coordinates, y_coordinates)*

Arguments

Argument	Data Type	I/O	Passed by	Description
number_of_points	integer	I	Reference	Number of points
x_coordinates	array (real)	I	Reference	X world coordinate points
y_coordinates	array (real)	I	Reference	Y world coordinate points

TEXT

Operating States: WSAC, SGOP

Syntax

GKS$TEXT *(x_coordinate, y_coordinate, text_string)*

Arguments

Argument	Data Type	I/O	Passed by	Description
x_coordinate	real	I	Reference	X starting position in world coordinates
y_coordinate	real	I	Reference	Y starting position in world coordinates
text_string	string	I	Descriptor	Text string

Chapter 6

Output Attribute Functions

The DEC GKS output attribute functions affect the appearance of generated output primitives. The following list presents the output attribute functions by category:

Category	GKS Functions
Fill Area Attributes	GKS$SET_FILL_COLOR_INDEX, GKSSET_FILL_INDEX, GKSSET_FILL_INT_STYLE, GKS$SET_FILL_STYLE_INDEX, GKS$SET_PAT_REF_PT, GKS$SET_PAT_SIZE
Polyline Attributes	GKS$SET_PLINE_COLOR_INDEX, GKSSET_PLINE_INDEX, GKSSET_PLINE_LINETYPE, GKS$SET_PLINE_LINEWIDTH
Polymarker Attributes	GKS$SET_PMARK_COLOR_INDEX, GKSSET_PMARK_INDEX, GKSSET_PMARK_SIZE, GKS$SET_PMARK_TYPE
Text Attributes	GKSSET_TEXT_ALIGN, GKSSET_TEXT_COLOR_INDEX, GKSSET_TEXT_EXPFAC, GKSSET_TEXT_FONTPREC, GKSSET_TEXT_HEIGHT, GKSSET_TEXT_INDEX, GKSSET_TEXT_PATH, GKSSET_TEXT_SPACING, GKS$SET_TEXT_UPVEC
Aspect Source Flags	GKS$SET_ASF
Representations	GKSSET_COLOR_REP, GKSSET_FILL_REP, GKSSET_PAT_REP, GKSSET_PLINE_REP, GKSSET_PMARK_REP, GKSSET_TEXT_REP

SET ASPECT SOURCE FLAGS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_ASF *(flags)*

Arguments

Argument	Data Type	I/O	Passed by	Description
flags	array (integer)	I	Reference	Array of length 13 containing the aspect source flags (GKS$K_ASF_BUNDLED, GKS$K_ASF_INDIVIDUAL), as follows: 1. Linetype 2. Linewidth scale factor 3. Polyline color index 4. Marker type 5. Marker size scale factor 6. Polymarker color index 7. Text font and precision 8. Character expansion factor 9. Character spacing 10. Text color index 11. Fill area interior style 12. Fill area style index 13. Fill area color index

SET CHARACTER EXPANSION FACTOR

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_EXPFAC *(expansion_factor)*

Arguments

Argument	Data Type	I/O	Passed by	Description
expansion_factor	real	I	Reference	Character expansion factor

SET COLOR REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_COLOR_REP *(workstation_id, color_index, red_intensity, green_intensity, blue_intensity)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
color_index	integer	I	Reference	Color index
red_intensity	real	I	Reference	Red intensity
green_intensity	real	I	Reference	Green intensity
blue_intensity	real	I	Reference	Blue intensity

SET FILL AREA COLOR INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_FILL_COLOR_INDEX *(color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
color_index	integer	I	Reference	Color index

SET FILL AREA INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_FILL_INDEX *(index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
index	integer	I	Reference	Fill area bundle index

SET FILL AREA INTERIOR STYLE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_FILL_INT_STYLE *(set_style)*

Arguments

Argument	Data Type	I/O	Passed by	Description
set_style	integer	I	Reference	Fill area interior style (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)

SET FILL AREA REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_FILL_REP *(workstation_id, fill_index, interior_style, style_index, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
fill_index	integer	I	Reference	Fill area index value
interior_style	integer	I	Reference	Interior style (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)
style_index	integer	I	Reference	Fill area style index
color_index	integer	I	Reference	Fill area color index

SET FILL AREA STYLE INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_FILL_STYLE_INDEX *(style_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
style_index	integer	I	Reference	Fill area style index

SET LINETYPE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PLINE_LINETYPE *(line_type)*

Arguments

Argument	Data Type	I/O	Passed by	Description
line_type	integer	I	Reference	Line type (GKS$K_LINETYPE_SOLID, GKS$K_LINETYPE_DASHED, GKS$K_LINETYPE_DOTTED, GKS$K_LINETYPE_DASHED_DOTTED, GKS$K_LINETYPE_DASH_2_DOT, GKS$K_LINETYPE_DASH_3_DOT, GKS$K_LINETYPE_LONG_DASH, GKS$K_LINETYPE_LONG_SHORT_DASH, GKS$K_LINETYPE_SPACED_DASH, GKS$K_LINETYPE_SPACED_DOT, GKS$K_LINETYPE_DOUBLE_DOT, GKS$K_LINETYPE_TRIPLE_DOT)

SET LINEWIDTH SCALE FACTOR

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PLINE_LINEWIDTH *(line_width_scale_factor)*

Arguments

Argument	Data Type	I/O	Passed by	Description
line_width_scale_factor	real	I	Reference	Linewidth scale factor

SET MARKER SIZE SCALE FACTOR

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PMARK_SIZE *(marker_size_scale_factor)*

Arguments

Argument	Data Type	I/O	Passed by	Description
marker_size_scale_factor	real	I	Reference	Polymarker size scale factor

SET MARKER TYPE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PMARK_TYPE *(marker_type)*

Arguments

Argument	Data Type	I/O	Passed by	Description
marker_ type	integer	I	Reference	Polymarker type (GKS$K_MARKERTYPE_DOT, GKS$K_MARKERTYPE_PLUS, GKS$K_MARKERTYPE_ASTERISK, GKS$K_MARKERTYPE_CIRCLE, GKS$K_MARKERTYPE_DIAGONAL_CROSS, GKS$K_MARKERTYPE_SOLID_CIRCLE, GKS$K_MARKERTYPE_TRIANGLE_UP, GKS$K_MARKERTYPE_SOLID_TRI_UP, GKS$K_MARKERTYPE_TRIANGLE_DOWN, GKS$K_MARKERTYPE_SOLID_TRI_DOWN, GKS$K_MARKERTYPE_SQUARE, GKS$K_MARKERTYPE_SOLID_SQUARE, GKS$K_MARKERTYPE_BOWTIE, GKS$K_MARKERTYPE_SOLID_BOWTIE, GKS$K_MARKERTYPE_HOURGLASS, GKS$K_MARKERTYPE_SOLID_HGLASS, GKS$K_MARKERTYPE_DIAMOND, GKS$K_MARKERTYPE_SOLID_DIAMOND)

SET PATTERN REFERENCE POINT

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PAT_REF_PT *(x_coordinate, y_coordinate)*

Arguments

Argument	Data Type	I/O	Passed by	Description
x_coordinate	real	I	Reference	X coordinate of pattern starting point in world coordinates
y_coordinate	real	I	Reference	Y coordinate of pattern starting point in world coordinates

SET PATTERN REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_PAT_REP *(workstation_id, pattern_index, offset_column_number, offset_row_number, num_columns, num_rows, color_index_array)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
pattern_index	integer	I	Reference	Pattern bundle table index value
offset_column_number	integer	I	Reference	Column offset into color index array
offset_row_number	integer	I	Reference	Row offset into color index array
num_columns	integer	I	Reference	Number of columns to map from the color index array
num_rows	integer	I	Reference	Number of rows to map from the color index array
color_index_array	2-D array (integer)	I	Descriptor	Array containing color index values

SET PATTERN SIZE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PAT_SIZE *(pattern_width, pattern_height)*

Arguments

Argument	Data Type	I/O	Passed by	Description
pattern_width	real	I	Reference	Pattern width in world coordinates units
pattern_height	real	I	Reference	Pattern height in world coordinates units

SET PICK IDENTIFIER

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PICK_ID *(pick_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
pick_id	integer	I	Reference	New pick identifier

SET POLYLINE COLOR INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PLINE_COLOR_INDEX *(color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
color_index	integer	I	Reference	Color index

SET POLYLINE INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PLINE_INDEX *(index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
index	integer	I	Reference	Polyline index

SET POLYLINE REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_PLINE_REP *(workstation_id, polyline_index, line_type, line_width, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
polyline_index	integer	I	Reference	Polyline bundle table index value
line_type	integer	I	Reference	Line type (GKS$K_LINETYPE_SOLID, GKS$K_LINETYPE_DASHED, GKS$K_LINETYPE_DOTTED, GKS$K_LINETYPE_DASHED_DOTTED, GKS$K_LINETYPE_DASH_2_DOT, GKS$K_LINETYPE_DASH_3_DOT, GKS$K_LINETYPE_LONG_DASH, GKS$K_LINETYPE_LONG_SHORT_ DASH, GKS$K_LINETYPE_SPACED_DASH, GKS$K_LINETYPE_SPACED_DOT, GKS$K_LINETYPE_DOUBLE_DOT, GKS$K_LINETYPE_TRIPLE_DOT)
line_width	real	I	Reference	Linewidth scale factor
color_index	integer	I	Reference	Color index

SET POLYMARKER COLOR INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PMARK_COLOR_INDEX *(color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
color_index	integer	I	Reference	Polymarker color index value

SET POLYMARKER INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_PMARK_INDEX *(index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
index	integer	I	Reference	Polymarker bundle index value

SET POLYMARKER REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_PMARK_REP *(workstation_id, polymarker_index, marker_type, marker_size, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
polymarker_index	integer	I	Reference	Polymarker bundle table index

Argument	Data Type	I/O	Passed by	Description
marker_type	integer	I	Reference	Polymarker type (GKS$K_MARKERTYPE_DOT, GKS$K_MARKERTYPE_PLUS, GKS$K_MARKERTYPE_ASTERISK, GKS$K_MARKERTYPE_CIRCLE, GKS$K_MARKERTYPE_DIAGONAL_ CROSS, GKS$K_MARKERTYPE_SOLID_ CIRCLE, GKS$K_MARKERTYPE_TRIANGLE_UP, GKS$K_MARKERTYPE_SOLID_TRI_ UP, GKS$K_MARKERTYPE_TRIANGLE_ DOWN, GKS$K_MARKERTYPE_SOLID_TRI_ DOWN, GKS$K_MARKERTYPE_SQUARE, GKS$K_MARKERTYPE_SOLID_ SQUARE, GKS$K_MARKERTYPE_BOWTIE, GKS$K_MARKERTYPE_SOLID_ BOWTIE, GKS$K_MARKERTYPE_HOURGLASS, GKS$K_MARKERTYPE_SOLID_ HGLASS, GKS$K_MARKERTYPE_DIAMOND, GKS$K_MARKERTYPE_SOLID_ DIAMOND)
marker_size	real	I	Reference	Polymarker size scale factor
color_index	integer	I	Reference	Polymarker color index

SET TEXT ALIGNMENT

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_ALIGN *(horizontal, vertical)*

Arguments

Argument	Data Type	I/O	Passed by	Description
horizontal	integer	I	Reference	Horizontal alignment (GKS$K_TEXT_HALIGN_NORMAL, GKS$K_TEXT_HALIGN_LEFT, GKS$K_TEXT_HALIGN_CENTER, GKS$K_TEXT_HALIGN_RIGHT)
vertical	integer	I	Reference	Vertical alignment (GKS$K_TEXT_VALIGN_NORMAL, GKS$K_TEXT_VALIGN_TOP, GKS$K_TEXT_VALIGN_CAP, GKS$K_TEXT_VALIGN_HALF, GKS$K_TEXT_VALIGN_BASE, GKS$K_TEXT_VALIGN_BOTTOM)

SET TEXT COLOR INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_COLOR_INDEX *(color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
color_index	integer	I	Reference	Color index

SET TEXT FONT AND PRECISION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_FONTPREC *(font_value, precision_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
font_value	integer	I	Reference	Text font
precision_value	integer	I	Reference	Text precision (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHAR, GKS$K_TEXT_PRECISION_STROKE)

SET TEXT HEIGHT

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_HEIGHT *(height)*

Arguments

Argument	Data Type	I/O	Passed by	Description
height	real	I	Reference	Character height in world coordinate units

SET TEXT INDEX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_INDEX *(index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
index	integer	I	Reference	Text bundle index value

SET TEXT PATH

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_PATH *(text_path)*

Arguments

Argument	Data Type	I/O	Passed by	Description
text_path	integer	I	Reference	Text path (GKS$K_TEXT_PATH_RIGHT, GKS$K_TEXT_PATH_LEFT, GKS$K_TEXT_PATH_UP, GKS$K_TEXT_PATH_DOWN)

SET TEXT REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_REP *(workstation_id, text_index, font_value, precision_value, expansion_factor, character_spacing, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
text_index	integer	I	Reference	Text index value
font_value	integer	I	Reference	Text font value
precision_value	integer	I	Reference	Text precision (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHAR, GKS$K_TEXT_PRECISION_STROKE)
expansion_factor	real	I	Reference	Character expansion factor
character_spacing	real	I	Reference	Character spacing factor
color_index	integer	I	Reference	Text color index

SET TEXT SPACING

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_SPACING *(spacing_percentage)*

Arguments

Argument	Data Type	I/O	Passed by	Description
spacing_percentage	real	I	Reference	Character spacing factor

SET TEXT UP VECTOR

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_TEXT_UPVEC *(x_vector, y_vector)*

Arguments

Argument	Data Type	I/O	Passed by	Description
x_vector	real	I	Reference	Horizontal component of slope of up vector
y_vector	real	I	Reference	Vertical component of slope of up vector

Chapter 7

Transformation Functions

The DEC GKS transformation functions allow you to compose a picture, to control how much of the picture is seen on the workstation surface, and to control how much of the workstation surface is used to display the picture. The following list presents the transformation functions by category:

Category	GKS Functions
Normalization	GKS$SELECT_XFORM, GKS$SET_CLIPPING, GKS$SET_VIEWPORT, GKS$SET_VIEWPORT_PRIORITY, GKS$SET_WINDOW
Workstation	GKS$SET_WS_VIEWPORT, GKS$SET_WS_WINDOW

SELECT NORMALIZATION TRANSFORMATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SELECT_XFORM *(transformation_number)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	I	Reference	Normalization transformation number

SET CLIPPING INDICATOR

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_CLIPPING *(clip)*

Arguments

Argument	Data Type	I/O	Passed by	Description
clip	integer	I	Reference	Clipping indicator (GKSK_NOCLIP, GKSK_CLIP)

SET VIEWPORT

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_VIEWPORT *(transformation_number, minimum_x_value, maximum_x_value, minimum_y_value, maximum_y_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	I	Reference	Normalization transformation number
minimum_x_value	real	I	Reference	Minimum X NDC value
maximum_x_value	real	I	Reference	Maximum X NDC value
minimum_y_value	real	I	Reference	Minimum Y NDC value
maximum_y_value	real	I	Reference	Maximum Y NDC value

SET VIEWPORT INPUT PRIORITY

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_VIEWPORT_PRIORITY *(transformation_number, reference_trans_number, relative_priority)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	I	Reference	Normalization transformation number
reference_trans_number	integer	I	Reference	Reference normalization transformation number
relative_priority	integer	I	Reference	Relative priority of normalization transformation number over reference transformation number (GKS$K_INPUT_PRIORITY_HIGHER, GKS$K_INPUT_PRIORITY_LOWER)

SET WINDOW

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$SET_WINDOW *(transformation_number, minimum_x_value, maximum_x_value, minimum_y_value, maximum_y_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	I	Reference	Normalization transformation number
minimum_x_value	real	I	Reference	Minimum X world coordinates value
maximum_x_value	real	I	Reference	Maximum X world coordinates value
minimum_y_value	real	I	Reference	Minimum Y world coordinates value
maximum_y_value	real	I	Reference	Maximum Y world coordinates value

SET WORKSTATION VIEWPORT

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_WS_VIEWPORT *(workstation_id, minimum_x_value, maximum_x_value, minimum_y_value, maximum_y_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
minimum_x_value	real	I	Reference	Minimum X device coordinate value
maximum_x_value	real	I	Reference	Maximum X device coordinate value
minimum_y_value	real	I	Reference	Minimum Y device coordinate value
maximum_y_value	real	I	Reference	Maximum Y device coordinate value

SET WORKSTATION WINDOW

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_WS_WINDOW *(workstation_id, minimum_x_value, maximum_x_value, minimum_y_value, maximum_y_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
minimum_x_value	real	I	Reference	Minimum X NDC value
maximum_x_value	real	I	Reference	Maximum X NDC value
minimum_y_value	real	I	Reference	Minimum Y NDC value
maximum_y_value	real	I	Reference	Maximum Y NDC value

Chapter 8

Input Functions

The DEC GKS input functions allow an application program to accept input from a user. The following list presents the input functions by category:

Category	GKS Functions
Initialization	GKS$INIT_CHOICE, GKS$INIT_LOCATOR, GKS$INIT_PICK, GKS$INIT_STRING, GKS$INIT_STROKE, GKS$INIT_VALUATOR
Mode Control	GKSSET_CHOICE_MODE, GKSSET_LOCATOR_MODE, GKSSET_PICK_MODE, GKSSET_STRING_MODE, GKSSET_STROKE_MODE, GKSSET_VALUATOR_MODE
Request Mode	GKS$REQUEST_CHOICE, GKS$REQUEST_LOCATOR, GKS$REQUEST_PICK, GKS$REQUEST_STRING, GKS$REQUEST_ STROKE, GKS$REQUEST_VALUATOR
Sample Mode	GKS$SAMPLE_CHOICE, GKS$SAMPLE_LOCATOR, GKS$SAMPLE_PICK, GKS$SAMPLE_STRING, GKS$SAMPLE_STROKE, GKS$SAMPLE_VALUATOR
Event Mode	GKS$AWAIT_EVENT, GKS$FLUSH_DEVICE_EVENTS, GKSGET_CHOICE, GKSGET_LOCATOR, GKSGET_PICK, GKSGET_STRING, GKSGET_STROKE, GKSGET_VALUATOR

For complete information concerning the input data record components, refer to Appendix J, DEC GKS Specific Input Values, in the *DEC GKS Reference Manual*.

