

DEC GKS Reference Manual Part 2

Order Number: AA-HW44D-TE

February 1990

This document is an encyclopedic reference to the DEC GKS level 2c run-time functions. This volume contains information on the DEC GKS inquiry functions, supported workstations, error messages, language-specific concerns, fonts, color representations, escapes, and GDPs.

Revision/Update Information: This revised document supersedes the *VAX GKS Reference Manual Volume II* (Order No. AA-HW44C-TE).

Operating System and Version: VMS Version 5.1 or higher. VAXstation requirement: VAXstation Windowing Software Versions 4.0 or higher, or DECwindows Version 1.0.

Software Version: DEC GKS Version 4.1

**digital equipment corporation
maynard, massachusetts**

First Printing, March 1984

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

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

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

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


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

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

All Rights Reserved.
Printed in U.S.A.

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 |
| DECmate | PDT | VAXcluster |
| DECnet | P/OS | VMS |
| DECsystem-10 | Professional | VT |
| DECSYSTEM-20 | Q-bus | Work Processor |
| DECUS | Rainbow | |
| DECwriter | RSTS | |
| DIBOL | RSX |  |

The following are third-party trademarks:

HP7475, HP7550, HP7580, and HP7585 are registered trademarks of Hewlett-Packard Company.

MPS2000 is a trademark of LaserGraphics, Inc.

PostScript is a registered trademark of Adobe Systems, Inc.

Tektronix is a registered trademark of Tektronix, Inc.

ZK5386

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

| | |
|---------------|------|
| Preface | xiii |
|---------------|------|

Chapter 11 Inquiry Functions

| | |
|---|-------------|
| 11.1 Using the Inquiry Functions | 11-3 |
| 11.1.1 The Error Status Argument | 11-5 |
| 11.1.2 The Value Type Argument | 11-6 |
| 11.2 Function Descriptions | 11-8 |
| GKS DESCRIPTION TABLE INQUIRIES | 11-9 |
| INQUIRE LEVEL OF GKS | 11-10 |
| INQUIRE LIST OF AVAILABLE WORKSTATION TYPES | 11-13 |
| INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER | 11-16 |
| INQUIRE WORKSTATION MAXIMUM NUMBERS | 11-18 |
| WORKSTATION DESCRIPTION TABLE INQUIRIES | 11-21 |
| INQUIRE COLOR FACILITIES | 11-22 |
| INQUIRE DEFAULT CHOICE DEVICE DATA | 11-26 |
| INQUIRE DEFAULT DEFERRAL STATE VALUES | 11-35 |
| INQUIRE DEFAULT LOCATOR DEVICE DATA | 11-38 |
| INQUIRE DEFAULT PICK DEVICE DATA | 11-45 |
| INQUIRE DEFAULT STRING DEVICE DATA | 11-51 |
| INQUIRE DEFAULT STROKE DEVICE DATA | 11-58 |
| INQUIRE DEFAULT VALUATOR DEVICE DATA | 11-65 |
| INQUIRE DISPLAY SPACE SIZE | 11-72 |
| INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES .. | 11-76 |
| INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES | 11-82 |
| INQUIRE FILL AREA FACILITIES | 11-88 |
| INQUIRE GENERALIZED DRAWING PRIMITIVE | 11-92 |

| | |
|---|--------|
| INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES | 11-96 |
| INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES | 11-99 |
| INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES | 11-102 |
| INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED | 11-106 |
| INQUIRE PATTERN FACILITIES | 11-109 |
| INQUIRE POLYLINE FACILITIES | 11-112 |
| INQUIRE POLYMARKER FACILITIES | 11-116 |
| INQUIRE PREDEFINED COLOR REPRESENTATION | 11-121 |
| INQUIRE PREDEFINED FILL AREA REPRESENTATION | 11-124 |
| INQUIRE PREDEFINED PATTERN REPRESENTATION | 11-128 |
| INQUIRE PREDEFINED POLYLINE REPRESENTATION | 11-132 |
| INQUIRE PREDEFINED POLYMARKER REPRESENTATION | 11-136 |
| INQUIRE PREDEFINED TEXT REPRESENTATION | 11-140 |
| INQUIRE TEXT FACILITIES | 11-145 |
| INQUIRE WORKSTATION CATEGORY | 11-150 |
| INQUIRE WORKSTATION CLASSIFICATION | 11-153 |
| GKS STATE LIST INQUIRIES | 11-156 |
| INQUIRE CLIPPING | 11-157 |
| INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES | 11-160 |
| INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER | 11-167 |
| INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES | 11-169 |
| INQUIRE INPUT QUEUE OVERFLOW | 11-175 |
| INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS | 11-178 |
| INQUIRE MORE SIMULTANEOUS EVENTS | 11-181 |
| INQUIRE NAME OF OPEN SEGMENT | 11-183 |
| INQUIRE NORMALIZATION TRANSFORMATION | 11-185 |
| INQUIRE OPERATING STATE VALUE | 11-188 |
| INQUIRE PICK IDENTIFIER VALUE | 11-190 |
| INQUIRE SET OF ACTIVE WORKSTATIONS | 11-192 |
| INQUIRE SET OF OPEN WORKSTATIONS | 11-195 |
| INQUIRE SET OF SEGMENT NAMES IN USE | 11-198 |
| WORKSTATION STATE LIST INQUIRIES | 11-201 |
| INQUIRE CHOICE DEVICE STATE | 11-202 |
| INQUIRE COLOR REPRESENTATION | 11-211 |
| INQUIRE FILL AREA REPRESENTATION | 11-215 |
| INQUIRE LIST OF COLOR INDICES | 11-219 |
| INQUIRE LIST OF FILL AREA INDICES | 11-222 |
| INQUIRE LIST OF PATTERN INDICES | 11-225 |
| INQUIRE LIST OF POLYLINE INDICES | 11-228 |
| INQUIRE LIST OF POLYMARKER INDICES | 11-231 |

| | |
|--|--------|
| INQUIRE LIST OF TEXT INDICES | 11-234 |
| INQUIRE LOCATOR DEVICE STATE | 11-237 |
| INQUIRE PATTERN REPRESENTATION | 11-245 |
| INQUIRE PICK DEVICE STATE | 11-249 |
| INQUIRE POLYLINE REPRESENTATION | 11-258 |
| INQUIRE POLYMARKER REPRESENTATION | 11-262 |
| INQUIRE SET OF SEGMENT NAMES ON WORKSTATION | 11-266 |
| INQUIRE STRING DEVICE STATE | 11-269 |
| INQUIRE STROKE DEVICE STATE | 11-276 |
| INQUIRE TEXT EXTENT | 11-286 |
| INQUIRE TEXT REPRESENTATION | 11-290 |
| INQUIRE VALUATOR DEVICE STATE | 11-295 |
| INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES | 11-302 |
| INQUIRE WORKSTATION CONNECTION AND TYPE | 11-307 |
| INQUIRE WORKSTATION STATE | 11-310 |
| INQUIRE WORKSTATION TRANSFORMATION | 11-313 |
| SEGMENT STATE LIST INQUIRIES | 11-317 |
| INQUIRE SEGMENT ATTRIBUTES | 11-318 |
| INQUIRE SET OF ASSOCIATED WORKSTATIONS | 11-322 |
| PIXEL INQUIRIES | 11-325 |
| INQUIRE PIXEL | 11-326 |
| INQUIRE PIXEL ARRAY | 11-329 |
| INQUIRE PIXEL ARRAY DIMENSIONS | 11-334 |

Appendix A DEC GKS-Supported Workstations

| | | |
|-------|---|-----|
| A.1 | Supported Workstation Types | A-1 |
| A.2 | Default Workstation Types | A-3 |
| A.3 | Output-Only Devices | A-4 |
| A.4 | Using Bit Masks for Workstation Types | A-4 |
| A.4.1 | An Alternative to Defining Bit Masks | A-5 |

Appendix B DEC GKS Constants

Appendix C DEC GKS Attribute Values

| | | |
|------------|--|------------|
| C.1 | Initial Polyline Attributes | C-1 |
| C.2 | Initial Polymarker Attributes | C-2 |
| C.3 | Initial Text Attributes | C-3 |
| C.4 | Initial Fill Area Attributes | C-4 |
| C.5 | Initial Segment Attributes | C-4 |
| C.6 | Initial Normalization Transformation Settings | C-5 |
| C.7 | DEC GKS-Specific Line Types | C-5 |
| C.8 | DEC GKS-Specific Marker Types | C-6 |

Appendix D DEC GKS Error Messages

| | | |
|-------------|---|-------------|
| D.1 | Implementation-Specific Errors | D-2 |
| D.2 | Operating State Errors | D-17 |
| D.3 | Workstation Errors | D-19 |
| D.4 | Transformation Function Errors | D-25 |
| D.5 | Output Attribute Errors | D-26 |
| D.6 | Output Function Errors | D-33 |
| D.7 | Segment Function Errors | D-34 |
| D.8 | Input Function Errors | D-36 |
| D.9 | Metafile Function Errors | D-39 |
| D.10 | Escape Function Errors | D-41 |
| D.11 | Miscellaneous Errors | D-41 |
| D.12 | System Errors | D-42 |