AWAIT EVENT

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$AWAIT_EVENT *(time_out, workstation_id, input_class, device_number)*

Arguments

Argument	Data Type	I/O	Passed by	Description
time_out	real	I	Reference	Time-out value (in seconds)
workstation_id	integer	O	Reference	Workstation identifier
input_class	integer	O	Reference	Input class (GKS$K_INPUT_CLASS_NONE, GKS$K_INPUT_CLASS_LOCATOR, GKS$K_INPUT_CLASS_STROKE, GKS$K_INPUT_CLASS_VALUATOR, GKS$K_INPUT_CLASS_CHOICE, GKS$K_INPUT_CLASS_PICK, GKS$K_INPUT_CLASS_ STRING)
device_number	integer	O	Reference	Logical input device number

FLUSH DEVICE EVENTS

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$FLUSH_DEVICE_EVENTS *(workstation_id, input_class, device_number)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
input_class	integer	I	Reference	Input class (GKS$K_CLASS_LOCATOR, GKS$K_CLASS_STROKE, GKS$K_CLASS_VALUATOR, GKS$K_CLASS_CHOICE, GKSK_CLASS_PICK, GKSK_CLASS_STRING)
device_number	integer	I	Reference	Logical input device number

GET CHOICE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_CHOICE *(input_status, choice_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_OK, GKSK_STATUS_NOCHOICE)
choice_value	integer	O	Reference	Choice selected

GET LOCATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_LOCATOR *(transformation_number, world_x, world_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	O	Reference	Normalization tranformation used
world_x	real	O	Reference	X world coordinate position of locator
world_y	real	O	Reference	Y world coordinate position of locator

GET PICK

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_PICK *(input_status, segment_name, pick_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_OK, GKSK_STATUS_NOPICK)
segment_name	integer	O	Reference	Picked segment
pick_id	integer	O	Reference	Chosen pick identifier

GET STRING

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_STRING *(string_buffer, string_size, report_string_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
string_buffer	string	O	Descriptor	Application's buffer for input character string
string_size	integer	O	Reference	Number of bytes in returned string
report_string_size	integer	O	Reference	Total size, in bytes, of string found in current event report

GET STROKE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_STROKE *(transformation_number, num_entered_points, stroke_buffer_x, stroke_buffer_y, stroke_size_x, stroke_size_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	O	Reference	Normalization transformation used
num_entered_points	integer	O	Reference	Number of points in current event report
stroke_buffer_x	array (real)	O	Descriptor	X world coordinates of application's stroke buffer
stroke_buffer_y	array (real)	O	Descriptor	Y world coordinates of application's stroke buffer
stroke_size_x	integer	O	Reference	Number of x-coordinates for stroke points accepted from current event report
stroke_size_y	integer	O	Reference	Number of y-coordinates for stroke points accepted from current event report

GET VALUATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_VALUATOR *(real_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
real_value	real	O	Reference	Current measure of the valuator device

INITIALIZE CHOICE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_CHOICE *(workstation_id, device_num, initial_status, initial_choice, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Choice device number
initial_status	integer	I	Reference	Initial input status (GKSK_STATUS_OK, GKSK_STATUS_ NOCHOICE)
initial_choice	integer	I	Reference	Initial choice number
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

INITIALIZE LOCATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_LOCATOR *(workstation_id, device_number, initial_x_value, initial_y_value, transformation_number, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Locator device number
initial_x_value	real	I	Reference	Initial X position in world coordinates
initial_y_value	real	I	Reference	Initial Y position in world coordinates
transformation_number	integer	I	Reference	Translation normalization transformation number
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

INITIALIZE PICK

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_PICK *(workstation_id, device_num, initial_status, initial_segment, initial_pick_id, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Pick device number
initial_status	integer	I	Reference	Initial pick status (GKSK_STATUS_OK, GKSK_STATUS_NOPICK)
initial_segment	integer	I	Reference	Initially picked segment
initial_pick_id	integer	I	Reference	Pick identifier specifying where to initially locate prompt
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

INITIALIZE STRING

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_STRING *(workstation_id, device_number, initial_string, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	String device number
initial_string	string	I	Descriptor	Initially displayed string
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

INITIALIZE STROKE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_STROKE *(workstation_id, device_number, initial_num_points, initial_stroke_x_values, initial_stroke_y_values, transformation_number, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Stroke device number
initial_num_points	integer	I	Reference	Number of points in initial stroke
initial_stroke_x_values	array (real)	I	Reference	X world coordinate values in initial stroke
initial_stroke_y_values	array (real)	I	Reference	Y world coordinate values in initial stroke
transformation_number	integer	I	Reference	Initial normalization transformation number

Argument	Data Type	I/O	Passed by	Description
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

INITIALIZE VALUATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INIT_VALUATOR *(workstation_id, device_number, initial_value, prompt_echo_type, echo_area, data_record, size_of_data)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Valuator device number
initial_value	real	I	Reference	Initial value
prompt_echo_type	integer	I	Reference	Prompt and echo type
echo_area	array (real)	I	Reference	Echo area (four-element array) in device coordinates (xmin, xmax, ymin, ymax)
data_record	record	I	Reference	Pointer to data record
size_of_data	integer	I	Reference	Size of data record buffer, in bytes

REQUEST CHOICE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_CHOICE *(workstation_id, device_number, input_status, choice_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK, GKS$K_STATUS_NOCHOICE)
choice_value	integer	O	Reference	Choice value selected

REQUEST LOCATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_LOCATOR *(workstation_id, device_number, input_status, transformation_number, world_x, world_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK)
transformation_number	integer	O	Reference	Normalization tranformation used
world_x	real	O	Reference	X world coordinate position of locator
world_y	real	O	Reference	Y world coordinate position of locator

REQUEST PICK

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_PICK *(workstation_id, device_number, input_status, segment_name, pick_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK, GKS$K_STATUS_NOPICK)
segment_name	integer	O	Reference	Picked segment
pick_id	integer	O	Reference	Chosen pick identifier

REQUEST STRING

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_STRING *(workstation_id, device_number, input_status, string_buffer, string_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK)
string_buffer	string	O	Descriptor	Application's string buffer
string_size	integer	O	Reference	Size of string in bytes

REQUEST STROKE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_STROKE *(workstation_id, device_number, input_status, transformation_number, num_entered_points, stroke_buffer_x, stroke_buffer_y, stroke_size_x, stroke_size_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK)
transformation_number	integer	O	Reference	Normalization transformation used

Argument	Data Type	I/O	Passed by	Description
num_entered_points	integer	O	Reference	Number of points entered by user
stroke_buffer_x	array (real)	O	Descriptor	X world coordinates of accepted stroke
stroke_buffer_y	array (real)	O	Descriptor	Y world coordinates of accepted stroke
stroke_size_x	integer	O	Reference	Number of stroke points returned
stroke_size_y	integer	O	Reference	Number of stroke points returned

REQUEST VALUATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$REQUEST_VALUATOR *(workstation_id, device_number, input_status, real_value)*

Arguments

Data Type	Argument	I/O	Description	
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_NONE, GKSK_STATUS_OK)
real_value	real	O	Reference	Real number chosen by user

SAMPLE CHOICE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_CHOICE *(workstation_id, device_number, input_status, choice_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_OK, GKSK_STATUS_NOCHOICE)
choice_value	integer	O	Reference	Choice value selected

SAMPLE LOCATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_LOCATOR *(workstation_id, device_number, transformation_number, world_x, world_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
transformation_number	integer	O	Reference	Normalization tranformation used
world_x	real	O	Reference	X world coordinate position of locator
world_y	real	O	Reference	Y world coordinate position of locator

SAMPLE PICK

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_PICK *(workstation_id, device_number, input_status, segment_name, pick_id)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
input_status	integer	O	Reference	Input status flag (GKSK_STATUS_OK, GKSK_INPUT_NOPICK)
segment_name	integer	O	Reference	Picked segment
pick_id	integer	O	Reference	Pick identifier associated with the chosen picked primitive

SAMPLE STRING

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_STRING *(workstation_id, device_number, string_buffer, string_size, total_string_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
string_buffer	string	O	Descriptor	Application's input character string buffer
string_size	integer	O	Reference	Number of bytes returned in string_buffer and removed from device's buffer
total_string_size	integer	O	Reference	Total number of characters in the device's buffer, in bytes

SAMPLE STROKE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_STROKE *(workstation_id, device_number, transformation_number, num_entered_points, stroke_buffer_x, stroke_buffer_y, stroke_size_x, stroke_size_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
transformation_number	integer	O	Reference	Normalization transformation used
num_entered_points	integer	O	Reference	Number of points in stroke entered by user

Argument	Data Type	I/O	Passed by	Description
stroke_buffer_x	array (real)	O	Descriptor	Array containing X world coordinate stroke values
stroke_buffer_y	array (real)	O	Descriptor	Array containing Y world coordinate stroke values
stroke_size_x	integer	O	Reference	Number of x-coordinates for stroke points returned in buffer stroke_buffer_x and removed from device's buffer
stroke_size_y	integer	O	Reference	Number of y-coordinates for stroke points returned in buffer stroke_buffer_y and removed from device's buffer

SAMPLE VALUATOR

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SAMPLE_VALUATOR *(workstation_id, device_number, real_value)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
real_value	real	O	Reference	Current measure of the valuator device

SET CHOICE MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_CHOICE_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

SET LOCATOR MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_LOCATOR_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

SET PICK MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_PICK_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

SET STRING MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_STRING_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

SET STROKE MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_STROKE_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

SET VALUATOR MODE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_VALUATOR_MODE *(workstation_id, device_number, operating_mode, echo_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
operating_mode	integer	I	Reference	Input operating mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	I	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

Chapter 9

Segment Functions

The DEC GKS segment functions create, manipulate, and delete stored groups of output primitives called *segments*. The segment functions can be divided into the following categories:

Category	GKS Functions
Using Segments	GKS$ASSOC_SEG_WITH_WS, GKS$CLOSE_SEG, GKS$COPY_SEG_TO_WS, GKS$CREATE_SEG, GKS$DELETE_SEG, GKS$DELETE_SEG_FROM_WS, GKS$INSERT_SEG, GKS$RENAME_SEG
Primitive Attributes	GKS$SET_PICK_ID
Segment Attributes	GKS$SET_SEG_DETECTABILITY, GKS$SET_SEG_HIGHLIGHTING, GKS$SET_SEG_PRIORITY, GKS$SET_SEG_VISIBILITY
Segment Transformations	GKS$ACCUM_XFORM_MATRIX, GKS$EVAL_XFORM_MATRIX, GKS$SET_SEG_XFORM

ASSOCIATE SEGMENT WITH WORKSTATION

Operating States: WSOP, WSAC

Syntax

GKS$ASSOC_SEG_WITH_WS *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
segment_name	integer	I	Reference	Segment name

CLOSE SEGMENT

Operating States: SGOP

Syntax

GKS$CLOSE_SEG *()*

COPY SEGMENT TO WORKSTATION

Operating States: WSOP, WSAC

Syntax

GKS$COPY_SEG_TO_WS *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
segment_name	integer	I	Reference	Segment name

CREATE SEGMENT

Operating States: WSAC

Syntax

GKS$CREATE_SEG *(segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name

DELETE SEGMENT

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$DELETE_SEG *(segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name

DELETE SEGMENT FROM WORKSTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$DELETE_SEG_FROM_WS *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
segment_name	integer	I	Reference	Segment name

INSERT SEGMENT

Operating States: WSAC, SGOP

Syntax

GKS$INSERT_SEG *(segment_name, insertion_xform_matrix)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
insertion_xform_matrix	array (real)	I	Reference	Six-element insertion transformation matrix

RENAME SEGMENT

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$RENAME_SEG *(old_name, new_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
old_name	integer	I	Reference	Segment's previous name
new_name	integer	I	Reference	Segment's new name

SET DETECTABILITY

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_SEG_DETECTABILITY *(segment_name, detectability_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
detectability_flag	integer	I	Reference	Detectability flag (GKS$K_UNDETECTABLE, GKS$K_DETECTABLE)

SET HIGHLIGHTING

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_SEG_HIGHLIGHTING *(segment_name, highlighting_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
highlighting_flag	integer	I	Reference	Highlighting flag (GKSK_NORMAL, GKSK_HIGHLIGHTED)

SET SEGMENT PRIORITY

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_SEG_PRIORITY *(segment_name, priority)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
priority	real	I	Reference	Segment priority

SET VISIBILITY

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_SEG_VISIBILITY *(segment_name, visibility_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
visibility_flag	integer	I	Reference	Segment visibility (GKS$K_INVISIBLE, GKS$K_VISIBLE)

SET SEGMENT TRANSFORMATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$SET_SEG_XFORM *(segment_name, transformation_matrix)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
transformation_matrix	array (real)	I	Reference	Six-element transformation matrix

Chapter 10

Metafile, Error-Handling, and Utility Functions

The DEC GKS metafile functions provide a mechanism for long-term storage, communication, and reproduction of a graphical image. Metafiles created by an application can be used by other applications on other computer systems to reproduce a picture. When you store picture information in a *metafile*, you store specific information concerning the output primitives contained in the picture, the corresponding output attributes, and other information that may be needed to reproduce the picture. The following list presents the DEC GKS metafile functions:

- GKS$GET_ITEM
- GKS$INTERPRET_ITEM
- GKS$READ_ITEM
- GKS$WRITE_ITEM

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. The following list presents the DEC GKS error-handling functions:

- GKS$EMERGENCY_CLOSE
- GKS$ERROR_HANDLER
- GKS$LOG_ERROR
- GKS$SET_ERROR_HANDLER

The DEC GKS utility functions allow you to evaluate and accumulate information about a transformation matrix. The following list presents the DEC GKS utility functions:

- ACCUMULATE TRANSFORMATION MATRIX
- EVALUATE TRANSFORMATION MATRIX

GET ITEM TYPE FROM GKSM

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$GET_ITEM *(workstation_id, item_type, item_data_length)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
item_type	integer	O	Reference	Item type
item_data_length	integer	O	Reference	Length of the data record, in bytes

INTERPRET ITEM

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INTERPRET_ITEM *(item_type, item_data_length, item_data_record)*

Arguments

Argument	Data Type	I/O	Passed by	Description
item_type	integer	I	Reference	Item type
item_data_length	integer	I	Reference	Total length of the data record, in bytes
item_data_record	record	I	Reference	Item data record

READ ITEM FROM GKSM

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$READ_ITEM *(workstation_id, max_record_length, item_data_record)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
max_record_length	integer	I	Reference	Maximum length of declared variable to hold item's data record, in bytes
item_data_record	record	O	Reference	Item's data record

WRITE ITEM TO GKSM

Operating States: WSAC, SGOP

Syntax

GKS$WRITE_ITEM *(workstation_id, item_type, item_data_length, item_data_record)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
item_type	integer	I	Reference	Item type
item_data_length	integer	I	Reference	Total length of the data record, in bytes
item_data_record	record	I	Reference	Item data record

EMERGENCY CLOSE GKS

Operating States: GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

GKS$EMERGENCY_CLOSE *( )*

ERROR HANDLING

Operating States: GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

GKS$ERROR_HANDLER *(error_number, function_name, error_file)*

Arguments

Argument	Data Type	Passed by	I/O	Description
error_number	integer	I	Reference	Error number
function_name	string	I	Descriptor	Function identifier
error_file	string	I	Descriptor	Name of error logging file

ERROR LOGGING

Operating States: GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

GKS$LOG_ERROR *(error_number, function_name, error_file)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_number	integer	I	Reference	Error number
function_name	string	I	Descriptor	Function identifier
error_file	string	I	Descriptor	Error file

ACCUMULATE TRANSFORMATION MATRIX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$ACCUM_XFORM_MATRIX *(first_xform_matrix, fixed_point_x, fixed_point_y, translation_vec_x, translation_vec_y, rotation, scale_x, scale_y, type_of_coordinates, accumulated_xform_matrix)*

Arguments

Argument	Data Type	I/O	Passed by	Description
first_xform_matrix	array (real)	I	Reference	Six-element segment transformation matrix
fixed_point_x	real	I	Reference	Fixed-point X value
fixed_point_y	real	I	Reference	Fixed-point Y value
translation_vec_x	real	I	Reference	X value of the shift vector
translation_vec_y	real	I	Reference	Y value of the shift vector
rotation	real	I	Reference	Rotation, in radians
scale_x	real	I	Reference	X scale factor

Argument	Data Type	I/O	Passed by	Description
scale_y	real	I	Reference	Y scale factor
type_of_coordinates	integer	I	Reference	Coordinate switch (GKS$K_COORDINATES_WC, GKS$K_COORDINATES_NDC)
accumulated_xform_matrix	array (real)	O	Reference	New six-element transformation matrix; the new component values concatenated to the old matrix

EVALUATE TRANSFORMATION MATRIX

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$EVAL_XFORM_MATRIX *(fixed_point_x, fixed_point_y, translation_vec_x, translation_vec_y, rotation, scale_x, scale_y, type_of_coordinates, transformation_matrix)*

Arguments

Argument	Data Type	I/O	Passed by	Description
fixed_point_x	real	I	Reference	Fixed-point X value
fixed_point_y	real	I	Reference	Fixed-point Y value
translation_vec_x	real	I	Reference	X value of the shift vector
translation_vec_y	real	I	Reference	Y value of the shift vector
rotation	real	I	Reference	Rotation in radians
scale_x	real	I	Reference	X scale factor
scale_y	real	I	Reference	Y scale factor
type_of_coordinates	integer	I	Reference	Coordinate flag (GKS$K_COORDINATES_WC, GKS$K_COORDINATES_NDC)
transformation_matrix	array (real)	O	Reference	Six-element evaluated transformation matrix

Chapter 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.

INQUIRE LEVEL OF GKS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_LEVEL *(error_status, gks_level)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
gks_level	integer	O	Reference	GKS level (GKSK_LEVEL_MA, GKSK_LEVEL_MB, GKSK_LEVEL_MC, GKSK_LEVEL_0A, GKSK_LEVEL_0B, GKSK_LEVEL_0C, GKSK_LEVEL_1A, GKSK_LEVEL_1B, GKSK_LEVEL_1C, GKSK_LEVEL_2A, GKSK_LEVEL_2B, GKSK_LEVEL_2C)

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_MAX_XFORM *(error_status, max_transformation)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
max_transformation	integer	O	Reference	Maximum normalization transformation number supported

INQUIRE WORKSTATION MAXIMUM NUMBERS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_MAX_NUM *(error_status,*
max_open_workstations,
max_active_workstations,
max_ws_with_segment)

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
max_open_workstations	integer	O	Reference	Maximum number of simultaneously open workstations supported
max_active_workstations	integer	O	Reference	Maximum number of simultaneously active workstations supported
max_ws_with_segment	integer	O	Reference	Maximum number of workstations that can be associated with a segment

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_WSTYPE_LIST *(error_status, num_workstation_types, workstation_type_list, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
num_workstation_types	integer	O	Reference	Number of different workstation types
workstation_type_list	array (integer)	O	Descriptor	Integers representing various supported workstations
return_size	integer	O	Reference	Actual number of workstation types passed back to the array

INQUIRE COLOR FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_COLOR_FAC *(workstation_type, error_status, num_colors, color_or_mono, num_color_indexes)*

Arguments

Argument	Data Type	I/O	Description	
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_colors	integer	O	Reference	Number of colors
color_or_mono	integer	O	Reference	Color availability flag (GKS$K_MONOCHROME, GKS$K_COLOR)
num_color_indexes	integer	O	Reference	Number of predefined indexes

INQUIRE DEFAULT CHOICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_CHOICE_DATA *(workstation_type, device_number, error_status, max_choices, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
max_choices	integer	O	Reference	Maximum number of choices
num_prompt_echo_types	integer	O	Reference	Number of choice prompt and echo types

Workstation Description Table Inquiry Functions
INQUIRE DEFAULT CHOICE DATA

Argument	Data Type	I/O	Passed by	Description
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	M	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DEFAULT DEFERRAL STATE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_DEFER_STATE *(workstation_type, error_status, deferral_mode, regeneration_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
deferral_mode	integer	O	Reference	Default deferral mode (GKSK_ASAP, GKSK_BNIG, GKSK_BNIL, GKSK_ASTI)
regeneration_flag	integer	O	Reference	Default regeneration mode (GKS$K_IRG_SUPPRESSED, GKS$K_IRG_ALLOWED)

INQUIRE DEFAULT LOCATOR DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_LOCATOR_DATA *(workstation_type, device_number, error_status, initial_world_x, initial_world_y, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
initial_world_x	real	O	Reference	X-coordinate of starting position of locator prompt in world coordinates
initial_world_y	real	O	Reference	Y-coordinate of starting position of locator prompt in world coordinates
num_prompt_echo_types	integer	O	Reference	Number of locator prompt and echo types

Argument	Data Type	I/O	Passed by	Description
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DEFAULT PICK DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_PICK_DATA *(workstation_type, device_number, error_status, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
num_prompt_echo_types	integer	O	Reference	Number of pick prompt and echo types
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation

Argument	Data Type	I/O	Passed by	Description
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DEFAULT STRING DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_STRING_DATA *(workstation_type, device_number, error_status, buffer_size, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
buffer_size	integer	O	Reference	Maximum string buffer size
num_prompt_echo_types	integer	O	Reference	Number of string prompt and echo types

Argument	Data Type	I/O	Passed by	Description
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DEFAULT STROKE DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_STROKE_DATA *(workstation_type, device_number, error_status, buffer_size, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
buffer_size	integer	O	Reference	Maximum stroke buffer size in bytes
num_prompt_echo_types	integer	O	Reference	Number of stroke prompt and echo types

Argument	Data Type	I/O	Passed by	Description
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DEFAULT VALUATOR DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DEF_VALUATOR_DATA *(workstation_type, device_number, error_status, initial_value, num_prompt_echo_types, prompt_echo_types, echo_area, data_record, num_returned_prompts, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
initial_value	real	O	Reference	Default initial value
num_prompt_echo_types	integer	O	Reference	Number of valuator prompt and echo types