Appendix E DEC GKS Metafile Structures (GKSM, CGM)

| | | |
|------------|--|------------|
| E.1 | GKSM Metafiles | E-1 |
| E.1.1 | Data Format Information | E-2 |
| E.1.2 | GKSM Structure | E-2 |
| E.1.2.1 | Metafile Header Structure | E-3 |
| E.1.2.2 | Metafile Item Structure | E-4 |
| E.1.2.3 | Item Header Structure | E-4 |
| E.1.2.4 | Layout of Item Data Records | E-5 |
| E.1.3 | GKSM Physical File Organization | E-9 |
| E.2 | Computer Graphics Metafiles (CGM) | E-9 |
| E.2.1 | CGM Structure | E-11 |
| E.2.2 | Differences Between GKS and CGM | E-13 |
| E.2.3 | Character Encoding | E-14 |
| E.2.4 | Clear Text Encoding | E-16 |
| E.2.5 | CGM Element Descriptions | E-17 |
| E.2.5.1 | CGM Encoding Examples | E-25 |
| E.2.6 | CGM Physical File Organization | E-28 |

Appendix F Language-Specific Programming Information

| | | |
|------------|--|------------|
| F.1 | Passing Arguments by Descriptor | F-1 |
| F.2 | Programming in BASIC | F-3 |
| F.3 | Programming in VAX C | F-3 |
| F.4 | Programming in VAX COBOL | F-3 |
| F.5 | Programming in VAX Pascal | F-7 |

Appendix G DEC GKS Device-Independent Fonts

| | | |
|------------|---|-------------|
| G.1 | Font File Formats | G-1 |
| G.2 | Font Design | G-2 |
| G.3 | Stroke Font File | G-4 |
| G.3.1 | Stroke Font File Header | G-5 |
| G.3.2 | Character Descriptor | G-9 |
| | DEC GKS DEVICE-INDEPENDENT FONTS | G-13 |

Appendix H DEC GKS Color Chart

Appendix I DEC GKS GDPs and Escapes

| | |
|---|-------|
| DATA RECORD FORMAT USED IN THIS APPENDIX | I-2 |
| GENERALIZED DRAWING PRIMITIVES (GDPS) | I-4 |
| UNFILLED GDPS | I-9 |
| FILLED GDPS | I-26 |
| CELL ARRAY GDPS | I-44 |
| TEXT GDPS | I-46 |
| ESCAPE FUNCTIONS | I-48 |
| CONTROL ESCAPE FUNCTIONS | I-49 |
| OUTPUT, ATTRIBUTE, AND TRANSFORMATION ESCAPE FUNCTIONS | I-62 |
| DEC GKS DECWINDOWS ESCAPE FUNCTIONS | I-83 |
| DEC GKS STATE LIST INQUIRY ESCAPE FUNCTIONS | I-101 |
| WORKSTATION STATE LIST INQUIRY ESCAPE FUNCTIONS | I-106 |
| WORKSTATION DESCRIPTION TABLE INQUIRY ESCAPE FUNCTIONS | I-122 |
| UTILITY ESCAPE FUNCTIONS | I-132 |

Appendix J DEC GKS-Specific Input Values

| | |
|---|------|
| LOGICAL INPUT DEVICE NUMBERS | J-2 |
| CHOICE DEVICES | J-4 |
| LOCATOR DEVICES | J-7 |
| PICK DEVICES | J-8 |
| STRING DEVICES | J-9 |
| STROKE DEVICES | J-11 |
| VALUATOR DEVICES | J-12 |
| INPUT DEVICES AND ECHO AREA TITLES | J-13 |
| PROMPT AND ECHO TYPES, AND DATA RECORDS | J-15 |
| CHOICE INPUT CLASS | J-16 |
| LOCATOR INPUT CLASS | J-18 |
| PICK INPUT CLASS | J-25 |
| STRING INPUT CLASS | J-26 |
| STROKE INPUT CLASS | J-27 |
| VALUATOR INPUT CLASS | J-31 |
| KEYPAD FUNCTIONALITY | J-33 |
| CYCLING LOGICAL INPUT DEVICES | J-34 |
| NUMERIC KEYPAD (ZONING MECHANISM) | J-35 |
| NUMERIC KEYPAD (CHOICE) | J-36 |
| AUXILIARY KEYPAD (CHOICE) | J-38 |

Index

Examples

| | | |
|-------|---|--------|
| 11-1 | Set and Realized Inquiry Value Types | 11-7 |
| 11-2 | Determining the Default Choice Input Values | 11-32 |
| 11-3 | Determining the Default Locator Input Values | 11-43 |
| 11-4 | Determining the Default Pick Input Values | 11-49 |
| 11-5 | Determining the Default String Input Values | 11-56 |
| 11-6 | Determining the Default Stroke Input Values | 11-63 |
| 11-7 | Determining the Default Valuator Input Values | 11-70 |
| 11-8 | Determining the State of the Choice Logical Input Device | 11-208 |
| 11-9 | Determining the Current Locator State | 11-243 |
| 11-10 | Determining the Values Associated with the Current Pick State | 11-256 |
| 11-11 | Determining the Initial String Logical Input Device Values | 11-274 |
| 11-12 | Determining the Initial Stroke Logical Input Device Values | 11-283 |
| 11-13 | Determining the Current Valuator State | 11-300 |
| 11-14 | Determining the Dimensions of a Pixel Array | 11-337 |
| E-1 | CGM Metafile Creation | E-26 |
| F-1 | Macro Subroutine Used to Build Array Descriptors | F-4 |
| F-2 | A Sample COBOL Program Using the Subroutine BUILDESC | F-6 |
| G-1 | Printing the ASCII Values of Font Characters | G-10 |

Figures

| | | |
|-----|--|------|
| A-1 | Hexadecimal Bit Masks as Workstation Type Values | A-5 |
| E-1 | GKSM Metafile Structure | E-3 |
| E-2 | GKSM Metafile Header Structure | E-3 |
| E-3 | GKSM Metafile Item Structure | E-4 |
| E-4 | GKSM Metafile Item Header Structure | E-5 |
| E-5 | CGM Components | E-11 |
| E-6 | CGM Basic Data Encoding Format | E-15 |
| E-7 | CGM Basic Encoding Format for Real Numbers | E-16 |
| G-1 | DEC GKS Font Lines | G-2 |

| | | |
|--|-------------------------------|------|
| G-2 | DEC GKS Default Font Number 1 | |
| ISO Standard Character Set | | G-13 |
| G-3 | DEC GKS Font Number -2 | |
| Small Uniplex Simplex Roman and Greek | | G-14 |
| G-4 | DEC GKS Font Number -3 | |
| Large Simplex Uniplex Roman | | G-15 |
| G-5 | DEC GKS Font Number -4 | |
| Large Uniplex Simplex Greek | | G-16 |
| G-6 | DEC GKS Font Number -5 | |
| Large Simplex Uniplex Script | | G-17 |
| G-7 | DEC GKS Font Number -6 | |
| Medium Complex Duplex Roman | | G-18 |
| G-8 | DEC GKS Font Number -7 | |
| Medium Complex Duplex Greek | | G-19 |
| G-9 | DEC GKS Font Number -8 | |
| Medium Complex Duplex Italic | | G-20 |
| G-10 | DEC GKS Font Number -9 | |
| Large Complex Duplex Roman | | G-21 |
| G-11 | DEC GKS Font Number -10 | |
| Large Complex Duplex Greek | | G-22 |
| G-12 | DEC GKS Font Number -11 | |
| Large Complex Duplex Italic | | G-23 |
| G-13 | DEC GKS Font Number -12 | |
| Large Simplex Duplex Roman | | G-24 |
| G-14 | DEC GKS Font Number -13 | |
| Large Complex Duplex Script | | G-25 |
| G-15 | DEC GKS Font Number -14 | |
| Large Complex Duplex Cyrillic | | G-26 |
| G-16 | DEC GKS Font Number -15 | |
| Large Complex Triplex Roman | | G-27 |
| G-17 | DEC GKS Font Number -16 | |
| Large Complex Triplex Italic | | G-28 |
| G-18 | DEC GKS Font Number -17 | |
| Large Gothic Triplex German | | G-29 |
| G-19 | DEC GKS Font Number -18 | |
| Large Gothic Triplex English | | G-30 |
| G-20 | DEC GKS Font Number -19 | |
| Large Gothic Triplex Italian | | G-31 |
| G-21 | DEC GKS Font Number -20 | |
| Medium Complex Duplex Special Characters | | G-32 |
| G-22 | DEC GKS Font Number -21 | |

| | |
|---|------|
| Music, Astronomy, and Business | G-33 |
| G-23 DEC GKS Font Number -22 | |
| Large Uniplex Special Characters | G-34 |
| G-24 DEC GKS Font Number -23 | |
| Large Special Characters | G-35 |
| I-1 Using Vector Origin Points | I-7 |
| I-2 Forming an Ellipse | I-8 |

Tables

| | |
|--|------|
| A-1 DEC GKS-Supported Workstation Types | A-1 |
| B-1 GKS\$ Constants | B-2 |
| E-1 GKSM Metafile Header Fields | E-3 |
| E-2 GKSM Metafile Item Header Fields | E-5 |
| E-3 GKSM Metafile Data Record Fields | E-5 |
| E-4 CGM Element Descriptions | E-17 |
| F-1 Type Definitions | F-8 |
| H-1 DEC GKS Color Chart | H-2 |

Manual Objectives

This manual provides encyclopedic reference to the DEC Graphical Kernel System (GKS) and provides examples illustrating DEC GKS function calls. DEC GKS is a level 2c GKS implementation. For more information concerning GKS implementation levels, refer to Chapter 1, Introduction to DEC GKS.