Argument	Data Type	I/O	Passed by	Description
prompt_echo_types	array (integer)	O	Descriptor	Available prompt and echo types on the specified workstation
echo_area	array (real)	O	Reference	Four device coordinates specifying the default echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to data input record
num_returned_prompts	integer	O	Reference	Actual number of prompt and echo types returned
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE DISPLAY SPACE SIZE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_MAX_DS_SIZE *(workstation_type, error_status, meters, device_coordinates_x, device_coordinates_y, raster_units_x, raster_units_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
meters	integer	O	Reference	Flag to determine device coordinate measurement units (GKSK_METERS, GKSK_OTHER_UNITS)
device_coordinates_x	real	O	Reference	Maximum X coordinate value in device coordinates
device_coordinates_y	real	O	Reference	Maximum Y coordinate value in device coordinates
raster_units_x	integer	O	Reference	Maximum X value in raster units
raster_units_y	integer	O	Reference	Maximum Y value in raster units

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DYN_MOD_SEG_ATTB *(workstation_type, error_status, transformation_change, visible_to_invisible, invisible_to_visible, highlight_change, priority_change, add_primitives, segment_deletion)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
transformation_change	integer	O	Reference	Segment transformation (GKSK_IRG, GKSK_IMM)
visible_to_invisible	integer	O	Reference	Visibility turned from ON to OFF (GKSK_IRG, GKSK_IMM)
invisible_to_visible	integer	O	Reference	Visibility turned from OFF to ON (GKSK_IRG, GKSK_IMM)

Argument	Data Type	I/O	Passed by	Description
highlight_change	integer	O	Reference	Highlighting changes (GKSK_IRG, GKSK_IMM)
priority_change	integer	O	Reference	Priority changes (GKSK_IRG, GKSK_IMM)
add_primitives	integer	O	Reference	Adding primitives to an open segment (GKSK_IRG, GKSK_IMM)
segment_deletion	integer	O	Reference	Segment deletion (GKSK_IRG, GKSK_IMM)

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_DYN_MOD_WS_ATTB *(workstation_type, error_status, polyline_representation, polymarker_representation, text_representation, fill_representation, pattern_representation, color_representation, workstation_transformation)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
polyline_representation	integer	O	Reference	Polyline representation changeable (GKSK_IRG, GKSK_IMM)
polymarker_representation	integer	O	Reference	Polymarker representation changeable (GKSK_IRG, GKSK_IMM)

Workstation Description Table Inquiry Functions
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Argument	Data Type	I/O	Passed by	Description
text_representation	integer	O	Reference	Text representation changeable (GKSK_IRG, GKSK_IMM)
fill_representation	integer	O	Reference	Fill area representation changeable (GKSK_IRG, GKSK_IMM)
pattern_representation	integer	O	Reference	Pattern representation changeable (GKSK_IRG, GKSK_IMM)
color_representation	integer	O	Reference	Color representation changeable (GKSK_IRG, GKSK_IMM)
workstation_transformation	integer	O	Reference	Workstation transformation changeable (GKSK_IRG, GKSK_IMM)

INQUIRE FILL AREA FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_FILL_FAC *(workstation_type, error_status, num_interior_styles, interior_style_list, num_hatch_styles, hatch_style_list, num_fill_indexes, hatch_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_interior_styles	integer	O	Reference	Number of available interior styles
interior_style_list	array (integer)	O	Reference	Four elements corresponding to four interior fill styles (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)

Argument	Data Type	I/O	Passed by	Description
num_hatch_styles	integer	O	Reference	Number of available hatch styles
hatch_style_list	array (integer)	O	Descriptor	List of available hatch styles
num_fill_indexes	integer	O	Reference	Number of predefined fill area index values available on specified workstation type
hatch_return_size	integer	O	Reference	Number of hatch styles returned to the list

INQUIRE GENERALIZED DRAWING PRIMITIVE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_GDP *(workstation_type, gdp_id, error_status, num_attribute_sets, attribute_list, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
gdp_id	integer	I	Reference	GDP identifier
error_status	integer	O	Reference	Error status
num_attribute_sets	integer	O	Reference	Number of attribute sets applicable to specified GDP
attribute_list	array (integer)	O	Descriptor	List of attribute sets associated with specified GDP
return_size	integer	O	Reference	Number of attribute sets returned to the attribute list

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_AVAIL_GDP *(workstation_id, error_status, num_gdps, gdp_list, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_gdps	integer	O	Reference	Number of different GDP types
gdp_list	array (integer)	O	Descriptor	Integers representing various supported GDPs for the specified workstation
return_size	integer	O	Reference	Actual number of GDPs passed back to the array

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLE

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_MAX_WS_STATE_TABLE *(workstation_type, error_status, max_pline, max_pmark, max_text, max_fill_area, max_pattern, max_color)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	I	Reference	Error status
max_pline	integer	O	Reference	Maximum polyline bundle entries
max_pmark	integer	O	Reference	Maximum polymarker bundle entries
max_text	integer	O	Reference	Maximum text bundle entries
max_fill_area	integer	O	Reference	Maximum fill area bundle entries
max_pattern	integer	O	Reference	Maximum pattern indexes
max_color	integer	O	Reference	Maximum color indexes

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_INPUT_DEV *(workstation_type, error_status, num_locator_devices, num_stroke_devices, num_valuator_devices, num_choice_devices, num_pick_devices, num_string_devices)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_locator_devices	integer	O	Reference	Number of locator devices
num_stroke_devices	integer	O	Reference	Number of stroke devices
num_valuator_devices	integer	O	Reference	Number of valuator devices
num_choice_devices	integer	O	Reference	Number of choice devices
num_pick_devices	integer	O	Reference	Number of pick devices
num_string_devices	integer	O	Reference	Number of string devices

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_SEG_PRIORITY *(workstation_type, error_status, num_priorities)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_priorities	integer	O	Reference	Number of supported segment priorities

INQUIRE PATTERN FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PAT_FAC *(workstation_type, error_status, num_pattern_indexes)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_pattern_indexes	integer	O	Reference	Number of predefined pattern indexes

INQUIRE POLYLINE FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PLINE_FAC *(workstation_type, error_status, num_line_types, line_types, num_line_widths, nominal_line_width, line_width_min, line_width_max, num_indexes, line_type_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_line_types	integer	O	Reference	Number of line types available
line_types	array (integer)	O	Descriptor	Line types available on specified workstation
num_line_widths	integer	O	Reference	Number of line widths
nominal_line_width	real	O	Reference	Nominal default size line supported by the graphics handler

Workstation Description Table Inquiry Functions
INQUIRE POLYLINE FACILITIES

Argument	Data Type	I/O	Passed by	Description
line_width_min	real	O	Reference	Smallest line width possible
line_width_max	real	O	Reference	Largest line width possible
num_indexes	integer	O	Reference	Number of predefined polyline index values
line_type_return_size	integer	O	Reference	Number of line types returned to the line type list

INQUIRE POLYMARKER FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PMARK_FAC *(workstation_type, error_status, num_marker_types, marker_types, num_marker_sizes, nominal_marker_size, marker_size_min, marker_size_max, marker_indexes, marker_type_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_marker_types	integer	O	Reference	Number of marker types available
marker_types	array (integer)	O	Descriptor	Marker types available on specified workstation
num_marker_sizes	integer	O	Reference	Number of marker sizes

Workstation Description Table Inquiry Functions
INQUIRE POLYMARKER FACILITIES

Argument	Data Type	I/O	Passed by	Description
nominal_marker_size	real	O	Reference	Nominal default size marker supported by the graphics handler
marker_size_min	real	O	Reference	Smallest marker size possible
marker_size_max	real	O	Reference	Largest marker size possible
marker_indexes	integer	O	Reference	Number of predefined polymarker index values
marker_type_return_size	integer	O	Reference	Number of marker types returned to the marker type list

INQUIRE PREDEFINED COLOR REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_COLOR_REP *(workstation_type, color_index, error_status, red_intensity, green_intensity, blue_intensity)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
color_index	integer	I	Reference	Color index
error_status	integer	O	Reference	Error status
red_intensity	real	O	Reference	Red intensity
green_intensity	real	O	Reference	Green intensity
blue_intensity	real	O	Reference	Blue intensity

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_FILL_REP *(workstation_type, fill_index, error_status, interior_style, style_index, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
fill_index	integer	I	Reference	Fill area index
error_status	integer	O	Reference	Error status
interior_style	integer	O	Reference	Fill area interior style (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)
style_index	integer	O	Reference	Fill area style index
color_index	integer	O	Reference	Fill area color index

INQUIRE PREDEFINED PATTERN REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_PAT_REP *(workstation_type, pattern_index, error_status, height, width, color_indexes, color_columns_return_size, color_rows_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
pattern_index	integer	I	Reference	Pattern index
error_status	integer	O	Reference	Error status
height	integer	O	Reference	Columns contained in the color index array used to create the pattern
width	integer	O	Reference	Rows contained in the color index array used to create the pattern
color_indexes	2-D array (integer)	O	Descriptor	Color indexes designating how GKS colors the pattern
color_columns_return_size	integer	O	Reference	Number of columns in array to which GKS returned index values

Argument	Data Type	I/O	Passed by	Description
color_rows_return_size	integer	O	Reference	Number of rows in array to which GKS returned index values

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_PLINE_REP *(workstation_type, polyline_index, error_status, line_type, color_index, line_width_scale_factor)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
polyline_index	integer	I	Reference	Polyline index
error_status	integer	O	Reference	Error status
line_type	integer	O	Reference	Linetype
color_index	integer	O	Reference	Polyline color index
line_width_scale_factor	real	O	Reference	Line width scale factor

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_PMARK_REP *(workstation_type, polymarker_index, error_status, marker_type, color_index, marker_size_scale_factor)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
polymarker_index	integer	I	Reference	Polymarker index
error_status	integer	O	Reference	Error status
marker_type	integer	O	Reference	Marker type
color_index	integer	O	Reference	Polymarker color index
marker_size_scale_factor	real	O	Reference	Marker size scale factor

INQUIRE PREDEFINED TEXT REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PREDEF_TEXT_REP *(workstation_type, text_index, error_status, font_number, precision, character_expansion_factor, character_spacing, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
text_index	integer	I	Reference	Text index
error_status	integer	O	Reference	Error status
font_number	integer	O	Reference	Text font
precision	integer	O	Reference	Text precision (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHAR, GKS$K_TEXT_PRECISION_STROKE)
character_expansion_factor	real	O	Reference	Text expansion factor
character_spacing	real	O	Reference	Text spacing
color_index	integer	O	Reference	Text color index

INQUIRE TEXT FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_TEXT_FAC *(workstation_type, error_status, num_fonts, font_list, precision_list, num_heights, height_min, height_max, num_character_exp, character_exp_min, character_exp_max, num_indexes, precision_return_size, font_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_fonts	integer	O	Reference	Number of fonts available on specified workstation type
font_list	array (integer)	O	Descriptor	Available font numbers
precision_list	array (integer)	O	Descriptor	Available precisions (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHAR, GKS$K_TEXT_PRECISION_STROKE)
num_heights	integer	O	Reference	Number of available character heights

Argument	Data Type	I/O	Passed by	Description
height_min	real	O	Reference	Minimum character height in device coordinates
height_max	real	O	Reference	Maximum character height in device coordinates
num_character_exp	integer	O	Reference	Number of available character expansions
character_exp_min	real	O	Reference	Minimum character expansion
character_exp_max	real	O	Reference	Maximum character expansion
num_indexes	integer	O	Reference	Number of predefined text indexes
precision_return_size	integer	O	Reference	Number of elements returned in precision array
font_return_size	integer	O	Reference	Number of elements returned in font array

INQUIRE WORKSTATION CATEGORY

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_CATEGORY *(workstation_type, error_status, workstation_category)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
workstation_category	integer	O	Reference	Workstation category (GKSK_WSCAT_OUTPUT, GKSK_WSCAT_INPUT, GKSK_WSCAT_OUTIN, GKSK_WSCAT_WISS, GKSK_WSCAT_MO, GKSK_WSCAT_MI)

INQUIRE WORKSTATION CLASSIFICATION

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_CLASSIFICATION *(workstation_type, error_status, classification)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
classification	integer	O	Reference	Workstation classification (GKS$K_WSCLASS_VECTOR, GKS$K_WSCLASS_RASTER, GKS$K_WSCLASS_OTHERD)

INQUIRE CLIPPING

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_CLIP *(error_status, clipping_indicator, clipping_rectangle)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
clipping_indicator	integer	O	Reference	Clipping indicator (GKSK_NOCLIP, GKSK_CLIP)
clipping_rectangle	array (real)	O	Reference	Four elements containing the NDC values of the clipping rectangle

INQUIRE INPUT QUEUE OVERFLOW

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_INPUT_QUEUE_OVERFLOW *(error_status, workstation_id, input_class, device_number)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
workstation_id	integer	O	Reference	Workstation identifier
input_class	integer	O	Reference	Input class
device_number	integer	O	Reference	Device number

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_INDIV_ATTB *(error_status, polyline_type, polyline_width, polyline_color_index, polymarker_type, polymarker_size, polymarker_color_index, text_font, text_precision, character_expansion_factor, character_spacing, text_color_index, interior_style, style_index, fill_color_index, aspect_source_flags)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
polyline_type	integer	O	Reference	Polyline type
polyline_width	real	O	Reference	Polyline width scale factor
polyline_color_index	integer	O	Reference	Polyline color index
polymarker_type	integer	O	Reference	Marker type
polymarker_size	real	O	Reference	Marker size scale factor
polymarker_color_index	integer	O	Reference	Polymarker color index
text_font	integer	O	Reference	Font number

Argument	Data Type	I/O	Passed by	Description
text_precision	integer	O	Reference	Precision (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHAR, GKS$K_TEXT_PRECISION_STROKE)
character_expansion_factor	real	O	Reference	Character expansion factor
character_spacing	real	O	Reference	Character spacing
text_color_index	integer	O	Reference	Text color index
interior_style	integer	O	Reference	Fill area interior style (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)
style_index	integer	O	Reference	Style index
fill_color_index	integer	O	Reference	Fill area color index
aspect_source_flags	array (integer)	O	Reference	Thirteen-element array containing aspect source flag for each of the nongeometric output attributes (GKS$K_ASF_BUNDLED, GKS$K_ASF_INDIVIDUAL)

INQUIRE MORE SIMULTANEOUS EVENTS

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_MORE_SIMUL_EVENTS *(error_status, more_events_flag)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
more_events_flag	integer	O	Reference	More simultaneously generated events flag (GKSK_NOMORE_EVENTS, GKSK_MORE_EVENTS)

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_CURRENT_XFORMNO *(error_status, transformation_number)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
transformation_number	integer	O	Reference	Current normalization transformation number

INQUIRE CURRENT PICK IDENTIFIER

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PICK_ID *(error_status, pick_identifier)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
pick_identifier	integer	O	Reference	Pick identifier

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_PRIM_ATTB *(error_status, list_bundle_indexes, text_height, character_up_vector, character_width, text_base_vector, character_path, character_alignment, pattern_width, pattern_height, pattern_reference_point)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
list_bundle_indexes	array (integer)	O	Reference	Four-element array containing bundle indexes (polyline index, polymarker index, text index, fill area index)
text_height	real	O	Reference	Character height specified by world coordinate value
character_up_vector	array (real)	O	Reference	X and Y coordinates comprising current up vector
character_width	real	O	Reference	Character width in world coordinates
text_base_vector	array (real)	O	Reference	Two-element array containing text base vectors

Argument	Data Type	I/O	Passed by	Description
character_path	integer	O	Reference	Character path (GKS$K_TEXT_PATH_RIGHT, GKS$K_TEXT_PATH_LEFT, GKS$K_TEXT_PATH_UP, GKS$K_TEXT_PATH_DOWN)
character_alignment	integer	O	Reference	Two-element array containing character alignment values (GKS$K_TEXT_HALIGN_NORMAL, GKS$K_TEXT_HALIGN_TOP, GKS$K_TEXT_HALIGN_CAP, GKS$K_TEXT_HALIGN_HALF GKS$K_TEXT_HALIGN_BASE, GKS$K_TEXT_HALIGN_BOTTOM)
pattern_width	array (real)	O	Reference	Two-element array containing pattern width vector
pattern_height	array (real)	O	Reference	Two-element array containing pattern height vector
pattern_reference_point	array (real)	O	Reference	Two-element array containing the x and y world coordinate values designating the pattern reference point

INQUIRE NAME OF OPEN SEGMENT

Operating States: SGOP

Syntax

GKS$INQ_NAME_OPEN_SEG *(error_status, segment_name)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
segment_name	integer	O	Reference	Segment name

INQUIRE OPERATING STATE VALUE

Operating States: GKCL, GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_OPERATING_STATE *(operating_state)*

Arguments

Argument	Data Type	I/O	Passed by	Description
operating_state	integer	O	Reference	GKS operating state (GKSK_GKCL, GKSK_GKOP, GKSK_WSOP, GKSK_WSAC, GKS$K_SGOP)

INQUIRE NORMALIZATION TRANSFORMATION NUMBER

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_XFORM *(transformation_number, error_status, world_window_boundaries, world_viewport_boundaries)*

Arguments

Argument	Data Type	I/O	Passed by	Description
transformation_number	integer	I	Reference	Specified normalization transformation number
error_status	integer	O	Reference	Error status
world_window_ boundaries	array (real)	O	Reference	Four world coordinate values comprising the world window (XMIN, XMAX, YMIN, YMAX)
world_viewport_ boundaries	array (real)	O	Reference	Four NDC values comprising the world viewport (XMIN, XMAX, YMIN, YMAX)

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_XFORM_LIST *(error_status, num_transformations, list_transformations, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
num_transformations	integer	O	Reference	Number of currently defined normalization transformations
list_transformations	array (integer)	O	Descriptor	List of all currently defined normalization transformations, in priority order
return_size	integer	O	Reference	Number of normalization transformations passed to the list

INQUIRE SET OF ACTIVE WORKSTATIONS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_ACTIVE_WS *(error_status, num_workstations, workstation_list, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
num_workstations	integer	O	Reference	Number of active workstations
workstation_list	array (integer)	O	Descriptor	Identifiers associated with currently active workstations
return_size	integer	O	Reference	Number of workstation identifiers returned to list

INQUIRE SET OF OPEN WORKSTATIONS

Operating States: GKOP, WSOP, WSAC, SGOP

Syntax

GKS$INQ_OPEN_WS *(error_status, num_open_workstations, workstation_list, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
num_open_workstations	integer	O	Reference	Number of open workstations
workstation_list	array (integer)	O	Descriptor	Identifiers associated with open workstations
return_size	integer	O	Reference	Number of workstation identifiers returned to the list

INQUIRE SET OF SEGMENT NAMES IN USE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_SEG_NAMES *(error_status, num_segments, list_segments, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
num_segments	integer	O	Reference	Number of currently existing segments
list_segments	array (integer)	O	Descriptor	Segment names corresponding to currently existing segments
return_size	integer	O	Reference	Actual number of segment names returned to the list

INQUIRE CHOICE DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_CHOICE_STATE *(workstation_id, device_number, error_status, operating_mode, echo_flag, initial_choice_status, initial_choice_number, prompt_echo_type, echo_area, data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

Argument	Data Type	I/O	Passed by	Description
initial_choice_status	integer	O	Reference	Determines if user can return a measure value of No Choice. (GKSK_STATUS_OK, GKSK_STATUS_NOCHOICE)
initial_choice_number	integer	O	Reference	Initial choice measure
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Choice data record
record_buffer_length	integer	M	Reference	Size, in bytes, of the data record buffer
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE COLOR REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_COLOR_REP *(workstation_id, color_index, value_type, error_status, red_intensity, green_intensity, blue_intensity)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
color_index	integer	I	Reference	Color index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status
red_intensity	real	O	Reference	Red intensity
green_intenisty	real	O	Reference	Green intensity
blue_intensity	real	O	Reference	Blue intensity

INQUIRE FILL AREA REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_FILL_REP *(workstation_id, fill_area_index, value_type, error_status, interior_style, style_index, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
fill_area_index	integer	I	Reference	Fill area index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status
interior_style	integer	O	Reference	Interior style (GKS$K_INTSTYLE_HOLLOW, GKS$K_INTSTYLE_SOLID, GKS$K_INTSTYLE_PATTERN, GKS$K_INTSTYLE_HATCH)
style_index	integer	O	Reference	Interior style index
color_index	integer	O	Reference	Fill area color index