Intended Audience

This manual is intended for experienced application programmers who need to reference information concerning the DEC GKS functions. Readers should be familiar with one high-level language and the DIGITAL Command Language (DCL). (For more information concerning DCL, refer to the *VMS DCL Dictionary*.)

Refer to the DEC GKS Binding Reference Manuals for information specific to the binding you use with DEC GKS. The available bindings for DEC GKS Version 4.1 are FORTRAN, C, and GKS\$. These manuals are designed for the experienced user of DEC GKS who needs to know the binding syntax and brief argument descriptions.

Although there are lengthy introductions at the beginning of each of the chapters, 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 two volumes. Part 1 contains the following information:

- Chapter 1, Introduction to DEC GKS, provides an introduction to the DEC GKS product and to the format of this reference manual.
- Chapter 2, Compiling, Linking, and Running DEC GKS Programs, provides information about DEC GKS and the VMS operating system.
- Chapter 3, Control Functions, provides information concerning the establishment of the DEC GKS and workstation environments.
- Chapter 4, Output Functions, provides information concerning the generation of output primitives.
- Chapter 5, Output Attribute Functions, provides information concerning the output attributes.
- Chapter 6, Transformation Functions, provides information concerning the normalization and workstation transformations.
- Chapter 7, Input Functions, provides information concerning input.
- Chapter 8, Segment Functions, provides information concerning the storage of output primitives in segments.
- Chapter 9, Metafile Functions, provides information concerning long-term storage of graphical images.
- Chapter 10, Error-Handling Functions, provides information concerning error-handling by the application program.

Part 2 of this manual contains the following information:

- Chapter 11, Inquiry Functions, provides information concerning the acquisition of DEC GKS and workstation status information.
- The appendixes, which include the following:
 - Appendix A, DEC GKS-Supported Workstations
 - Appendix B, DEC GKS Constants
 - Appendix C, DEC GKS Attribute Values
 - Appendix D, DEC GKS Error Messages
 - Appendix E, DEC GKS Metafile Structures (GKSM, CGM)
 - Appendix F, Language-Specific Programming Information
 - Appendix G, DEC GKS Device-Independent Fonts
 - Appendix H, DEC GKS Color Chart

- Appendix I, DEC GKS GDPs and Escapes
- Appendix J, DEC GKS-Specific Input Values

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 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.
- *DEC GKS Device Specifics Reference Manual*—For programmers who need information about specific devices.
- *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 Installation Guide*—For system managers who install DEC GKS software, including the Run-Time installation.

NOTE

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

```
$ HELP GKS RELEASE_NOTES RETURN
```

Conventions

| Convention | Meaning |
|---|---|
| <code>RETURN</code> | The symbol <code>RETURN</code> represents a single stroke of the RETURN key on a terminal. |
| <code>\$ RUN GKSPROG RETURN</code> | In interactive examples, the user's response to a prompt is printed in red; system prompts are printed in black. |
| <code>INTEGER X</code> . . . <code>X = 5</code> | A vertical ellipsis indicates that not all the text of a program or program output is illustrated. Only relevant material is shown in the example. |
| <code>option, ...</code> | 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. |
| <code>[output-source, ...]</code> | Square brackets, in function synopses and a few other contexts, indicate that a syntactic element is optional. |
| <i>deferral mode</i> | All names of the DEC GKS description table and state list entries, and of the workstation description table and state list entries, are italicized. Also, argument names are italicized when they are referenced in the text. |
| <code>GKS\$K_LINETYPE_DASHED</code> | In the text, constants and function names are in uppercase letters. |

Inquiry Functions

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

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

| Table/List | Description |
|-----------------------|--|
| GKS Description Table | <p>This table contains constant information about the DEC GKS implementation you are using, such as the level of GKS (with DEC GKS, level 2c), the number of available workstation types, the list of workstation types, the maximum allowable open workstations, and so forth.</p> <p>If you are transporting your programs from one implementation of GKS to another, you may need to inquire about the implementation level of GKS on a given system, so that your program does not call unsupported functions.</p> |

| Table/List | Description |
|-------------------------------|---|
| Workstation Description Table | <p>This type of table contains constant information about one particular workstation, such as the workstation type, the workstation category, the device-specific maximum coordinate values, the different bundled output attribute values, and so forth. Each graphics handler contains a workstation description table describing that particular device.</p> <p>If your DEC GKS application uses more than one workstation at a time, or if you are unsure of the capabilities of your workstation, you may need to inquire about the values contained in the workstation description table.</p> |
| GKS State List | <p>This list contains entries that specify the current DEC GKS values such as the set of open workstations (if any), the current normalization transformation number, the current character height, and so forth.</p> <p>If you need to check the alterable DEC GKS values, you may need to inquire about the values contained in the DEC GKS state list.</p> |
| Workstation State List | <p>For each workstation you open, DEC GKS allocates space for a workstation state list. This list contains entries that specify whether output is "on hold" (deferred), whether or not the surface has to be redrawn to fulfill an output request, whether the workstation surface is "empty" by GKS definition, whether the picture on the surface represents all the requests for output made thus far by the application program, and so forth.</p> <p>If you need information concerning the current state of a particular workstation, you may need to inquire about the values contained in the workstation state list.</p> |
| Segment State List | <p>When you create a segment, DEC GKS creates a segment state list. The segment state list contains entries that specify the segment name, the set of associated workstations, the detectability of the segment, and so forth.</p> <p>If you need information concerning a particular segment, you may need to inquire about the values contained in the segment state list.</p> |