INQUIRE LIST OF COLOR INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_COLOR_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of color indexes
list_indexes	array (integer)	O	Descriptor	Color index values
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LIST OF FILL AREA INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_FILL_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of fill area indexes
list_indexes	array (integer)	O	Descriptor	Fill area index values
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LIST OF PATTERN INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PAT_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of pattern indexes
list_indexes	array (integer)	O	Descriptor	Pattern index values
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LIST OF POLYLINE INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PLINE_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of polyline indexes
list_indexes	array (integer)	O	Descriptor	Polyline index values
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LIST OF POLYMARKER INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PMARK_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of polymarker indexes
list_indexes	array (integer)	O	Descriptor	Polymarker index values
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LIST OF TEXT INDEXES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_TEXT_INDEXES *(workstation_id, error_status, num_indexes, list_indexes, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
num_indexes	integer	O	Reference	Number of text indexes
list_indexes	array (integer)	O	Descriptor	List of text indexes
return_size	integer	O	Reference	Number of indexes returned to the list

INQUIRE LOCATOR DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_LOCATOR_STATE *(workstation_id, device_number, value_type, error_status, operating_mode, echo_flag, transformation_number, world_location_x, world_location_y, prompt_echo_type, echo_area, data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status

Argument	Data Type	I/O	Passed by	Description
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)
transformation_number	integer	O	Reference	Initial normalization transformation used to translate the locator points
world_locator_x	real	O	Reference	Initial X value of locator position, in world coordinates
world_locator_y	real	O	Reference	Initial Y value of locator position, in world coordinates
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Array of four device coordinates specifying an echo area
data_record	record	O	Reference	Locator data record
record_buffer_length	integer	M	Reference	Size of the data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE PATTERN REPRESENTATION

Operating States: WSOP, WSAC, SGOP

SYNTAX

GKS$INQ_PAT_REP *(workstation_id, pattern_index, value_type, error_status, pattern_width, pattern_height, list_color_indexes, color_columns_return_szie, color_rows_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
pattern_index	integer	I	Reference	Pattern index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status
pattern_width	integer	O	Reference	Number of columns within the color array for use in creating pattern
pattern_height	integer	O	Reference	Number of rows within the color array for use in creating the pattern

Argument	Data Type	I/O	Passed by	Description
list_color_indexes	2-D array (integer)	O	Descriptor	List of color indexes to use in creating the pattern
color_columns_return_size	integer	O	Reference	X dimension of color list array
color_rows_return_size	integer	O	Reference	Y dimension of color list array

INQUIRE PICK DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PICK_STATE *(workstation_id, device_number, value_type, error_status, operating_mode, echo_flag, initial_pick_status, initial_segment, initial_pick_id, prompt_echo_type, echo_area, data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)

Argument	Data Type	I/O	Passed by	Description
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)
initial_pick_status	integer	O	Reference	Initial input status (GKSK_STATUS_OK, GKSK_STATUS_NOPICK)
initial_segment	integer	O	Reference	Initially picked segment
initial_pick_id	integer	O	Reference	Initially chosen pick identifier
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to current pick input data record
record_buffer_length	integer	M	Reference	Size of data record buffer. in bytes
record_size	integer	O	Reference	Total size, in bytes, of the data record

INQUIRE POLYLINE REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PLINE_REP *(workstation_id, polyline_index, value_type, error_status, line_type, line_width_scale_factor, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
polyline_index	integer	I	Reference	Polyline index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status

Argument	Data Type	I/O	Passed by	Description
line_type	integer	O	Reference	Line type associated with specified polyline bundle index (GKS$K_LINETYPE_SOLID, GKS$K_LINETYPE_DASHED, GKS$K_LINETYPE_DOTTED, GKS$K_LINETYPE_DASHED_DOTTED, GKS$K_LINETYPE_DASH_2_DOT, GKS$K_LINETYPE_DASH_3_DOT, GKS$K_LINETYPE_LONG_DASH, GKS$K_LINETYPE_LONG_SHORT_DASH, GKS$K_LINETYPE_SPACED_DASH, GKS$K_LINETYPE_SPACED_DOT, GKS$K_LINETYPE_DOUBLE_DOT, GKS$K_LINETYPE_TRIPLE_DOT)
line_width_scale_factor	integer	O	Reference	Line width scale factor associated with specified polyline bundle index
color_index	integer	O	Reference	Color index associated with specified polyline bundle index

INQUIRE POLYMARKER REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PMARK_REP *(workstation_id, polymarker_index, value_type, error_status, marker_type, marker_size_scale_factor, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_type	integer	I	Reference	Workstation type
polymarker_index	integer	I	Reference	Polymarker index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_REALIZED)
error_status	integer	O	Reference	Error status

Argument	Data Type	I/O	Passed by	Description
marker_type	integer	O	Reference	Polymarker type (GKS$K_MARKERTYPE_DOT, GKS$K_MARKERTYPE_PLUS, GKS$K_MARKERTYPE_ASTERISK, GKS$K_MARKERTYPE_CIRCLE, GKS$K_MARKERTYPE_DIAGONAL_CROSS, GKS$K_MARKERTYPE_SOLID_CIRCLE, GKS$K_MARKERTYPE_TRIANGLE_UP, GKS$K_MARKERTYPE_SOLID_TRI_UP, GKS$K_MARKERTYPE_TRIANGLE_DOWN, GKS$K_MARKERTYPE_SOLID_TRI_DOWN, GKS$K_MARKERTYPE_SQUARE, GKS$K_MARKERTYPE_SOLID_SQUARE, GKS$K_MARKERTYPE_BOWTIE, GKS$K_MARKERTYPE_SOLID_BOWTIE, GKS$K_MARKERTYPE_HOURGLASS, GKS$K_MARKERTYPE_SOLID_HGLASS, GKS$K_MARKERTYPE_DIAMOND, GKS$K_MARKERTYPE_SOLID_DIAMOND)
marker_size_scale_factor	real	O	Reference	Polymarker size scale factor
color_index	integer	O	Reference	Color index associated with specified polymarker index value

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_SEG_NAMES_ON_WS *(workstation_id, error_status, num_segment_names, list_segment_names, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation type
error_status	integer	O	Reference	Error status
num_segment_names	integer	O	Reference	Number of segment names for specified workstation
list_segment_names	array (integer)	O	Descriptor	List of defined segment names
return_size	integer	O	Reference	Number of segment names returned to the list

INQUIRE STRING DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_STRING_STATE *(workstation_id, device_number, error_status, operating_mode, echo_flag, default_string, string_return_size, prompt_echo_type, echo_area, data_record, record_buffer_ length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_ REQUEST, GKS$K_INPUT_MODE_ SAMPLE, GKS$K_INPUT_MODE_ EVENT)
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

Workstation State List Inquiry Functions
INQUIRE STRING DEVICE STATE

Argument	Data Type	I/O	Passed by	Description
default_string	string	O	Descriptor	Default input string
string_return_size	integer	O	Reference	Length of returned string, in bytes
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to string input data record
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE STROKE DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_STROKE_STATE *(workstation_id, device_number, value_type, num_elements, error_status, operating_mode, echo_flag, transformation_number, total_points, world_x_points, world_y_points, prompt_echo_type, echo_area, data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
num_elements	integer	M	Reference	Number of elements in the declared world_x_points and world_y_points array buffers
error_status	integer	O	Reference	Error status

Argument	Data Type	I/O	Passed by	Description
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)
transformation_number	integer	O	Reference	Normalization transformation
total_points	integer	O	Reference	Number of points in initial stroke
world_x_points	array (real)	O	Reference	Initial X world coordinates points of the stroke
world_y_points	array (real)	O	Reference	Initial Y world coordinates points of the stroke
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Four device coordinates specifying an echo area (XMIN, XMAX, YMIN, YMAX)
data_record	record	O	Reference	Pointer to stroke input data record
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE TEXT EXTENT

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_TEXT_EXTENT *(workstation_id, string_position_x, string_position_y, string, error_status, concatenation_x, concatenation_y, extent_rectangle_x, extent_rectangle_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
string_position_x	real	I	Reference	Starting X world coordinate value of the text string
string_position_y	real	I	Reference	Starting Y world coordinate value of the text string
string	string	I	Descriptor	The character string
error_status	integer	O	Reference	Error status
concatenation_x	real	O	Reference	X world coordinate value of the concatenation point

Argument	Data Type	I/O	Passed by	Description
concatenation_y	real	O	Reference	Y world coordinate value of the concatenation point
extent_rectangle_x	real	O	Reference	Array of four elements containing the X world coordinate values of the extent rectangle
extent_rectangle_y	real	O	Reference	Array of four elements containing the Y world coordinate values of the extent rectangle

INQUIRE TEXT REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_TEXT_REP *(workstation_id, text_index, value_type, error_status, text_font, text_precision, character_expansion_factor, character_spacing, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
text_index	integer	I	Reference	Text index
value_type	integer	I	Reference	Type of output values (GKSK_VALUE_SET, GKSK_VALUE_REALIZED)
error_status	integer	O	Reference	Error status
text_font	integer	O	Reference	Text font
text_precision	integer	O	Reference	Text precision (GKS$K_TEXT_PRECISION_STRING, GKS$K_TEXT_PRECISION_CHARACTER, GKS$K_TEXT_PRECISION_STROKE)

Workstation State List Inquiry Functions
INQUIRE TEXT REPRESENTATION

Argument	Data Type	I/O	Passed by	Description
character_expansion_ factor	real	O	Reference	Character expansion factor
character_spacing	real	O	Reference	Characater spacing
color_index	integer	O	Reference	Text color index

INQUIRE VALUATOR DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_VALUATOR_STATE *(workstation_id, device_number, error_status, operating_mode, echo_flag, default_value, prompt_echo_type, echo_area, data_record, record_buffer_length, record_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
device_number	integer	I	Reference	Device number
error_status	integer	O	Reference	Error status
operating_mode	integer	O	Reference	Operating input mode (GKS$K_INPUT_MODE_REQUEST, GKS$K_INPUT_MODE_SAMPLE, GKS$K_INPUT_MODE_EVENT)
echo_flag	integer	O	Reference	Echo flag (GKSK_NOECHO, GKSK_ECHO)

Workstation State List Inquiry Functions
INQUIRE VALUATOR DEVICE STATE

Argument	Data Type	I/O	Passed by	Description
default_value	real	O	Reference	Default value of valuator input device
prompt_echo_type	integer	O	Reference	Prompt and echo type
echo_area	array (real)	O	Reference	Four device coordinates specifying an echo area
data_record	record	O	Reference	Pointer to current valuator input data record
record_buffer_length	integer	M	Reference	Size of data record buffer, in bytes
record_size	integer	O	Reference	Total size, in bytes, of data record

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_DEFER_AND_UPDATE *(workstation_id, error_status, deferral_mode, regeneration_mode, surface_empty, new_frame_necessary)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
deferral_mode	integer	O	Reference	Deferral mode (GKSK_ASAP, GKSK_BNIG, GKSK_BNIL, GKSK_ASTI)
regeneration_mode	integer	O	Reference	Implicit regeneration mode (GKS$K_IRG_SUPPRESSED, GKS$K_IRG_ALLOWED)

Workstation State List Inquiry Functions
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

Argument	Data Type	I/O	Passed by	Description
surface_empty	integer	O	Reference	Display surface empty flag (GKSK_EMPTY, GKSK_NOEMPTY)
new_frame_necessary	integer	O	Reference	New frame necessary at update (GKS$K_NEWFRAME_NOTNECESSARY, GKS$K_NEWFRAME_NECESSARY)

INQUIRE WORKSTATION CONNECTION AND TYPE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_TYPE *(workstation_id, error_status, connection_logical_name, workstation_type, logical_return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
connection_logical_name	string	O	Descriptor	Connection identifier
workstation_type	integer	O	Reference	Workstation type
logical_return_size	integer	O	Reference	Return size, in bytes, of the string specifying the connection logical name

INQUIRE WORKSTATION STATE

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_STATE *(workstation_id, error_status, workstation_state)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
workstation_state	integer	O	Reference	Workstation state (GKS$K_WS_INACTIVE, GKS$K_WS_ACTIVE)

INQUIRE WORKSTATION TRANSFORMATION

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_WS_XFORM *(workstation_id, error_status, transformation_pending, requested_window, current_window, requested_viewport, current_viewport)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
error_status	integer	O	Reference	Error status
transformation_pending	integer	O	Reference	Transformation and update state (GKS$K_NOTPENDING, GKS$K_PENDING)
requested_window	array (real)	O	Reference	Four NDC points composing the requested workstation window (XMIN, XMAX, YMIN, YMAX)

Argument	Data Type	I/O	Passed by	Description
current_window	array (real)	O	Reference	Four NDC points composing the current workstation window (XMIN, XMAX, YMIN, YMAX)
requested_viewport	array (real)	O	Reference	Four device coordinates composing the requested workstation viewport (XMIN, XMAX, YMIN, YMAX)
current_viewport	array (real)	O	Reference	Four device coordinates composing the current workstation viewport (XMIN, XMAX, YMIN, YMAX)

INQUIRE SEGMENT ATTRIBUTES

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_SEG_ATTB *(segment_name, error_status, transformation_matrix, visibility, highlighting, priority, detectability)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
error_status	integer	O	Reference	Error status
transformation_matrix	array (real)	O	Reference	Six-element array containing segment transformation matrix
visibility	integer	O	Reference	Visibility (GKS$K_INVISIBLE, GKS$K_VISIBLE)
highlighting	integer	O	Reference	Highlighting (GKSK_NORMAL, GKSK_HIGHLIGHTED)
priority	real	O	Reference	Priority
detectability	integer	O	Reference	Detectability (GKS$K_UNDETECTABLE, GKS$K_DETECTABLE)

INQUIRE LIST OF ASSOCIATED WORKSTATIONS

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_SET_ASSOC_WS *(segment_name, error_status, num_workstations, list_workstations, return_size)*

Arguments

Argument	Data Type	I/O	Passed by	Description
segment_name	integer	I	Reference	Segment name
error_status	integer	O	Reference	Error status
num_workstations	integer	O	Reference	Number of associated workstations
list_workstations	array (integer)	O	Descriptor	List of workstation identifiers
return_size	integer	O	Reference	Number of workstation identifiers returned to the list

INQUIRE PIXEL

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PIXEL *(workstation_id, world_x, world_y, error_status, color_index)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
world_x	real	I	Reference	X world coordinate value of pixel
world_y	real	I	Reference	Y world coordinate value of pixel
error_status	integer	O	Reference	Error status
color_index	integer	O	Reference	Color index

INQUIRE PIXEL ARRAY

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PIXEL_ARRAY *(workstation_id, column_number, row_number, max_columns, max_rows, world_x, world_y, error_status, invalid_indexes_flag, color_index_array)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
column_number	integer	I	Reference	Starting column in color index array
row_number	integer	I	Reference	Starting row in color index array
max_columns	integer	I	Reference	Number of columns of pixels
max_rows	integer	I	Reference	Number of rows of pixels
world_x	real	I	Reference	X coordinate of upper left corner of the pixel array
world_y	real	I	Reference	Y coordinate of upper left corner of the pixel array

Argument	Data Type	I/O	Passed by	Description
error_status	integer	O	Reference	Error status
invalid_indexes_flag	integer	O	Reference	Invalid color index value flag (GKS$K_INVALID_ABSENT, GKS$K_INVALID_PRESENT)
color_index_array	2-D array (integer)	O	Descriptor	Color index array

INQUIRE PIXEL ARRAY DIMENSIONS

Operating States: WSOP, WSAC, SGOP

Syntax

GKS$INQ_PIXEL_ARRAY_DIM *(workstation_id, starting_point_x, starting_point_y, diagonal_point_x, diagonal_point_y, error_status, dimension_device_x, dimension_device_y)*

Arguments

Argument	Data Type	I/O	Passed by	Description
workstation_id	integer	I	Reference	Workstation identifier
starting_point_x	real	I	Reference	X world coordinate value of the starting point
starting_point_y	real	I	Reference	Y world coordinate value of the starting point
diagonal_point_x	real	I	Reference	X world coordinate value of the point diagonal to the starting point
diagonal_point_y	real	I	Reference	Y world coordinate value of the point diagonal to the starting point
error_status	integer	O	Reference	Error status
dimension_device_x	integer	O	Reference	Pixel array columns
dimension_device_y	integer	O	Reference	Pixel array rows

Appendix A

DEC GKS Function Names and FORTRAN Binding Function Names

A.1 DEC GKS Function Names and FORTRAN Binding Function Names

Appendix A lists the DEC GKS function names that use the GKS$ interface, and the corresponding FORTRAN binding name or names (if applicable). For a listing of the GKS$ binding constant names, refer to Appendix B, DEC GKS Constants, in the *DEC GKS Reference Manual*.

Table A–1: DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
Control Functions:	
GKS$ACTIVATE_WS	GACWK
GKS$CLEAR_WS	GCLRWK
GKS$CLOSE_GKS	GCLKS
GKS$CLOSE_WS	GCLWK
GKS$DEACTIVATE_WS	GDAWK
GKS$ESCAPE	GESC
GKS$MESSAGE_GKS	GMSG (GKS$ 77)
N/A	GMSGS (GKS$ 77 subset)

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$OPEN_GKS	GOPKS
GKS$OPEN_WS	GOPWK
GKS$REDRAW_SEG_ON_WS	GRSGWK
GKS$SET_DEFER_STATE	GSDS
GKS$UPDATE_WS	GUWK
Output Functions:	
GKS$CELL_ARRAY	GCA
GKS$FILL_AREA	GFA
GKS$GDP	GGDP
GKS$POLYLINE	GPL
GKS$POLYMARKER	GPM
GKS$TEXT	GTX (GKS$ 77)
N/A	GTXS (GKS$ 77 subset)
Attribute Functions:	
GKS$SET_ASF	GSASF
GKS$SET_COLOR_REP	GSCR
GKS$SET_FILL_COLOR_INDEX	GSFACI
GKS$SET_FILL_INDEX	GSFAI
GKS$SET_FILL_INT_STYLE	GSFAIS
GKS$SET_FILL_REP	GSFAR
GKS$SET_FILL_STYLE_INDEX	GSFASI
GKS$SET_PAT_REF_PT	GSPARF
GKS$SET_PAT_REP	GSPAR
GKS$SET_PAT_SIZE	GSPA
GKS$SET_PICK_ID	GSPKID
GKS$SET_PLINE_COLOR_INDEX	GSPLCI

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$SET_PLINE_INDEX	GSPLI
GKS$SET_PLINE_LINETYPE	GSLN
GKS$SET_PLINE_LINEWIDTH	GSLWSC
GKS$SET_PLINE_REP	GSPLR
GKS$SET_PMARK_COLOR_INDEX	GSPMCI
GKS$SET_PMARK_INDEX	GSPMI
GKS$SET_PMARK_REP	GSPMR
GKS$SET_PMARK_SIZE	GSMKSC
GKS$SET_PMARK_TYPE	GSMK
GKS$SET_TEXT_ALIGN	GSTXAL
GKS$SET_TEXT_COLOR_INDEX	GSTXCI
GKS$SET_TEXT_EXPFAC	GSCHXP
GKS$SET_TEXT_FONTPREC	GSTXFP
GKS$SET_TEXT_HEIGHT	GSCHH
GKS$SET_TEXT_INDEX	GSTXI
GKS$SET_TEXT_PATH	GSTXP
GKS$SET_TEXT_REP	GSTXR
GKS$SET_TEXT_SPACING	GSCHSP
GKS$SET_TEXT_UPVEC	GSCHUP
Transformation Functions:	
GKS$SELECT_XFORM	GSELNT
GKS$SET_CLIPPING	GSCLIP
GKS$SET_WINDOW	GSWN
GKS$SET_VIEWPORT	GSVP
GKS$SET_VIEWPORT_PRIORITY	GSVPIP
GKS$SET_WS_WINDOW	GSWKWN
GKS$SET_WS_VIEWPORT	GSWKVP