NOTE

You cannot inquire from the VAXstation workstation description table unless you are logged onto a MicroVAX running DEC GKS.

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

Calling the inquiry functions is simple. Consequently, only the INQUIRE DEFAULT DEVICE DATA and the INQUIRE DEVICE STATE function descriptions contain program examples. For complete examples that use calls to these input inquiry functions, refer to Chapter 7, Input Functions.

To gain an understanding of knowing when to call certain DEC GKS inquiry functions, refer to the *DEC GKS User Manual*. For more information concerning the state lists and description tables, refer to Chapter 3, Control Functions.

11.1 Using the Inquiry Functions

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

```
GKSS$INQ_LEVEL ( error_status, gks_level )
```

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

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

```
GKS$INQ_LOCATOR_STATE ( workstation_id, device_type,  
                        value_type, error_status, operating_operating_mode,  
                        echo_flag, transformation_number, world_x_value, world_y_value,  
                        prompt_and_echo_type, echo_area, data_record, record_buffer_length,  
                        record_size )
```

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

The argument *record_buffer_length* is a modifiable argument unique to the INQUIRE DEFAULT DEVICE DATA and INQUIRE DEVICE STATE functions. On input, the argument must contain the size of the data record buffer you declare. On output, the graphics handler writes the amount of the buffer filled with data. If on output the argument *record_size* is larger than the argument *record_buffer_size*, you know that the graphics handler truncated the input data record when writing to the buffer and data was lost.

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

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

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

For a better understanding of this process, review the following code example.


```

      .
      .
      .
      INTEGER WS_ID, DATA_RECORD( 1 ), PROMPT_ECHO_TYPE,
* ERROR_INDICATOR, INPUT_MODE, ECHO_FLAG, TRANSFRM_NUMBER,
* RECORD_BUFFER_LENGTH, RECORD_SIZE, INPUT_STATUS, DEVICE_NUM
      REAL ECHO_AREA( 4 ), WORLD_X, WORLD_Y
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /
      .
      .
C     Let the graphics handler know how large the data record buffer is...
      RECORD_BUFFER_LENGTH = 4
C     Initialize variables by passing them to the inquiry function.
      CALL GKSSINQ_LOCATOR_STATE( WS_ID, DEVICE_NUM,
* GKSSK_VALUE_REALIZED, ERROR_INDICATOR, INPUT_MODE,
* ECHO_FLAG, TRANSFRM_NUMBER, WORLD_X, WORLD_Y,
* PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
* RECORD_BUFFER_LENGTH, RECORD_SIZE )
C     Change only one variable value.
      PROMPT_ECHO_TYPE = 1

C     Initialize the logical input device with the necessary variable
C     values.
      CALL GKSSINIT_LOCATOR( WS_ID, DEVICE_NUM, WORLD_X,
* WORLD_Y, TRANSFRM_NUMBER, PROMPT_ECHO_TYPE, ECHO_AREA,
* DATA_RECORD, RECORD_BUFFER_LENGTH )

C     Request input from the device.
      CALL GKSSREQUEST_LOCATOR( WS_ID, DEVICE_NUM, INPUT_STATUS,
* TRANSFRM_NUMBER, WORLD_X, WORLD_Y )
      .
      .
      .

```

For more information concerning the workstation identifier, refer to Chapter 3, Control Functions. For more information concerning the input device type or general input concepts, refer to Chapter 7, Input Functions.

11.1.1 The Error Status Argument

DEC GKS inquiry functions never generate an error, but they can encounter error conditions. For all inquiry functions, the first write-only argument within the argument list is always the *error status* argument. The value passed to this argument determines whether the values passed to the remaining write-only arguments are valid.

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

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

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

For more information concerning the DEC GKS error messages and their numbers, refer to Appendix D, DEC GKS Error Messages. For more information concerning DEC GKS error handling, refer to Chapter 10, Error-Handling Functions.

11.1.2 The Value Type Argument

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

The possible value types are as follows:

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

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

you can inquire about both types of values. Example 11-1 illustrates this process on a VAXstation.