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
Segment Functions:	
GKS$ACCUM_XFORM_MATRIX	GACTM
GKS$ASSOC_SEG_WITH_WS	GASGWK
GKS$CLOSE_SEG	GCLSG
GKS$COPY_SEG_TO_WS	GCSGWK
GKS$CREATE_SEG	GCRSG
GKS$DELETE_SEG	GDSG
GKS$DELETE_SEG_FROM_WS	GDSGWK
GKS$EVAL_XFORM_MATRIX	GEVTM
GKS$INSERT_SEG	GINSG
GKS$RENAME_SEG	GRENSG
GKS$SET_SEG_DETECTABILITY	GSDTEC
GKS$SET_SEG_HIGHLIGHTING	GSHLIT
GKS$SET_SEG_PRIORITY	GSSGP
GKS$SET_SEG_VISIBILITY	GSVIS
GKS$SET_SEG_XFORM	GSSGT
Input Functions:	
GKS$AWAIT_EVENT	GWAIT
GKS$FLUSH_DEVICE_EVENTS	GFLUSH
GKS$GET_CHOICE	GGTCH
GKS$GET_LOCATOR	GGTLC
GKS$GET_PICK	GGTPK
GKS$GET_STRING	GGTST (GKS$ 77)
N/A	GGTST (GKS$ 77 subset)
GKS$GET_STROKE	GGTSK
GKS$GET_VALUATOR	GGTVL
GKS$INIT_CHOICE	GINCH

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$INIT_LOCATOR	GINLC
GKS$INIT_PICK	GINPK
GKS$INIT_STRING	GINST (GKS$ 77)
N/A	GINST (GKS$ 77 subset)
GKS$INIT_STROKE	GINSK
GKS$INIT_VALUATOR	GINVL
GKS$REQUEST_CHOICE	GRQCH
GKS$REQUEST_LOCATOR	GRQLC
GKS$REQUEST_PICK	GRQPK
GKS$REQUEST_STRING	GRQST (GKS$ 77)
N/A	GRQST (GKS$ 77 subset)
GKS$REQUEST_STROKE	GRQSK
GKS$REQUEST_VALUATOR	GRQVL
GKS$SAMPLE_CHOICE	GSMCH
GKS$SAMPLE_LOCATOR	GSMLC
GKS$SAMPLE_PICK	GSMPK
GKS$SAMPLE_STRING	GSMST (GKS$ 77)
N/A	GGTST (GKS$ 77 subset)
GKS$SAMPLE_STROKE	GSMSK
GKS$SAMPLE_VALUATOR	GSMVL
GKS$SET_CHOICE_MODE	GSCHM
GKS$SET_LOCATOR_MODE	GSLCM
GKS$SET_PICK_ID	GSPKID
GKS$SET_PICK_MODE	GSPKM
GKS$SET_STRING_MODE	GSSTM
GKS$SET_STROKE_MODE	GSSKM
GKS$SET_VALUATOR_MODE	GSVLM

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
Inquiry Functions	
GKS Description Table:	
GKS$INQ_LEVEL	GQLVKS
GKS$INQ_MAX_XFORM	GQMNTN
GKS$INQ_WS_MAX_NUM	GQWKM
GKS$INQ_WSTYPE_LIST	GQEWK
GKS State List:	
GKS$INQ_ACTIVE_WS	GQACWK
GKS$INQ_CLIP	GQCLIP
GKS$INQ_CURRENT_XFORMNO	GQCNTN
GKS$INQ_INDIV_ATTB	GQLN
N/A	GQLWSC
N/A	GQPLCI
N/A	GQMK
N/A	GQMKSC
N/A	GQPMCI
N/A	GQTXFP
N/A	GQCHXP
N/A	GQCHSP
N/A	GQTXCI
N/A	GQFAIS
N/A	GQFASI
N/A	GQFACI
N/A	GQASF
GKS$INQ_INPUT_QUEUE_OVERFLOW	GQIQOV
GKS$INQ_MORE_SIMUL_EVENTS	GQMORE
GKS$INQ_NAME_OPEN_SEG	GQOPSG

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$INQ_OPERATING_STATE	GQOPS
GKS$INQ_OPEN_WS	GQOPWK
GKS$INQ_PICK_ID	GQPKID
GKS$INQ_PRIM_ATTB	GQPLI
N/A	GQPMI
N/A	GQTXI
N/A	GQCHH
N/A	GQCHUP
N/A	GQCHW
N/A	GQCHB
N/A	GQTXP
N/A	GQTXAL
N/A	GQFAI
N/A	GQPA
N/A	GQPARF
GKS$INQ_SEG_NAMES	GQSGUS
GKS$INQ_XFORM	GQNT
GKS$INQ_XFORM_LIST	GQENTN
Workstation State List:	
GKS$INQ_CHOICE_STATE	GQCHS
GKS$INQ_COLOR_INDEXES	GQECI
GKS$INQ_COLOR_REP	GQCR
GKS$INQ_FILL_INDEXES	GQEFAI
GKS$INQ_FILL_REP	GQFAR
GKS$INQ_LOCATOR_STATE	GQLCS
GKS$INQ_PAT_INDEXES	GQEPAI
GKS$INQ_PAT_REP	GQPAR

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$INQ_PICK_STATE	GQPKS
GKS$INQ_PLINE_INDEXES	GQEPLI
GKS$INQ_PLINE_REP	GQPLR
GKS$INQ_PMARK_INDEXES	GQEPMI
GKS$INQ_PMARK_REP	GQPMR
GKS$INQ_SEG_NAMES_ON_WS	GQSGWK
GKS$INQ_STRING_STATE	GQSTS (GKS$ 77)
GKS$INQ_STRING_STATE	GQSTS (GKS$ 77 subset)
GKS$INQ_STROKE_STATE	GQSKS
GKS$INQ_TEXT_EXTENT	GQTXX (GKS$ 77)
N/A	GQTXXS (GKS$ 77 subset)
GKS$INQ_TEXT_INDEXES	GQETXI
GKS$INQ_TEXT_REP	GQTXR
GKS$INQ_VALUATOR_STATE	GQVLS
GKS$INQ_WS_DEFER_AND_UPDATE	GQWKDU
GKS$INQ_WS_STATE	GQWKS
GKS$INQ_WS_TYPE	GQWKC
GKS$INQ_WS_XFORM	GQWKT
Workstation Description Table:	
GKS$INQ_AVAIL_GDP	GQEGDP
GKS$INQ_COLOR_FAC	GQCF
GKS$INQ_DEF_CHOICE_DATA	GQDCH
GKS$INQ_DEF_DEFER_STATE	GQDDS
GKS$INQ_DEF_PICK_DATA	GQDPK
GKS$INQ_DEF_STRING_DATA	GQDST
GKS$INQ_DEF_STROKE_DATA	GQDSK
GKS$INQ_DEF_VALUATOR_DATA	GQDVL

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
GKS$INQ_DYN_MOD_SEG_ATTB	GQDSGA
GKS$INQ_DYN_MOD_WS_ATTB	GQDWKA
GKS$INQ_FILL_FAC	GQFAF
GKS$INQ_GDP	GQGDP
GKS$INQ_INPUT_DEV	GQLI
GKS$INQ_MAX_WS_STATE_TABLE	GQLWK
GKS$INQ_MAX_DS_SIZE	GQDSP
GKS$INQ_PAT_FAC	GQPAF
GKS$INQ_PLINE_FAC	GQPLF
GKS$INQ_PMARK_FAC	GQPMF
GKS$INQ_PREDEF_COLOR_REP	GQPCR
GKS$INQ_PREDEF_FILL_REP	GQPFAR
GKS$INQ_PREDEF_PAT_REP	GQPPAR
GKS$INQ_PREDEF_PLINE_REP	GQPPLR
GKS$INQ_PREDEF_PMARK_REP	GQPPMR
GKS$INQ_PREDEF_TEXT_REP	GQPTXR
GKS$INQ_SEG_PRIORITY	GQSGP
GKS$INQ_TEXT_FAC	GQTXF
GKS$INQ_WS_CATEGORY	GQWKCA
GKS$INQ_WS_CLASSIFICATION	GQWKCL
Segment:	
GKS$INQ_SEG_ASSOC_WS	GQASWK
GKS$INQ_SEG_ATTB	GQSGA

(continued on next page)

Table A–1 (Cont.): DEC GKS Function Names and Corresponding FORTRAN Binding Names

DEC GKS Function	GKS$ Binding Function(s)
Pixel:	
GKS$INQ_PIXEL	GQPX
GKS$INQ_PIXEL_ARRAY	GQPXA
GKS$INQ_PIXEL_ARRAY_DIM	GQPXAD
Metafile Functions:	
GKS$GET_ITEM	GGTITM
GKS$INTERPRET_ITEM	GIITM
GKS$READ_ITEM	GRDITM
GKS$WRITE_ITEM	GWITM
Error Functions:	
GKS$EMERGENCY_CLOSE	GECLKS
GKS$ERROR_HANDLER	GERHND
GKS$LOG_ERROR	GERLOG
Data Record Functions:	
N/A	GPREC
N/A	GUREC

Appendix B

DEC GKS Error Messages

This appendix lists each of the DEC GKS error messages, the DEC GKS error numbers, and the VMS completion status codes.

The VMS completion status codes correspond to the longword condition value returned by each DEC GKS function. You can compare the completion status codes directly to the function return values. In this way, you do not have to directly access the individual bits of the returned longword condition value to determine the cause of the error. Consider the following example:

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

C     Check for success.
      IF ( GKS$_SUCCESS = GKS$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
          .
          .
          .
```

The DEC GKS completion success status code symbol defined in the DEC GKS image library file is GKS$_SUCCESS. The remaining codes begin with the prefix DECGKS$_ERROR_NEG or GKS$_ERROR, and end with the number of the generated error.

Each of the condition status codes corresponds to the number of the appropriate DEC GKS error message. The GKS$_SUCCESS code is of severity *success*; all of the codes with positive numbers are of severity *error*; and, all negative errors are implementation-specific messages of severity *error* or *fatal error*.

If you choose, you can perform the normal VMS processing of the returned longword condition value by using LIB$SIGNAL, $GETMSG, or $PUTMSG. For detailed information concerning this type of processing, refer to *VAX/VMS Run-Time Library Routines Reference Manual.*

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.

B.1 Implementation-Specific Errors

All of the DEC GKS specific errors are negative in number; their condition status codes read DECGKS$_ERROR_NEG_number. These errors are either errors or fatal errors as described.

–2 Requested color map could not be created as specified in routine ****

DECGKS$_ERROR_NEG_2:

Error: Specified color map is too large.

User Action: Check to make sure that you specified the correct color map size and type (either physical or virtual). Remember the limitations of your VAXstation when reserving color indexes.

–3 Invalid data in workstation description file in routine ****

DECGKS$_ERROR_NEG_3:

Error: Workstation description file contains invalid data.

User Action: Make sure that the format of your description file is valid for your particular workstation.

–4 Invalid bit mask in workstation type in routine ****

DECGKS$_ERROR_NEG_4:

Error: Bit mask of the workstation type value is invalid.

User Action: Check to make sure that 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 ****

DECGKS$_ERROR_NEG_5:

Error: Illegal array of string pointers passed to the choice data record in routine ****

User Action: Make sure that you properly initialized the arrays containing string addresses and string lengths. Also, make sure that 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. For more information, refer to the program example for GKS$INQ_DEF_CHOICE_DATA in Chapter 12, Inquiry Functions, in the *DEC GKS Reference Manual.*

–6 Echo area is too narrow for data in routine ****

DECGKS$_ERROR_NEG_6:

Error: The specified input echo area minimum and maximum X values are too close in proximity.

User Action: Make sure that you did not swap X and Y values, and that your specified X values are of a greater distance from each other.

–7 Maximum number of representable choices exceeded in routine ****

DECGKS$_ERROR_NEG_7:

Error: The number of requested choices is too large for the workstation type.

User Action: You can use GKS$INQ_DEF_CHOICE_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 ****

DECGKS$_ERROR_NEG_8:

Error: The specified input echo area minimum and maximum Y values are too close in proximity.

User Action: Make sure that you did not swap X and Y values, and that your specified Y values are of a greater distance from each other.

–9 Binary format and integer number representation not supported in this implementation of GKS in routine ****

DECGKS$_ERROR_NEG_9:

Error: You opened a metafile of an incompatible type.

User Action: Check the metafile type.

–10 Invalid value specified for ASF in routine ****

DECGKS$_ERROR_NEG_10:

Error: You specified an incorrect value within the aspect source flag array.

User Action: Check the array to make sure that 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 ****

DECGKS$_ERROR_NEG_11:

Error: You did not specify a proper integer value for an interior style argument.

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_12:

Error: You did not specify a proper integer value for a horizontal text alignment argument.

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_13:

Error: You did not specify a proper integer value for a vertical text alignment argument.

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_14:

Error: You did not specify a proper integer value for a text precision argument.

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_15:

Error: You did not specify a proper integer value for a text path argument.

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_16:

Error: You did not specify a proper integer value for an echo switch in one of the arguments to the SET MODE input functions.

User Action: Make sure that you passed 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 ****

DECGKS$_ERROR_NEG_17:

Error: You neglected to specify GKS$K_VALUE_SET or GKS$K_VALUE_REALIZED when calling an inquiry function.

User Action: Check the value type argument to make sure that it is either GKS$K_VALUE_SET or GKS$K_VALUE_REALIZED.

–18 The following error occurred when GKS was interpreting an item ****

DECGKS$_ERROR_NEG_18:

Error: An error occurred while interpreting a metafile item.

User Action: DEC GKS follows this error message with another message that signals the appropriate action.

–19 Invalid error status parameter specified in routine ****

DECGKS$_ERROR_NEG_19:

Error: You passed an illegal error code to GKS$LOG_ERROR.

User Action: Make sure that the error code passed to GKS$LOG_ERROR is one of the codes described in this appendix.

–20 GKS not in proper state: GKS in the ERROR state in routine ****

DECGKS$_ERROR_NEG_20:

Error: You attempted to execute a DEC GKS function other than an error-handling or inquiry function.

User Action: 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 ****

DECGKS$_ERROR_NEG_21:

User Action: Remove the call to the unsupported function.

–22 Invalid segment transformation in routine ****

DECGKS$_ERROR_NEG_22:

Error: You specified an invalid transformation matrix.

User Action: Check your calls to GKS$EVAL_XFORM_MATRIX and to GKS$ACCUM_XFORM_MATRIX to make sure that you passed valid transformation components. Also, make sure that you specified a transformation matrix to GKS$SET_SEG_XFORM or to GKS$INSERT_SEG.

–23 Invalid value specified for clipping flag in routine ****

DECGKS$_ERROR_NEG_23:

User Action: Make sure that you passed either the value GKS$K_NOCLIP (0) or GKS$K_CLIP (1).

–24 Invalid value specified for viewport priority flag in routine ****

DECGKS$_ERROR_NEG_24:

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_25:

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_26:

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_27:

User Action: Make sure that 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 ****

DECGKS$_ERROR_NEG_28:

User Action: Check to make sure that 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 ****

DECGKS$_ERROR_NEG_29:

User Action: Check to make sure that 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 ****

DECGKS$_ERROR_NEG_30:

Error: DEC GKS could not activate your workstation handler's shareable image.

User Action: Make sure that your workstation handler is a valid, shareable image.

–31 Cannot load graphics handler: invalid DFT in routine ****

DECGKS$_ERROR_NEG_31:

Error: Your device function tables are incompatible.

User Action: You need to build your device function table again using the appropriate macro. For more information, refer to *Building a DEC GKS Workstation Handler System* or *Building a DEC GKS Device Handler System.*

–32 Font file for stroke precision text not found or unusable in routine ****

DECGKS$_ERROR_NEG_32:

Error: DEC GKS could not locate the specified stroke font.

User Action: Refer to the appropriate device-specific appendix in this manual to determine if the specified font is supported on your device. If you are not using a DEC GKS supported graphics handler, make sure that your handler defines the proper logical names, and that the logicals reference a valid file.

–33 Array descriptor is not acceptable in routine ****

DECGKS$_ERROR_NEG_33:

Error: An item in the array descriptor is either invalid or inconsistent.

User Action: Make sure that 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 of 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 that you have declared a two-dimensional array.

–34 String length less that or equal to 0 in routine ****

DECGKS$_ERROR_NEG_34:

Error: You specified an invalid character string.

User Action: Check the declaration, definition, or assignment statements involving the character variable.

–35 Kernel has detected an unexpected error from a device handler in routine ****

DECGKS$_ERROR_NEG_35:

Error: The device handler encountered an error.

User Action: 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 ****

DECGKS$_ERROR_NEG_36:

Error: DEC GKS could not activate your device handler's shareable image

User Action: Make sure that 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)™

™ PostScript is a registered trademark of Adobe Systems, Inc.

–37 Error in device handler during event flag allocation in routine ****

DECGKS$_ERROR_NEG_37:

Error: A graphics handler was unable to acquire all of its needed event flags.

User Action: The application must release event flags for use by the graphics handler.

–38 Error in device handler, cannot allocate device in routine ****

DECGKS$_ERROR_NEG_38:

Error: You used your graphics handler with an invalid physical device.

User Action: Make sure that you use the proper physical device or that you specify the correct workstation type value to GKS$OPEN_WS.

–39 Descriptor is not acceptable in routine ****

DECGKS$_ERROR_NEG_39:

User Action: Make sure that you have passed the variable by descriptor. If you have, and you use an inquiry function to initialize the variable, make sure that all of the arguments are specified to the inquiry function in the correct order.

–40 Illegal device pointer, in routine ****

DECGKS$_ERROR_NEG_40:

User Action: Check your handler code for null pointers or otherwise invalid pointers.

–41 Driver handler WDT is invalid in routine ****

DECGKS$_ERROR_NEG_41:

Error: You illegally defined a workstation description table entry.

User Action: 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 ****

DECGKS$_ERROR_NEG_42:

Error: You improperly defined the logical name.

User Action: Make sure that the translation of GKS$LIST_TYPES is as expected.

–43 VAX Workstation Software is not present, workstation type is invalid in routine ****

DECGKS$_ERROR_NEG_43:

Explanation: Check to make sure either that you specify the correct workstation type when opening a non-VAXstation workstation, 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 that you install the VAXstation Windowing Software.

The following errors are fatal errors. Should one occur, submit a Software Performance Report (SPR) indicating the error number, corresponding message, and any relevant particulars. For more information concerning SPRs, refer to the *DEC GKS Installation Guide*.

–90 Internal GKS error: Bad memory address freed in routine ****

DECGKS$_ERROR_NEG_90:

Fatal: DEC GKS memory data structures were corrupted.

User Action: Submit an SPR.

–91 Internal GKS error: Invalid function pointer parameter in error handler in routine ****

DECGKS$_ERROR_NEG_91:

Fatal: A DEC GKS internal data structure was corrupted.

User Action: Submit an SPR.

–92 Internal GKS error: Insufficient virtual memory in routine ****

DECGKS$_ERROR_NEG_92:

Fatal: DEC GKS was unable to allocate enough virtual memory.

User Action: Check to make sure that the problem is not caused by storing too much in segment storage or by defining a very large cell array. If you cannot reduce storage by checking segments and cell arrays, submit an SPR.

–93 Internal GKS error: Prompt and echo type not supported in routine ****

DECGKS$_ERROR_NEG_93:

Fatal:

User Action: Submit an SPR.

–94 Internal GKS error: Corrupted segment memory in routine ****

DECGKS$_ERROR_NEG_94:

Fatal:

User Action: Submit an SPR.

–95 Internal GKS error: Negative size passed to allocate memory in routine ****

DECGKS$_ERROR_NEG_95:

Fatal: An invalid size was passed to the DEC GKS memory allocation routines.

User Action: If you generate this error using a user-written graphics handler, make sure that 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 ****

DECGKS$_ERROR_NEG_96:

Fatal:

User Action: Submit an SPR.

–97 Internal GKS error: Insufficient buffer size for translated logical name in routine ****

DECGKS$_ERROR_NEG_97:

Fatal:

User Action: Submit an SPR.

–98 Internal GKS error: Too many translations of logical name in routine ****

DECGKS$_ERROR_NEG_98:

Fatal: You may have recursively defined a logical name.

User Action: 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 ****

DECGKS$_ERROR_NEG_99:

Fatal:

User Action: Submit an SPR.

–100 Internal GKS error: Device handler received unexpected input access in routine ****

DECGKS$_ERROR_NEG_100:

Fatal:

User Action: Submit an SPR.

–150 Edge index is less than zero in routine ****

DECGKS$_ERROR_NEG_150:

User Action:

–151 Edge width calse factor is less than zero ****

DECGKS$_ERROR_NEG_151:

User Action:

–152 Text font/precision cannot be named in routine ****

DECGKS$_ERROR_NEG_152:

User Action:

–153 Text font name is invalid in routine ****

DECGKS$_ERROR_NEG_153:

User Action:

–154 A representation for the specified edge index has not been predefined on this workstation in routine ****

DECGKS$_ERROR_NEG_154:

User Action:

–155 Display speed is less than zero in routine ****

DECGKS$_ERROR_NEG_155:

User Action: Pass a positive real value to GKS$K_ESC_SET_SPEED.

–156 Loudness is outside range [0,1] in routine ****

DECGKS$_ERROR_NEG_156:

User Action: Pass a valid value to GKS$K_ESC_BEEP.

–157 Duration is less than zero in routine ****

DECGKS$_ERROR_NEG_157:

User Action: Make sure that your duration value is greater than or equal to zero.

–158 GDP primitive is not defined by the supplied data in routine ****

DECGKS$_ERROR_NEG_158:

Error: DEC GKS is unable to form the desired primitive.

User Action: Refer to the error message listing in the description of the GDP that generated the error (Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual*). This listing gives specific information concerning the primitive you attempted to draw.

–159 Arc type is invalid in routine ****

DECGKS$_ERROR_NEG_159:

User Action: Refer to the error message listing in the description of the GDP that generated the error (Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual*). This listing gives specific information concerning the primitive you attempted to draw.

–160 Insufficient space in escape output data record arrays in routine ****

DECGKS$_ERROR_NEG_160:

Error: You passed addresses of arrays that were too small to contain the data to be written to them.

User Action: 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 ****

DECGKS$_ERROR_NEG_161:

Error: You specified text attributes that were too large to fill the text in the bounding box (the extent rectangle).

User Action: Use a larger bounding box, or reduce the text height or the character expansion factor.

–300 Invalid value specified for highlighting in routine ****

DECGKS$_ERROR_NEG_300:

User Action: Make sure that you specify either GKS$K_NORMAL (0) or GKS$K_HIGHLIGHTED (1).

–301 Invalid value specified for visibility in routine ****

DECGKS$_ERROR_NEG_301:

User Action: Make sure that you specify either GKS$K_INVISIBLE (0) or GKS$K_VISIBLE (1).

–302 Invalid value specified for detectability in routine ****

DECGKS$_ERROR_NEG_302:

User Action: Make sure that you specify either GKS$K_UNDETECTABLE (0) or GKS$K_DETECTABLE (1).

–303 Input device can not be activated due to conflict with another input device that is currently active in routine ****

DECGKS$_ERROR_NEG_303:

–304 Cannot set input device echo on due to conflict with other input devices active in the same echo area in routine ****

DECGKS$_ERROR_NEG_304:

–306 The definition of GKS$HPGL_THRESHOLD is invalid (contains nonnumeric values in routine) ****

DECGKS$_ERROR_NEG_306:

User Action: Check the definition of GKS$HPGL_THRESHOLD and redefine to range 0 to 1023.

B.2 Operating State Errors

This section 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, refer to Chapter 4, Control Functions, in *DEC GKS Reference Manual.*

1 GKS not in proper state: GKS shall be in the state GKCL in routine ****

GKS$_ERROR_1:

Error: You called a function unsupported in the current operating state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call GKS$CLOSE_GKS before the current DEC GKS state changes to GKS$K_GKCL.)

2 GKS not in proper state: GKS shall be in the state GKOP in routine ****

GKS$_ERROR_2:

Error: You called a function unsupported in the current operating state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call either the function

GKS$OPEN_GKS or GKS$CLOSE_WS before the DEC GKS state changes to GKS$K_GKOP.)

3 GKS not in proper state: GKS shall be in the state WSAC in routine ****

GKS$_ERROR_3:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call either the function GKS$ACTIVATE_WS or GKS$CLOSE_SEG before the DEC GKS state changes to GKS$K_WSAC.)

4 GKS not in proper state: GKS shall be in the state SGOP in routine ****

GKS$_ERROR_4:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function GKS$CREATE_SEG before the DEC GKS state changes to GKS$K_SGOP.)

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP in routine ****

GKS$_ERROR_5:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function GKS$ACTIVATE_WS before the DEC GKS state changes to GKS$K_WSAC.)

6 GKS not in proper state: GKS shall be in the state WSOP or in the state WSAC in routine ****

GKS$_ERROR_6:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function GKS$OPEN_WS before the DEC GKS state changes to GKS$K_WSOP.)

7 GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine ****

GKS$_ERROR_7:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function GKS$OPEN_WS before the DEC GKS state changes to GKS$K_WSOP.)

8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine ****

GKS$_ERROR_8:

Error: You called a function unsupported in the current state.

User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function GKS$OPEN_WS before the DEC GKS state changes to GKS$K_WSOP.)

B.3 Workstation Errors

This section lists the errors that result when you call a DEC GKS function with invalid or undefined arguments pertaining to workstations.

20 Specified workstation identifier is invalid in routine ****

GKS$_ERROR_20:

User Action: Make sure that 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 ****

GKS$_ERROR_21:

User Action: Make sure that the specified connection exists and is allocated to your process (by typing SHOW DEVICES at the DCL command line), that the workstation type supports the specified connection identifier (especially in the case of output-only workstations that write information to files, such as

GKS$K_VT_OUTPUT), and that the arguments are specified in the correct order.

22 Specified workstation type is invalid in routine ****

GKS$_ERROR_22:

User Action: Check to make sure that you passed either a DEC GKS constant (GKS$K_WSTYPE_DEFAULT, GKS$K_VT241), or the corresponding integer values.

23 Specified workstation type does not exist in routine ****

GKS$_ERROR_23:

Error: The implementation of GKS does not support a device handler associated with the identifier you passed.

User Action: Pass an identifier associated with a supported device. If you are using the constant GKS$K_WSTYPE_DEFAULT, you should use GKS$INQ_WS_TYPE to check to see if DEC GKS supports the currently defined workstation type.

24 Specified workstation is open in routine ****

GKS$_ERROR_24:

Error: You tried to reopen a workstation.

User Action: Either remove the function call to GKS$OPEN_WS, or replace the incorrect workstation-type argument.

25 Specified workstation is not open in routine ****

GKS$_ERROR_25:

Error: You tried to input or generate output on a closed workstation.

User Action: Call GKS$OPEN_WS and pass the appropriate workstation identifier.

26 Specified workstation cannot be opened in routine ****

GKS$_ERROR_26:

User Action: Make sure that you specify valid workstation types, bit masks, or logical name definitions (GKS$CONID and GKS$WSTYPE), and make sure that the information corresponds to a supported, functional physical device.

27 Workstation Independent Segment Storage is not open in routine ****

GKS$_ERROR_27:

Error: You tried to copy, associate, or insert a segment from WISS to another workstation.

User Action: Make sure that you have opened WISS in a call to GKS$OPEN_WS, passing GKS$K_WSTYPE_WISS as an argument.

28 Workstation Independent Segment Storage is already open in routine ****

GKS$_ERROR_28:

User Action: Either remove the function call to GKS$OPEN_WS, or replace the incorrect workstation-type argument.

29 Specified workstation is active in routine ****

GKS$_ERROR_29:

Error: You tried to activate a workstation twice.

User Action: Either remove the function call to GKS$ACTIVATE_WS, or replace the incorrect workstation-type argument.

30 Specified workstation is not active in routine ****

GKS$_ERROR_30:

Error: You tried to generate output on an inactive workstation.

User Action: Call GKS$ACTIVATE_WS passing the appropriate workstation.

31 Specified workstation is of category MO in routine ****

GKS$_ERROR_31:

Error: You attempted to perform an operation that is not permissible on MO workstations.

User Action: Either remove the function call, change the state of the MO workstation, or check to see if you passed the correct arguments to GKS$OPEN_WS.

32 Specified workstation is not of category MO in routine ****

GKS$_ERROR_32:

User Action: Open and activate an MO workstation.

33 Specified workstation is of category MI in routine ****

GKS$_ERROR_33:

Error: You attempted to perform an operation that is not permissible on MI workstations.

User Action: Either remove the function call, change the state of the MI workstation, or check to see if you passed the correct arguments to GKS$OPEN_WS.