Example 11-1: Set and Realized Inquiry Value Types

```
C      This program writes set and realized pick aperture sizes to the
C      workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, INITIAL_STATUS, SEGMENT, PICK_ID,
*      PROMPT_ECHO_TYPE, ERROR_STATUS, INPUT_MODE, ECHO_FLAG,
*      DATA_LENGTH, RETURN_SIZE, INPUT_STATUS, DEVICE_NUM
      REAL ECHO_AREA( 4 ), DATA_RECORD_SET( 1 ),
*      DATA_RECORD_REALIZED
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, 0 )
      CALL GKS$ACTIVATE_WS( WS_ID )

C      Inquire default values so that you can initialize the device.
      DATA_LENGTH = 4
      CALL GKS$INQ_PICK_STATE( WS_ID, DEVICE_NUM,
*      GKS$K_VALUE_SET, ERROR_STATUS, INPUT_MODE, ECHO_FLAG,
*      INITIAL_STATUS, SEGMENT, PICK_ID, PROMPT_ECHO_TYPE,
*      ECHO_AREA, DATA_RECORD_SET, DATA_LENGTH, RETURN_SIZE )

C      Set the aperture size to be 0.1 in device coordinates.
      DATA_RECORD_SET( 1 ) = 0.001

C      Initialize the device with the new aperture size.
      CALL GKS$INIT_PICK( WS_ID, DEVICE_NUM, INITIAL_STATUS,
*      SEGMENT, PICK_ID, PROMPT_ECHO_TYPE, ECHO_AREA,
*      DATA_RECORD_SET, DATA_LENGTH, RETURN_SIZE )

C      Obtain the set value...
      DATA_LENGTH = 4          ! One longword for aperture size.
      CALL GKS$INQ_PICK_STATE( WS_ID, DEVICE_NUM,
*      GKS$K_VALUE_SET, ERROR_STATUS, INPUT_MODE, ECHO_FLAG,
*      INITIAL_STATUS, SEGMENT, PICK_ID, PROMPT_ECHO_TYPE,
*      ECHO_AREA, DATA_RECORD_SET, DATA_LENGTH, RETURN_SIZE )

C      Obtain the realized value...
      DATA_LENGTH = 4          ! One longword for aperture size.
      CALL GKS$INQ_PICK_STATE( WS_ID, DEVICE_NUM,
*      GKS$K_VALUE_REALIZED, ERROR_STATUS, INPUT_MODE, ECHO_FLAG,
*      INITIAL_STATUS, SEGMENT, PICK_ID, PROMPT_ECHO_TYPE,
*      ECHO_AREA, DATA_RECORD_REALIZED, DATA_LENGTH, RETURN_SIZE )
```

(continued on next page)

Example 11-1 (Cont.): Set and Realized Inquiry Value Types

```
WRITE(6,*) 'Set value:', DATA_RECORD_SET
WRITE(6,*) 'Realized value:', DATA_RECORD_REALIZED

CALL GKSS$DEACTIVATE_WS( WS_ID )
CALL GKSS$CLOSE_WS( WS_ID )
CALL GKSS$CLOSE_GKS()
END
```

You see the following when you compile, link, and execute this program:

```
$ FORTRAN EXAMPLE_1 RETURN
$ LINK EXAMPLE_1 RETURN
$ RUN EXAMPLE_1 RETURN
Set value: 1.0000000E-03
Realized value: 4.2635733E-03
$
```

For more information concerning pick input, refer to Chapter 7, Input Functions.

11.2 Function Descriptions

This section describes the DEC GKS inquiry functions in detail. The functions are organized alphabetically and by the type of inquiry they perform: DEC GKS description table, workstation description table, DEC GKS state list, workstation state list, segment state list, and pixel inquiries.

GKS Description Table Inquiries

This section describes the DEC GKS description table inquiries. You use these functions if you are not sure which implementation of DEC GKS you are using.

GKS Description Table Inquiries

INQUIRE LEVEL OF GKS

INQUIRE LEVEL OF GKS

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The implementation level is available when DEC GKS is in any operating state except GKS\$K_GKCL. If the state is GKS\$K_GKCL, the output argument is undefined. The function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

GKS\$INQ_LEVEL (*error_status*, *gks_level*)

GQLVKS (*error_status*, *level*)

ginqllevelgks (*level*, *error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS Description Table Inquiries INQUIRE LEVEL OF GKS

gks_level

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is the level of the GKS implementation you are using (with DEC GKS, level 2c). The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|-----------------|--|
| -3 | GKS\$K_LEVEL_MA | Minimal output, no input |
| -2 | GKS\$K_LEVEL_MB | Minimal output, request input |
| -1 | GKS\$K_LEVEL_MC | Minimal output, full input |
| 0 | GKS\$K_LEVEL_0A | All primitives and attributes, no input |
| 1 | GKS\$K_LEVEL_0B | All primitives and attributes, request input |
| 2 | GKS\$K_LEVEL_0C | All primitives and attributes, full input |
| 3 | GKS\$K_LEVEL_1A | Basic segmentation with full output, no input |
| 4 | GKS\$K_LEVEL_1B | Basic segmentation with full output, request input |
| 5 | GKS\$K_LEVEL_1C | Basic segmentation with full output, full input |
| 6 | GKS\$K_LEVEL_2A | Workstation independent and segment storage, no input |
| 7 | GKS\$K_LEVEL_2B | Workstation independent and segment storage, request input |
| 8 | GKS\$K_LEVEL_2C | Workstation independent and segment storage, full input |