34 Specified workstation is not of category MI in routine ****

GKS$_ERROR_34:

Error: You tried to interpret a file that was not associated with an MI workstation.

User Action: Open a workstation of category MI.

35 Specified workstation is of category INPUT in routine ****

GKS$_ERROR_35:

Error: You attempted to perform an operation that is not permissible on workstations of category INPUT, such as generating output.

User Action: Either remove the function call, change the state of the INPUT workstation, or check to see if you passed the correct arguments to GKS$OPEN_WS.

36 Specified workstation is Workstation Independent Segment Storage in routine ****

GKS$_ERROR_36:

Error: You attempted to perform an operation that is not permissible on workstations of category WISS, such as requesting input.

User Action: Either remove the function workstation identifier or check to see if you passed the correct arguments to GKS$OPEN_WS.

37 Specified workstation is not of category OUTIN in routine ****

GKS$_ERROR_37:

Error: You attempted to perform an operation that is only permissible on workstations of category OUTIN.

User Action: Either remove the function call, open and activate an OUTIN workstation, or check to see if you passed the correct arguments to GKS$OPEN_WS.

38 Specified workstation is neither of category INPUT nor of category OUTIN in routine ****

GKS$_ERROR_38:

Error: You attempted to perform an operation that is only permissible on workstations of category INPUT and OUTIN, such as requesting input.

User Action: Either remove the function call, change the state of the INPUT workstation, or check to see if you passed the correct arguments to GKS$OPEN_WS.

39 Specified workstation is neither of category OUTPUT nor of category OUTIN in routine ****

GKS$_ERROR_39:

Error: You attempted to perform an operation that is only permissible on workstations of category OUTPUT or OUTIN, such as generating output.

User Action: 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 GKS$OPEN_WS.

40 Specified workstation has no pixel store readback capability in routine ****

GKS$_ERROR_40:

Error: You called one of the pixel inquiry functions for a device incapable of returning such information.

User Action: Either remove the function call, or make sure that you passed the correct workstation identifier.

41 Specified workstation type is not able to generate the specified generalized drawing primitive in routine ****

GKS$_ERROR_41:

User Action: Either remove the function call to GKS$GDP, or make sure that you passed the correct GDP identifier.

42 Maximum number of simultaneously open workstations would be exceeded in routine ****

GKS$_ERROR_42:

User Action: You must remove the function call to GKS$OPEN_WS. You can use GKS$INQ_WS_MAX_NUM to determine the maximum number of open workstations that DEC GKS supports.

43 Maximum number of simultaneously active workstations would be exceeded in routine ****

GKS$_ERROR_43:

User Action: You must remove the function call to GKS$ACTIVATE_WS. You can use GKS$INQ_WS_MAX_NUM to determine the maximum number of active workstations that DEC GKS supports.

B.4 Transformation Function Errors

This section lists the errors that result when you call a DEC GKS transformation function with invalid or undefined arguments.

50 Transformation number is invalid in routine ****

GKS$_ERROR_50:

User Action: Either make sure that the arguments are specified in the correct order, that the transformation number is not negative, or that the transformation number is an integer.

51 Rectangle definition is invalid in routine ****

GKS$_ERROR_51:

Error: Either the normalization window or viewport is invalid.

User Action: Either make sure that 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 ****

GKS$_ERROR_52:

Error: DEC GKS allows unclipped primitives to exceed the NDC unit square ([0,1] x [0,1]), but does not allow you to define a normalization viewport whose boundaries exceed this square.

User Action: Redefine the function normalization viewport.

53 Workstation window is not within the Normalized Device Coordinate unit square in routine ****

GKS$_ERROR_53:

User Action: Redefine the function normalization viewport to be within the NDC square ([0,1] x [0,1]).

54 Workstation viewport is not within the display space in routine ****

GKS$_ERROR_54:

User Action: Either make sure that 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. You can use the function GKS$INQ_MAX_DS_SIZE to determine the maximum X and Y values of the device coordinate plane.

B.5 Output Attribute Errors

This section lists the errors that result when you call the DEC GKS output attribute functions with invalid or undefined arguments.

60 Polyline index is invalid in routine ****

GKS$_ERROR_60:

User Action: Make sure that the arguments are specified in the correct order and that the index is an integer.

61 A representation for the specified polyline index has not been defined on this workstation in routine ****

GKS$_ERROR_61:

User Action: Use GKS$SET_PLINE_REP to define a representation for the index, or use another, defined index value.

62 A representation for the specified polyline index has not been predefined on this workstation in routine ****

GKS$_ERROR_62:

User Action: Use GKS$SET_PLINE_REP to define a representation for the index, or use another, predefined index value.

63 Specified linetype is equal to zero in routine ****

GKS$_ERROR_63:

User Action: Make sure that 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 linetype is not supported on this workstation in routine ****

GKS$_ERROR_64:

Error: You specified a line type value that is workstation dependent but is not supported by the specified workstation.

User Action: Change the line type specification. You can use the function GKS$INQ_PLINE_FAC to obtain a list of supported line types for a given workstation.

65 Linewidth scale factor is less than zero in routine ****

GKS$_ERROR_65:

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 ****

GKS$_ERROR_66:

User Action: Make sure that the arguments are specified in the correct order and that the index is an integer.

67 A representation for the specified polymarker index has not been defined on this workstation in routine ****

GKS$_ERROR_67:

User Action: Use GKS$SET_PMARK_REP to define a representation for a given index, or use another, defined index value.

68 A representation for the specified polymarker index has not been predefined on this workstation in routine ****

GKS$_ERROR_68:

User Action: Use GKS$SET_PMARK_REP to define a representation for a given index, or use another, predefined index value.

69 Specified marker type is equal to zero in routine ****

GKS$_ERROR_69:

User Action: Make sure that 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 ****

GKS$_ERROR_70:

Error: You specified a marker type value that is workstation dependent but is not supported by the specified workstation.

User Action: Change the marker type specification. You can use the function GKS$INQ_PMARK_FAC to obtain a list of supported line types for a given workstation.

71 Marker size scale factor is less than zero in routine ****

GKS$_ERROR_71:

User Action: Either change the scale factor, or check the order and the number of the specified arguments.

72 Text index is invalid in routine ****

GKS$_ERROR_72:

User Action: Make sure that the arguments are specified in the correct order and that the index is an integer.

73 A representation for the specified text index has not been defined on this workstation in routine ****

GKS$_ERROR_73:

User Action: Use GKS$SET_TEXT_REP to define a representation for the index value, or use another, defined index value.

74 A representation for the specified text index has not been predefined on this workstation in routine ****

GKS$_ERROR_74:

User Action: Use GKS$SET_TEXT_REP to define a representation for the index value, or use another, predefined index value.

75 Text font is equal to zero in routine ****

GKS$_ERROR_75:

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 ****

GKS$_ERROR_76:

User Action: Lower the precision or change the font number.

77 Character expansion factor is less than or equal to zero in routine ****

GKS$_ERROR_77:

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 ****

GKS$_ERROR_78:

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 ****

GKS$_ERROR_79:

User Action: 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 ****

GKS$_ERROR_80:

User Action: Make sure that the arguments are specified in the correct order and that the index is an integer.

81 A representation for the specified fill area index has not been defined on this workstation in routine ****

GKS$_ERROR_81:

User Action: Use GKS$SET_FILL_REP to define a representation for the given index value, or pass another, defined index value.

82 A representation for the specified fill area index has not been predefined on this workstation in routine ****

GKS$_ERROR_82:

User Action: Use GKS$SET_FILL_REP 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 ****

GKS$_ERROR_83:

Error: You specified a fill area interior style value that is workstation-dependent but is not supported by the specified workstation.

User Action: Change the interior style specification. You can use

the function GKS$INQ_FILL_FAC to obtain a list of supported interior styles for a given workstation.

84 Style (pattern or hatch) index is equal to zero in routine ****

GKS$_ERROR_84:

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 ****

GKS$_ERROR_85:

User Action: Make sure that the arguments are specified in the correct order and that the index is an integer.

86 Specified hatch style is not supported on this workstation in routine ****

GKS$_ERROR_86:

User Action: Either replace the hatch style index, or check the order and the number of the arguments. The inquiry function GKS$INQ_FILL_FAC returns the list of available hatch style indexes.

87 Pattern size value is not positive in routine ****

GKS$_ERROR_87:

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 ****

GKS$_ERROR_88:

User Action: Use GKS$SET_PAT_REP to define a representation for the pattern index, or pass another, defined index to the function.

89 A representation for the specified pattern index has not been predefined on this workstation in routine ****

GKS$_ERROR_89:

User Action: Use GKS$SET_PAT_REP to define a representation for the pattern index, or pass another, predefined index to the function.

90 Interior style PATTERN is not supported on this workstation in routine ****

GKS$_ERROR_90:

User Action: Specify another interior style to GKS$SET_FILL_INT_STYLE.

91 Dimensions of color array are invalid in routine ****

GKS$_ERROR_91:

Error: One or more of the arguments passed to GKS$CELL_ARRAY are invalid.

User Action: Make sure that the color array is a two-dimensional array. Also, make sure 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. Also, make sure that the cell array contains integers representing colors supported on that workstation.

92 Color index is less than zero in routine ****

GKS$_ERROR_92:

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 index value, check the order and the number of the arguments passed to the inquiry function.

93 Color index is invalid in routine ****

GKS$_ERROR_93:

User Action: Make sure that 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 ****

GKS$_ERROR_94:

User Action: Use GKS$SET_COLOR_REP to define a color representation for the index value, or pass another, defined index value.

95 A representation for the specified color index has not been predefined on this workstation in routine ****

GKS$_ERROR_95:

User Action: Use GKS$SET_COLOR_REP to define a color representation for the index value, or pass another, defined index value.

96 Color index is outside range [0,1] in routine ****

GKS$_ERROR_96:

User Action: Specify either the value 0 or 1 for the color index value.

97 Pick identifier is invalid in routine ****

GKS$_ERROR_97:

User Action: Either remove the call to GKS$SET_PICK_ID or make sure that 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.

B.6 Output Function Errors

This section lists the errors that result when you call a DEC GKS output function with invalid or undefined arguments.

100 Number of points is invalid in routine ****

GKS$_ERROR_100:

Error: The number of points specified does not match the number of coordinate points passed.

User Action: 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 ****

GKS$_ERROR_101:

Error: Your text string contained characters that cannot be printed.

User Action: Remove the characters.

102 Generalized drawing primitive identifier is invalid in routine ****

GKS$_ERROR_102:

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 ****

GKS$_ERROR_103:

User Action: Make sure that you passed a correct size as the data record size.

104 At least one active workstation is not able to generate the specified generalized drawing primitive in routine ****

GKS$_ERROR_104:

User Action: Deactivate the workstations that do not generate the GDPs, or redefine the GDP data record so that all of 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 ****

GKS$_ERROR_105:

User Action: Either redefine the current normalization transformation (creating a different clipping rectangle), or supply different world coordinate points so that the GDP falls within the current clipping rectangle.

B.7 Segment Function Errors

This section lists the errors that result when you call a DEC GKS segment function with invalid or undefined arguments.

120 Specified segment name is invalid in routine ****

GKS$_ERROR_120:

User Action: Either check the number and the order of the arguments or make sure that 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 ****

GKS$_ERROR_121:

User Action: Either remove the call to GKS$CREATE_SEG or check to make sure that you specified the correct segment name.

122 Specified segment does not exist in routine ****

GKS$_ERROR_122:

User Action: Either check the order and the number of the arguments or make sure that 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 ****

GKS$_ERROR_123:

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 ****

GKS$_ERROR_124:

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 ****

GKS$_ERROR_125:

User Action: Either remove the call to GKS$CREATE_SEG or specify another segment name.

126 Segment priority is outside the range [0,1] in routine ****

GKS$_ERROR_126:

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.

B.8 Input Function Errors

This section lists the errors that result when you call a DEC GKS input function with invalid or undefined arguments.

140 Specified input device is not present on workstation in routine ****

GKS$_ERROR_140:

User Action: Make sure that 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 ****

GKS$_ERROR_141:

User Action: Use one of the SET MODE input functions to set request mode before using this logical input device.

142 Input device is not in SAMPLE mode in routine ****

GKS$_ERROR_142:

User Action: Use one of the SET MODE input functions to set to sample mode before using this logical input device.

143 EVENT and SAMPLE input mode are not available at this level of GKS in routine ****

GKS$_ERROR_143:

User Action: DEC GKS does not generate this error.

144 Specified prompt and echo type is not supported on this workstation in routine ****

GKS$_ERROR_144:

User Action: Make sure that 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 ****

GKS$_ERROR_145:

User Action: Make sure that 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 ****

GKS$_ERROR_146:

User Action: Make sure that 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 that 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 ****

GKS$_ERROR_147:

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 ****

GKS$_ERROR_148:

Error: You called GKS$INQ_INPUT_QUEUE_OVERFLOW when the queue was not full, and had not been filled since the beginning of your application.

User Action: Continue to generate events, if your application still requires input.

149 Input queue has overflowed, but associated workstation has been closed in routine ****

GKS$_ERROR_149:

Error: You called GKS$INQ_INPUT_QUEUE_OVERFLOW when the queue was full, but since the workstation is closed, information about the overflow is not available.

User Action: 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 ****

GKS$_ERROR_150:

User Action: Make sure that you check the input class argument passed to GKS$AWAIT_EVENT before you try to call the appropriate GET function.

151 Timeout is invalid in routine ****

GKS$_ERROR_151:

User Action: Make sure that the timer argument in GKS$AWAIT_EVENT is a real value between 0.0 and 356,400, specified in the format described in the GKS$AWAIT_EVENT function description in Chapter 8, Input Functions, in the *DEC GKS Reference Manual.*

152 Initial value is invalid in routine ****

GKS$_ERROR_152:

User Action: Either check to make sure that 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 ****

GKS$_ERROR_153:

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 ****

GKS$_ERROR_154:

User Action: Either increase the size of the buffer or decrease the size of the initial string.

B.9 Metafile Function Errors

This section lists the errors that result when you call a DEC GKS metafile function with invalid or undefined arguments.

160 Item type is not allowed for user items in routine ****

GKS$_ERROR_160:

Error: You used an item type less than 101 to write to a GKSM.

User Action: Use an item type greater than or equal to 101.

161 Item length is invalid in routine ****

GKS$_ERROR_161:

Error: The length of the data item was shorter than necessary for its type.

User Action: Make sure that 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 ****

GKS$_ERROR_162:

Error: You tried to read past the end of the GKSM.

User Action: Do not attempt to read items past the item of type 0.

163 Metafile item is invalid in routine ****

GKS$_ERROR_163:

Error: Your item data was incorrect.

User Action: Make sure that DEC GKS did not truncate the item while reading from a GKSM and that you specified correct sizes and types. Make sure that you are not trying to interpret a user-defined record type. User-defined records have item numbers greater than 100.

164 Item type is not a valid GKS item in routine ****

GKS$_ERROR_164:

Error: You tried to interpret an item of type less than 0 or greater than 100.

User Action: Make sure that 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 ****

GKS$_ERROR_165:

Error: There was unexpected or incorrect information in the data record.

User Action: Make sure that you pass the correct storage area.

166 Maximum item data record length is invalid in routine ****

GKS$_ERROR_166:

User Action: Make sure that the data length is not negative.

167 User item cannot be interpreted in routine ****

GKS$_ERROR_167:

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 ****

GKS$_ERROR_168:

User Action: DEC GKS does not generate this error.

B.10 Escape Function Errors

This section lists the errors that result when you call a DEC GKS escape function with invalid or undefined arguments.

180 Specified escape function is not supported in routine ****

GKS$_ERROR_180:

User Action: Check the escape function identifier to make sure that it is a valid integer representing an escape function, and make sure that you specified the correct workstation identifier.

181 Specified escape function identifier is invalid in routine ****

GKS$_ERROR_181:

User Action: Make sure that the escape function identifier is a valid integer value.

182 Contents of escape data record are invalid in routine ****

GKS$_ERROR_182:

User Action: Make sure that you specified the correct size of the data record. Also, make sure that the elements of the data record are declared to be the correct data type.

B.11 Miscellaneous Errors

This section lists the DEC GKS miscellaneous errors.

200 Specified error file is invalid in routine ****

GKS$_ERROR_200:

User Action: Make sure that your specified error handler exists and that it includes the three required parameters in its definition.

B.12 System Errors

This section lists implementation-dependent errors.

300 Storage overflow has occurred in GKS ****

GKS$_ERROR_300:

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 index value, check the order and the number of the arguments passed to the inquiry function.

301 Storage overflow has occurred in segment storage ****

GKS$_ERROR_301:

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 index value, check the order and the number of the arguments passed to the inquiry function.

302 Input/Output error has occurred while reading in routine ****

GKS$_ERROR_302:

Error: You specified an illegal metafile for a metafile input workstation.

User Action: Make sure that you work with a valid GKSM metafile, and that you correctly specify the connection identifier.

303 Input/Output error has occurred while writing in routine ****

GKS$_ERROR_303:

Error: You specified an illegal metafile for a metafile output workstation.

User Action: Make sure that you work with a valid GKSM Metafile, and that you correctly specify the connection identifier.

304 Input/Output error has occurred while sending data to a workstation ****

GKS$_ERROR_304:

User Action:

305 Input/Output error has occurred while receiving data from a workstation ****

GKS$_ERROR_305:

User Action:

306 Input/Output error has occurred during program library management ****

GKS$_ERROR_307:

User Action:

307 Input/Output error has occurred while reading workstation description table ****

GKS$_ERROR_307:

User Action:

308 Arithmetic error has occurred in routine ****

GKS$_ERROR_308:

Error: You either divided by zero or caused data overflow.

User Action: Check the arguments passed in the function call.

B.13 FORTRAN Binding Errors

This section lists those error messages that are specific to the FORTRAN binding functions.

2000 Enumeration type out of range–the INTEGER passed as a GKS enumerated type is not within the range of valid values in routine ****

GKS$_ERROR_2000:

User Action: Make sure that you properly define the enumerated values.

2001 Output parameter size insufficient–a FORTRAN array or string being passed as an output parameter is too small to contain the returned information in routine ****

GKS$_ERROR_2001:

User Action: Redefine the size of the output string, or check the order or number of arguments passed to the function.

2002 List or set element not available–for a nonempty list or set, a value less than zero or greater than the size of the list or set was passed as the requested list or set element in an inquiry routine ****

GKS$_ERROR_2002:

User Action: Either check the declaration of the value passed, or check the order and number of all arguments passed.

2003 Invalid data record—the data record passed to a GKS routine cannot be decoded, or there was a problem encountered when GKS was creating a data record, making the result invalid in routine ****

GKS$_ERROR_2003:

User Action: Make sure that you used the function GPREC to pack and create the data record. Also, check the order and number of arguments passed to the function. If you used an inquiry function to obtain the data record value, check the order and number of arguments passed to the inquiry function.

Index

A

Access type, 1–6
Accumulate Transformation Matrix, 10–9
Activate Workstation, 4–2
ANSI
  GKS standard, 1–1
Arguments
  characteristics, 1–6
  GKS$ binding functions, 4–1, 5–1, 6–1, 7–1, 8–1, 9–1, 10–1, 11–1
  list, 1–3, 3–3
    C binding, 1–3, 3–3
    FORTRAN binding, 1–3, 3–3
Associate Segment with Workstation, 9–2
Attribute functions, 6–1 to 6–33
  GKS$SET_ASF, 6–2
  GKS$SET_COLOR_REP, 6–4
  GKS$SET_FILL_COLOR_INDEX, 6–5
  GKS$SET_FILL_INDEX, 6–6
  GKS$SET_FILL_INT_STYLE, 6–7
  GKS$SET_FILL_REP, 6–8
  GKS$SET_FILL_STYLE_INDEX, 6–9
  GKS$SET_PAT_REF_PT, 6–14
  GKS$SET_PAT_REP, 6–15
  GKS$SET_PAT_SIZE, 6–16
  GKS$SET_PICK_ID, 6–17
  GKS$SET_PLINE_COLOR_INDEX, 6–18
  GKS$SET_PLINE_INDEX, 6–19
  GKS$SET_PLINE_LINETYPE, 6–10
  GKS$SET_PLINE_LINEWIDTH, 6–11
  GKS$SET_PLINE_REP, 6–20
  GKS$SET_PMARK_COLOR_INDEX, 6–21
  GKS$SET_PMARK_INDEX, 6–22
  GKS$SET_PMARK_REP, 6–23
  GKS$SET_PMARK_SIZE, 6–12
Attribute functions (cont'd.)
  GKS$SET_PMARK_TYPE, 6–13
  GKS$SET_TEXT_ALIGN, 6–25
  GKS$SET_TEXT_COLOR_INDEX, 6–26
  GKS$SET_TEXT_EXPFAC, 6–3
  GKS$SET_TEXT_FONTPREC, 6–27
  GKS$SET_TEXT_HEIGHT, 6–28
  GKS$SET_TEXT_INDEX, 6–29
  GKS$SET_TEXT_PATH, 6–30
  GKS$SET_TEXT_REP, 6–31
  GKS$SET_TEXT_SPACING, 6–32
  GKS$SET_TEXT_UPVEC, 6–33
Attributes
  output
    list of errors, B–24 to B–31
Await Event, 8–2

B

Bindings, 1–1
Bit masks, 3–7

C

Calling sequences, 1–3, 3–2
CALL statement, 1–3, 3–2
Cell Array, 5–2
Clear Workstation, 4–3
Close GKS, 4–4
Close Segment, 9–3
Close Workstation, 4–5
Compile
  FORTRAN binding functions
    VMS, 2–2
Compiling
  ULTRIX programs, 3–6

Completion status codes, B–1
Conditions
  error, 10–1, B–1 to B–42
Connection identifiers
  GKSconid, 3–6
Constants
  arguments, 1–4, 3–4
  for errors, 1–8
  requirements, 1–5, 3–5
Control functions, 4–1 to 4–13
  GKS$ACTIVATE_WS, 4–2
  GKS$CLEAR_WS, 4–3
  GKS$CLOSE_GKS, 4–4
  GKS$CLOSE_WS, 4–5
  GKS$DEACTIVATE_WS, 4–6
  GKS$ESCAPE, 4–7
  GKS$MESSAGE, 4–8
  GKS$OPEN_GKS, 4–9
  GKS$OPEN_WS, 4–10
  GKS$REDRAW_SEG_ON_WS, 4–11
  GKS$SET_DEFER_STATE, 4–12
  GKS$UPDATE_WS, 4–13
Copy Segment to Workstation, 9–4
Create Segment, 9–5
Current
  state list entries, 6–1

D

Data records
  escape/GDP
    standard, 1–8
Data types
  arguments, 1–6
Deactivate Workstation, 4–6
Decimal
  workstation type value, 3–7
Declaring
  GKS functions
    externally, 3–3
Definition files, 1–4, 3–4
  including, 1–5, 3–5
  list of, 3–5
Delete Segment, 9–6
Delete Segment from Workstation, 9–7

E

Emergency Close GKS, 10–6
Entries
  GKS state list
    output attributes, 6–1
Environment variables
  GKS programming, 3–6
Error Handling, 10–7
Error-handling functions, 1–7
Error-Handling functions, 10–6 to 10–8
  GKS$EMERGENCY_CLOSE, 10–6
  GKS$ERROR_HANDLER, 10–7
  GKS$LOG_ERROR, 10–8
Error Logging, 10–8
Errors
  constants, 1–8
  messages, B–1 to B–42
    escape functions, B–39
    FORTRAN binding, B–41 to B–42
    implementation-specific, B–2 to B–13
    input, B–34 to B–37
    metafiles, B–37 to B–39
    miscellaneous, B–39
    operating state, B–16 to B–18
    output, B–31 to B–32
    output attributes, B–24 to B–31
    segments, B–32 to B–34
    system, B–40 to B–41
    transformations, B–23 to B–24
    workstation, B–18 to B–23
  status files, 1–4, 3–4
Error status files
  list of, 3–5
Escape, 4–7
Escape functions
  errors
    list of, B–39
Escapes
  data records, 1–8
Evaluate Transformation Matrix, 10–11
Executing
  FORTRAN binding functions
    VMS, 2–2
External functions
  declaring GKS functions, 3–3

F

Files
  definition, 1–4, 3–4
    list of, 3–5
  error status, 1–4, 3–4
    list of, 3–5

Fill Area, 5–3
Flush Device Events, 8–3
FORTRAN binding, 1–1
  errors
    list of, B–41 to B–42
  GKS$ function names, A–1
  VMS specific, 2–1 to 2–2
Functions
  attribute, 6–1
  control, 4–1
  DEC GKS organization, 1–7
  error-handling, 10–6
  external
    declaring, 3–3
  FORTRAN binding and GKS$ names, A–1
  GKS$ binding, 4–1, 5–1, 6–1, 7–1, 8–1, 9–1, 10–1, 11–1
  identifiers, 1–3, 3–2
  input, 8–1
  inquiry, 11–1
  metafile, 10–1
  output, 5–1
  segment, 9–1
  transformation, 7–1
  utility, 10–8

G

GDPs
  data records, 1–8
Generalized Drawing Primitive, 5–4
Get Choice, 8–4
Get Item Type from GKSM, 10–2
Get Locator, 8–5
Get Pick, 8–6
Get String, 8–7
Get Stroke, 8–8
Get Valuator, 8–9
GKS
  ANSI and ISO standards, 1–1
  environment variables, 3–6
  HELP, 1–2
  input
    levels of, 1–1
  introduction to, 1–1 to 1–10
  levels, 1–1
  operating state
    errors, B–16 to B–18
  organization of functions, 1–7
  output
    levels of, 1–1
GKS (cont'd.)
  programming, 1–2 to 1–5, 3–1 to 3–8
  release notes, 1–2
  state list
    output attributes, 6–1
GKS$ACCUM_XFORM_MATRIX, 10–9
GKS$ASSOC_SEG_WITH_WS, 9–2
GKS$AWAIT_EVENT, 8–2
GKS$CELL_ARRAY, 5–2
GKS$CLEAR_WS, 4–3
GKS$CLOSE_GKS, 4–4
GKS$CLOSE_SEG, 9–3
GKS$CLOSE_WS, 4–5
GKS$COPY_SEG_TO_WS, 9–4
GKS$CREATE_SEG, 9–5
GKS$DEACTIVATE_WS, 4–6
GKS$DELETE_SEG, 9–6
GKS$DELETE_SEG_FROM_WS, 9–7
GKS$EMERGENCY_CLOSE, 10–6
GKS$ERROR_HANDLER, 10–7
GKS$ESCAPE, 4–7
GKS$EVAL_XFORM_MATRIX, 10–11
GKS$FILL_AREA, 5–3
GKS$FLUSH_DEVICE_EVENTS, 8–3
GKS$GDP, 5–4
GKS$GET_CHOICE, 8–4
GKS$GET_ITEM, 10–2
GKS$GET_LOCATOR, 8–5
GKS$GET_PICK, 8–6
GKS$GET_STRING, 8–7
GKS$GET_STROKE, 8–8
GKS$GET_VALUATOR, 8–9
GKS$INIT_CHOICE, 8–10
GKS$INIT_LOCATOR, 8–11
GKS$INIT_PICK, 8–12
GKS$INIT_STRING, 8–13
GKS$INIT_STROKE, 8–14
GKS$INIT_VALUATOR, 8–16
GKS$INQ_ACTIVE_WS, 11–61
GKS$INQ_AVAIL_GDP, 11–28
GKS$INQ_CHOICE_STATE, 11–64
GKS$INQ_CLIP, 11–48
GKS$INQ_COLOR_FAC, 11–6
GKS$INQ_COLOR_INDEXES, 11–68
GKS$INQ_COLOR_REP, 11–66
GKS$INQ_CURRENT_XFORMNO, 11–53
GKS$INQ_DEF_CHOICE_DATA, 11–7
GKS$INQ_DEF_DEFER_STATE, 11–9
GKS$INQ_DEF_LOCATOR_DATA, 11–10
GKS$INQ_DEF_PICK_DATA, 11–12
GKS$INQ_DEF_STRING_DATA, 11–14

GKS$INQ_DEF_STROKE_DATA, 11–16
GKS$INQ_DEF_VALUATOR_DATA, 11–18
GKS$INQ_DYN_MOD_SEG_ATTB, 11–21
GKS$INQ_DYN_MOD_WS_ATTB, 11–23
GKS$INQ_FILL_FAC, 11–25
GKS$INQ_FILL_INDEXES, 11–69
GKS$INQ_FILL_REP, 11–67
GKS$INQ_GDP, 11–27
GKS$INQ_INDIV_ATTB, 11–50
GKS$INQ_INPUT_DEV, 11–30
GKS$INQ_INPUT_QUEUE_OVERFLOW, 11–49
GKS$INQ_LEVEL, 11–2
GKS$INQ_LOCATOR_STATE, 11–74
GKS$INQ_MAX_DS_SIZE, 11–20
GKS$INQ_MAX_WS_STATE_TABLE, 11–29
GKS$INQ_MAX_XFORM, 11–3
GKS$INQ_MORE_SIMUL_EVENTS, 11–52
GKS$INQ_NAME_OPEN_SEG, 11–57
GKS$INQ_OPEN_WS, 11–62
GKS$INQ_OPERATING_STATE, 11–58
GKS$INQ_PAT_FAC, 11–32
GKS$INQ_PAT_INDEXES, 11–70
GKS$INQ_PAT_REP, 11–76
GKS$INQ_PICK_ID, 11–54
GKS$INQ_PICK_STATE, 11–78
GKS$INQ_PIXEL, 11–103
GKS$INQ_PIXEL_ARRAY, 11–104
GKS$INQ_PIXEL_ARRAY_DIM, 11–106
GKS$INQ_PLINE_FAC, 11–33
GKS$INQ_PLINE_INDEXES, 11–71
GKS$INQ_PLINE_REP, 11–80
GKS$INQ_PMARK_FAC, 11–35
GKS$INQ_PMARK_INDEXES, 11–72
GKS$INQ_PMARK_REP, 11–82
GKS$INQ_PREDEF_COLOR_REP, 11–37
GKS$INQ_PREDEF_FILL_REP, 11–38
GKS$INQ_PREDEF_PAT_REP, 11–39
GKS$INQ_PREDEF_PLINE_REP, 11–41
GKS$INQ_PREDEF_PMARK_REP, 11–42
GKS$INQ_PREDEF_TEXT_REP, 11–43
GKS$INQ_PRIM_ATTB, 11–55
GKS$INQ_SEG_ATTB, 11–101
GKS$INQ_SEG_NAMES, 11–63
GKS$INQ_SEG_NAMES_ON_WS, 11–84
GKS$INQ_SEG_PRIORITY, 11–31
GKS$INQ_SET_ASSOC_WS, 11–102
GKS$INQ_STRING_STATE, 11–85
GKS$INQ_STROKE_STATE, 11–87
GKS$INQ_TEXT_EXTENT, 11–89
GKS$INQ_TEXT_FAC, 11–44
GKS$INQ_TEXT_INDEXES, 11–73
GKS$INQ_TEXT_REP, 11–91
GKS$INQ_VALUATOR_STATE, 11–93
GKS$INQ_WSTYPE_LIST, 11–5
GKS$INQ_WS_CATEGORY, 11–46
GKS$INQ_WS_CLASSIFICATION, 11–47
GKS$INQ_WS_DEFER_AND_UPDATE, 11–95
GKS$INQ_WS_MAX_NUM, 11–4
GKS$INQ_WS_STATE, 11–98
GKS$INQ_WS_TYPE, 11–97
GKS$INQ_WS_XFORM, 11–99
GKS$INQ_XFORM, 11–59
GKS$INQ_XFORM_LIST, 11–60
GKS$INSERT_SEG, 9–8
GKS$INTERPRET_ITEM, 10–3
GKS$LOG_ERROR, 10–8
GKS$MESSAGE, 4–8
GKS$OPEN_GKS, 4–9
GKS$OPEN_WS, 4–10
GKS$POLYLINE, 5–5
GKS$POLYMARKER, 5–6
GKS$READ_ITEM, 10–4
GKS$REDRAW_SEG_ON_WS, 4–11
GKS$RENAME_SEG, 9–9
GKS$REQUEST_LOCATOR, 8–18
GKS$REQUEST_PICK, 8–17, 8–19
GKS$REQUEST_STRING, 8–20
GKS$REQUEST_STROKE, 8–21
GKS$REQUEST_VALUATOR, 8–23
GKS$SAMPLE_CHOICE, 8–24
GKS$SAMPLE_LOCATOR, 8–25
GKS$SAMPLE_PICK, 8–26
GKS$SAMPLE_STRING, 8–27
GKS$SAMPLE_STROKE, 8–28
GKS$SAMPLE_VALUATOR, 8–30
GKS$SELECT_XFORM, 7–2
GKS$SET_ASF, 6–2
GKS$SET_CHOICE_MODE, 8–31
GKS$SET_CLIPPING, 7–3
GKS$SET_COLOR_REP, 6–4
GKS$SET_DEFER_STATE, 4–12
GKS$SET_FILL_COLOR_INDEX, 6–5
GKS$SET_FILL_INDEX, 6–6
GKS$SET_FILL_INT_STYLE, 6–7
GKS$SET_FILL_REP, 6–8
GKS$SET_FILL_STYLE_INDEX, 6–9
GKS$SET_LOCATOR_MODE, 8–32
GKS$SET_PAT_REF_PT, 6–14
GKS$SET_PAT_REP, 6–15
GKS$SET_PAT_SIZE, 6–16
GKS$SET_PICK_ID, 6–17
GKS$SET_PICK_MODE, 8–33

GKS$SET_PLINE_COLOR_INDEX, 6–18
GKS$SET_PLINE_INDEX, 6–19
GKS$SET_PLINE_LINETYPE, 6–10
GKS$SET_PLINE_LINEWIDTH, 6–11
GKS$SET_PLINE_REP, 6–20
GKS$SET_PMARK_COLOR_INDEX, 6–21
GKS$SET_PMARK_INDEX, 6–22
GKS$SET_PMARK_REP, 6–23
GKS$SET_PMARK_SIZE, 6–12
GKS$SET_PMARK_TYPE, 6–13
GKS$SET_SEG_DETECTABILITY, 9–10
GKS$SET_SEG_HIGHLIGHTING, 9–11
GKS$SET_SEG_PRIORITY, 9–12
GKS$SET_SEG_VISIBILITY, 9–13
GKS$SET_SEG_XFORM, 9–14
GKS$SET_STRING_MODE, 8–34
GKS$SET_STROKE_MODE, 8–35
GKS$SET_TEXT_ALIGN, 6–25
GKS$SET_TEXT_COLOR_INDEX, 6–26
GKS$SET_TEXT_EXPFAC, 6–3
GKS$SET_TEXT_FONTPREC, 6–27
GKS$SET_TEXT_HEIGHT, 6–28
GKS$SET_TEXT_INDEX, 6–29
GKS$SET_TEXT_PATH, 6–30
GKS$SET_TEXT_REP, 6–31
GKS$SET_TEXT_SPACING, 6–32
GKS$SET_TEXT_UPVEC, 6–33
GKS$SET_VALUATOR_MODE, 8–36
GKS$SET_VIEWPORT, 7–4
GKS$SET_VIEWPORT_PRIORITY, 7–5
GKS$SET_WINDOW, 7–6
GKS$SET_WS_VIEWPORT, 7–7
GKS$SET_WS_WINDOW, 7–8
GKS$TEXT, 5–7
GKS$UPDATE_WS, 4–13
GKS$WRITE_ITEM, 10–5
GKSconid, 3–6
GKSwstype, 3–6

H

Handlers
  for errors, 1–7
HELP
  GKS, 1–2

I

Implementation specific errors
  list of, B–2 to B–13
Include
  definition files, 1–5, 3–5
  files, 1–4, 3–4
INCLUDE statement
  all languages, 3–5
Initialize Choice, 8–10
Initialize Locator, 8–11
Initialize Pick, 8–12
Initialize String, 8–13
Initialize Stroke, 8–14
Initialize Valuator, 8–16
Input
  errors
    list of, B–34 to B–37
Input functions, 8–1 to 8–36
  GKS$AWAIT_EVENT, 8–2
  GKS$FLUSH_DEVICE_EVENTS, 8–3
  GKS$GET_CHOICE, 8–4
  GKS$GET_LOCATOR, 8–5
  GKS$GET_PICK, 8–6
  GKS$GET_STRING, 8–7
  GKS$GET_STROKE, 8–8
  GKS$GET_VALUATOR, 8–9
  GKS$INIT_CHOICE, 8–10
  GKS$INIT_LOCATOR, 8–11
  GKS$INIT_PICK, 8–12
  GKS$INIT_STRING, 8–13
  GKS$INIT_STROKE, 8–14
  GKS$INIT_VALUATOR, 8–16
  GKS$REQUEST_LOCATOR, 8–18
  GKS$REQUEST_PICK, 8–17, 8–19
  GKS$REQUEST_STRING, 8–20
  GKS$REQUEST_STROKE, 8–21
  GKS$REQUEST_VALUATOR, 8–23
  GKS$SAMPLE_CHOICE, 8–24
  GKS$SAMPLE_LOCATOR, 8–25
  GKS$SAMPLE_PICK, 8–26
  GKS$SAMPLE_STRING, 8–27
  GKS$SAMPLE_STROKE, 8–28
  GKS$SAMPLE_VALUATOR, 8–30
  GKS$SET STROKE_MODE, 8–35
  GKS$SET_CHOICE_MODE, 8–31
  GKS$SET_LOCATOR_MODE, 8–32
  GKS$SET_PICK_MODE, 8–33
  GKS$SET_STRING_MODE, 8–34
  GKS$SET_VALUATOR_MODE, 8–36
Inquire Aspect Source Flags, 11–50
Inquire Character Base Vector, 11–55
Inquire Character Expansion Factor, 11–50
Inquire Character Height, 11–55
Inquire Character Spacing, 11–50

Inquire Character Up Vector, 11–55
Inquire Character Width, 11–55
Inquire Choice Device State, 11–64
Inquire Clipping, 11–48
Inquire Color Facilities, 11–6
Inquire Color Representation, 11–66
Inquire Current Normalization Transformation Number, 11–53
Inquire Current Pick Identifier, 11–54
Inquire Current Primitive Attribute Values, 11–55
Inquire Default Choice Data, 11–7
Inquire Default Deferral State Values, 11–9
Inquire Default Locator Device Data, 11–10
Inquire Default Pick Device Data, 11–12
Inquire Default String Device Data, 11–14
Inquire Default Stroke Device Data, 11–16
Inquire Default Valuator Device Data, 11–18
Inquire Display Space Size, 11–20
Inquire Dynamic Modification of Segment Attributes, 11–21
Inquire Dynamic Modification of Workstation Attributes, 11–23
Inquire Fill Area Color Index, 11–50
Inquire Fill Area Facilities, 11–25
Inquire Fill Area Interior Style, 11–50, 11–55
Inquire Fill Area Representation, 11–67
Inquire Fill Area Style Index, 11–50
Inquire Generalized Drawing Primitive, 11–27
Inquire Input Queue Overflow, 11–49
Inquire Level of GKS, 11–2
Inquire Linetype, 11–50
Inquire Linewidth Scale Factor, 11–50
Inquire List of Associated Workstations, 11–102
Inquire List of Available Generalized Drawing Primitives, 11–28
Inquire List of Available Workstation Types, 11–5
Inquire List of Color Indexes, 11–68
Inquire List of Fill Area Indexes, 11–69
Inquire List of Normalization Transformation Numbers, 11–60
Inquire List of Pattern Indexes, 11–70
Inquire List of Polyline Indexes, 11–71
Inquire List of Polymarker Indexes, 11–72
Inquire List of Text Indexes, 11–73
Inquire Locator Device State, 11–74
Inquire Marker Size Scale Factor, 11–50
Inquire Markertype, 11–50
Inquire Maximum Length of Workstation State, 11–29
Inquire Maximum Normalization Transformation Number, 11–3
Inquire More Simultaneous Events, 11–52
Inquire Name of Open Segment, 11–57
Inquire Normalization Transformation Number, 11–59
Inquire Number of Available Logical Input Devices, 11–30
Inquire Number of Segment Priorities Supported, 11–31
Inquire Operating State Value, 11–58
Inquire Pattern Facilities, 11–32
Inquire Pattern Reference Point, 11–55
Inquire Pattern Representation, 11–76
Inquire Pattern Size, 11–55
Inquire Pick Device State, 11–78
Inquire Pixel, 11–103
Inquire Pixel Array, 11–104
Inquire Pixel Array Dimensions, 11–106
Inquire Polyline Color Index, 11–50
Inquire Polyline Facilities, 11–33
Inquire Polyline Index, 11–50, 11–55
Inquire Polyline Representation, 11–80
Inquire Polymarker Color Index, 11–50
Inquire Polymarker Facilities, 11–35
Inquire Polymarker Index, 11–55
Inquire Polymarker Representation, 11–82
Inquire Predefined Color Representation, 11–37
Inquire Predefined Fill Representation, 11–38
Inquire Predefined Pattern Representation, 11–39
Inquire Predefined Polyline Representation, 11–41
Inquire Predefined Polymarker Representation, 11–42
Inquire Predefined Text Representation, 11–43
Inquire Segment Attributes, 11–101
Inquire Set of Active Workstations, 11–61
Inquire Set of Open Workstations, 11–62
Inquire Set of Segment Names in Use, 11–63
Inquire Set of Segment Names on Workstation, 11–84
Inquire String Device State, 11–85
Inquire Stroke Device State, 11–87
Inquire Text Alignment, 11–55
Inquire Text Color Index, 11–50
Inquire Text Extent, 11–89
Inquire Text Facilities, 11–44
Inquire Text Font and Precision, 11–50
Inquire Text Index, 11–55
Inquire Text Path, 11–55
Inquire Text Representation, 11–91
Inquire Valuator Device State, 11–93
Inquire Workstation Category, 11–46
Inquire Workstation Classification, 11–47
Inquire Workstation Connection and Type, 11–97
Inquire Workstation Deferral and Update States, 11–95
Inquire Workstation Maximum Numbers, 11–4

Inquire Workstation State, 11–98
Inquire Workstation Transformation, 11–99
Inquiry functions, 11–1 to 11–106
GKS$INQ_ACTIVE_WS, 11–61
GKS$INQ_AVAIL_GDP, 11–28
GKS$INQ_CHOICE_STATE, 11–64
GKS$INQ_CLIP, 11–48
GKS$INQ_COLOR_FAC, 11–6
GKS$INQ_COLOR_INDEXES, 11–68
GKS$INQ_COLOR_REP, 11–66
GKS$INQ_CURRENT_XFORMNO, 11–53
GKS$INQ_DEF_CHOICE_DATA, 11–7
GKS$INQ_DEF_DEFER_STATE, 11–9
GKS$INQ_DEF_LOCATOR_DATA, 11–10
GKS$INQ_DEF_PICK_DATA, 11–12
GKS$INQ_DEF_STRING_DATA, 11–14
GKS$INQ_DEF_STROKE_DATA, 11–16
GKS$INQ_DEF_VALUATOR_DATA, 11–18
GKS$INQ_DYN_MOD_SEG_ATTB, 11–21
GKS$INQ_DYN_MOD_WS_ATTB, 11–23
GKS$INQ_FILL_FAC, 11–25
GKS$INQ_FILL_INDEXES, 11–69
GKS$INQ_FILL_REP, 11–67
GKS$INQ_FORM, 11–59
GKS$INQ_GDP, 11–27
GKS$INQ_INDIV_ATTB, 11–50
GKS$INQ_INPUT_DEV, 11–30
GKS$INQ_INPUT_QUEUE_OVERFLOW, 11–49
GKS$INQ_LEVEL, 11–2
GKS$INQ_LOCATOR_STATE, 11–74
GKS$INQ_MAX_DS_SIZE, 11–20
GKS$INQ_MAX_WS_STATE_TABLE, 11–29
GKS$INQ_MAX_XFORM, 11–3
GKS$INQ_MORE_SIMUL_EVENTS, 11–52
GKS$INQ_NAME_OPEN_SEG, 11–57
GKS$INQ_OPEN_WS, 11–62
GKS$INQ_OPERATING_STATE, 11–58
GKS$INQ_PAT_FAC, 11–32
GKS$INQ_PAT_INDEXES, 11–70
GKS$INQ_PAT_REP, 11–76
GKS$INQ_PICK_ID, 11–54
GKS$INQ_PICK_STATE, 11–78
GKS$INQ_PIXEL, 11–103
GKS$INQ_PIXEL_ARRAY, 11–104
GKS$INQ_PIXEL_ARRAY_DIM, 11–106
GKS$INQ_PLINE_FAC, 11–33
GKS$INQ_PLINE_INDEXES, 11–71
GKS$INQ_PLINE_REP, 11–80
GKS$INQ_PMARK_FAC, 11–35
GKS$INQ_PMARK_INDEXES, 11–72
GKS$INQ_PMARK_REP, 11–82
Inquiry functions (cont'd.)
GKS$INQ_PREDEF_COLOR_REP, 11–37
GKS$INQ_PREDEF_FILL_REP, 11–38
GKS$INQ_PREDEF_PAT_REP, 11–39
GKS$INQ_PREDEF_PLINE_REP, 11–41
GKS$INQ_PREDEF_PMARK_REP, 11–42
GKS$INQ_PREDEF_TEXT_REP, 11–43
GKS$INQ_PRIM_ATTB, 11–55
GKS$INQ_SEG_ATTB, 11–101
GKS$INQ_SEG_NAMES, 11–63
GKS$INQ_SEG_NAMES_ON_WS, 11–84
GKS$INQ_SEG_PRIORITY, 11–31
GKS$INQ_SET_ASSOC_WS, 11–102
GKS$INQ_STRING_STATE, 11–85
GKS$INQ_STROKE_STATE, 11–87
GKS$INQ_TEXT_EXTENT, 11–89
GKS$INQ_TEXT_FAC, 11–44
GKS$INQ_TEXT_INDEXES, 11–73
GKS$INQ_TEXT_REP, 11–91
GKS$INQ_VALUATOR_STATE, 11–93
GKS$INQ_WSTYPE_LIST, 11–5
GKS$INQ_WS_CATEGORY, 11–46
GKS$INQ_WS_CLASSIFICATION, 11–47
GKS$INQ_WS_DEFER_AND_UPDATE, 11–95
GKS$INQ_WS_MAX_NUM, 11–4
GKS$INQ_WS_STATE, 11–98
GKS$INQ_WS_TYPE, 11–97
GKS$INQ_WS_XFORM, 11–99
GKS$INQ_XFORM_LIST, 11–60
Insert Segment, 9–8
Interpret Item, 10–3

L

Languages
argument data types, 1–3, 3–3
bindings, 1–1
calling sequences, 1–3, 3–2
declaring external functions, 3–3
GKS, 1–2, 3–1
Levels
of GKS, 1–1
Linking
FORTRAN binding functions
VMS, 2–2
ULTRIX GKS$ programs, 3–6
ULTRIX programs, 3–6
Lists
argument, 1–3, 3–3

M

Message, 4–8
Messages
  error, B–1 to B–42
Metafile functions, 10–1 to 10–5
  GKS$GET_ITEM, 10–2
  GKS$INTERPRET_ITEM, 10–3
  GKS$READ_ITEM, 10–4
  GKS$WRITE_ITEM, 10–5
Metafiles
  errors
    list of, B–37 to B–39

N

Names
  FORTRAN binding and GKS$, A–1
Numbers
  error, B–1

O

Open GKS, 4–9
Open Workstation, 4–10
Operating states
  errors
    list of, B–16 to B–18
Operating system
  ULTRIX, 3–1
Output
  errors
    list of, B–31 to B–32
Output attributes
  See Attributes
Output functions, 5–1 to 5–7
  GKS$CELL_ARRAY, 5–2
  GKS$FILL_AREA, 5–3
  GKS$GDP, 5–4
  GKS$POLYLINE, 5–5
  GKS$POLYMARKER, 5–6
  GKS$TEXT, 5–7

P

Passing mechanisms
  arguments, 1–6
Polyline, 5–5
Polymarker, 5–6
Programming
  GKS, 1–2, 3–1
Programs
  logical names, 3–6

R

Read Item from GKSM, 10–4
Records
  escape/GDP data, 1–8
Redraw All Segments on Workstation. 4–11
Release notes
  GKS, 1–2
Rename Segment, 9–9
Request Choice, 8–17
Request Locator, 8–18
Request Pick, 8–19
Request String, 8–20
Request Stroke, 8–21
Request Valuator, 8–23
Running
  FORTRAN binding functions
    VMS, 2–2

S

Sample Choice, 8–24
Sample Locator, 8–25
Sample Pick, 8–26
Sample String, 8–27
Sample Stroke, 8–28
Sample Valuator, 8–30
Segment functions, 9–1 to 9–14
  GKS$ASSOC_SEG_WITH_WS, 9–2
  GKS$CLOSE_SEG, 9–3
  GKS$COPY_SEG_TO_WS, 9–4
  GKS$CREATE_SEG, 9–5
  GKS$DELETE_SEG, 9–6
  GKS$DELETE_SEG_FROM_WS, 9–7
  GKS$INSERT_SEG, 9–8
  GKS$RENAME_SEG, 9–9
  GKS$SET_SEG_DETECTABILITY, 9–10
  GKS$SET_SEG_HIGHLIGHTING, 9–11
  GKS$SET_SEG_PRIORITY, 9–12
  GKS$SET_SEG_VISIBILITY, 9–13
  GKS$SET_SEG_XFORM, 9–14
Segments
  errors
    list of, B–32 to B–34
Select Normalization Transformation, 7–2
Set Aspect Source Flags, 6–2
Set Character Expansion Factor, 6–3

Set Character Height
See Set Text Height, 6–28
Set Character Spacing
See Set Text Spacing, 6–32
Set Character Up Vector
See Set Text Up Vector, 6–33
Set Choice Mode, 8–31
Set Clipping Indicator, 7–3
Set Color Representation, 6–4
Set Deferral State, 4–12
Set Detectability, 9–10
Set Fill Area Color Index, 6–5
Set Fill Area Index, 6–6
Set Fill Area Interior Style, 6–7
Set Fill Area Representation, 6–8
Set Fill Area Style Index, 6–9
Set Highlighting, 9–11
Set Linetype, 6–10
Set Linewidth Scale Factor, 6–11
Set Locator Mode, 8–32
Set Marker Size Scale Factor, 6–12
Set Marker Type, 6–13
Set Pattern Reference Point, 6–14
Set Pattern Representation, 6–15
Set Pattern Size, 6–16
Set Pick Identifier, 6–17
Set Pick Mode, 8–33
Set Polyline Color Index, 6–18
Set Polyline Index, 6–19
Set Polyline Representation, 6–20
Set Polymarker Color Index, 6–21
Set Polymarker Index, 6–22
Set Polymarker Representation, 6–23
Set Segment Priority, 9–12
Set Segment Transformation, 9–14
Set String Mode, 8–34
Set Stroke Mode, 8–35
Set Text Alignment, 6–25
Set Text Color Index, 6–26
Set Text Font and Precision, 6–27
Set Text Height, 6–28
Set Text Index, 6–29
Set Text Path, 6–30
Set Text Representation, 6–31
Set Text Spacing, 6–32
Set Text Up Vector, 6–33
Set Valuator Mode, 8–36
Set Viewport, 7–4
Set Viewport Input Priority, 7–5
Set Visibility, 9–13
Set Window, 7–6
Set Workstation Viewport, 7–7
Set Workstation Window, 7–8
Standards
DEC GKS escape/GDP data records, 1–8
functional vs. syntactical, 1–1
State lists
GKS
output attributes, 6–1
Statements
CALL, 1–3, 3–2
INCLUDE, 3–5
Status
values
VMS, B–1
stderr, 3–6
System errors
list of, B–40 to B–41

T

Text, 5–7
Transformation functions, 7–1 to 7–8
GKS$SELECT_XFORM, 7–2
GKS$SET_CLIPPING, 7–3
GKS$SET_VIEWPORT, 7–4
GKS$SET_VIEWPORT_PRIORITY, 7–5
GKS$SET_WINDOW, 7–6
GKS$SET_WS_VIEWPORT, 7–7
GKS$SET_WS_WINDOW, 7–8
Transformations
errors
list of, B–23 to B–24
TTY, 3–6

U

ULTRIX operating system, 3–1 to 3–8
Update Workstation, 4–13
Utility functions, 10–8 to 10–11
GKS$ACCUM_XFORM_MATRIX, 10–9
GKS$EVAL_XFORM_MATRIX, 10–11

V

VAX
languages, 1–2
VAX languages, 3–1
VMS
FORTRAN binding specific, 2–1

W

Workstations
  errors
    list of, B–18 to B–23
  types
    decimal, 3–7
Write Item to GKSM, 10–5

Reader's Comments

DEC GKS GKS$
Binding Reference Manual
AA–MJ31A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

— Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— Do Not Tear - Fold Here ----------

Cut Along Dotted Line

Reader's Comments

DEC GKS GKS$
Binding Reference Manual
AA–MJ31A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line