GKS Description Table Inquiries

INQUIRE LEVEL OF GKS

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS Description Table Inquiries

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE LIST OF AVAILABLE WORKSTATION TYPES returns a list of the supported workstation types.

The list of supported workstations is available when DEC GKS is in any operating state except GKS\$K_GKCL. If the state is GKS\$K_GKCL, the output argument is undefined. The function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

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

GQEWK (*element, error_status, num_types, relement*)

ginqavailwstypes (*bufsize, start, wstypes, actual_types, error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

GKS Description Table Inquiries

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_workstation_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of different workstation types.

workstation_type_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array that contains the integers representing the various supported workstations. For a list of the DEC GKS-supported workstation types, refer to Appendix A, DEC GKS-Supported Workstations.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the actual number of workstation types passed back to the array. You can use this value to determine whether you defined an array large enough to hold all the returned values.

GKS Description Table Inquiries INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS Description Table Inquiries

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The maximum normalization transformation number is available when DEC GKS is in any operating state except GKS\$K_GKCL. If the state is GKS\$K_GKCL, the output argument is undefined. The function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

GKS\$MAX_XFORM (*error_status*, *max_transformation*)

GQMNTN (*error_status*, *max*)

ginqmaxntrannum (*maxtran*, *error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function

GKS Description Table Inquiries

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

max_transformation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the maximum normalization transformation number supported by the GKS implementation. You can associate window and viewport boundaries to transformation numbers 1 through *max_transformations*.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS Description Table Inquiries

INQUIRE WORKSTATION MAXIMUM NUMBERS

INQUIRE WORKSTATION MAXIMUM NUMBERS

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE WORKSTATION MAXIMUM NUMBERS returns the maximum number of open workstations, active workstations, and the maximum number of workstations that can be associated with a segment.

The maximum number of types of workstations is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQWKM (*error_status*, *sim_open*, *sim_active*, *ws_w_seg*)

ginqwsmaxnum (*maxws*, *error_status*)

GKS Description Table Inquiries

INQUIRE WORKSTATION MAXIMUM NUMBERS

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

max_open_workstations *max_active_workstations*

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the maximum number of open and active workstations supported by the implementation of GKS.

max_ws_with_segment

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the maximum number of workstations that the GKS implementation can associate with a segment.

GKS Description Table Inquiries

INQUIRE WORKSTATION MAXIMUM NUMBERS

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

Workstation Description Table Inquiries

Workstation Description Table Inquiries

This section describes the workstation description table inquiries. (For more information concerning the workstation description table, refer to Chapter 3, Control Functions.) After you have determined on what type of workstation you are working, you use these functions to determine the workstation's capabilities and limits.

Workstation Description Table Inquiries

INQUIRE COLOR FACILITIES

INQUIRE COLOR FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE COLOR FACILITIES returns the number of color indexes, the number of available colors, and the color capabilities of a specified workstation.

The color facilities are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined. The function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

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

GQCF (*workstation_type, error_status, ncolors, color_flag, nindexes*)

ginqcolourfacil (*workstation_type, bufsize, fac_size, fac, error_status*)

Workstation Description Table Inquiries

INQUIRE COLOR FACILITIES

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_colors

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of colors supported by the workstation. If the function returns a value of 0 to this argument, a continuous range of colors is available.

color_or_mono

data type: **integer**
access: **write-only**
mechanism: **by reference**

Workstation Description Table Inquiries

INQUIRE COLOR FACILITIES

This argument is a flag specifying whether color is available on the specified workstation. The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|-------------------|-------------------|
| 0 | GKS\$K_MONOCHROME | Monochrome device |
| 1 | GKS\$K_COLOR | Color device |

num_color_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of predefined color indexes available for the specified workstation.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |

Workstation Description Table Inquiries INQUIRE COLOR FACILITIES

| Error Number | Completion Status Code | Message |
|-------------------------|-------------------------------|---|
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of cate- gory OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

INQUIRE DEFAULT CHOICE DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the choice input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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,

Workstation Description Table Inquiries INQUIRE DEFAULT CHOICE DEVICE DATA

record_buffer_length,
record_size)

GQDCH (*workstation_type, device_number, element, dim_dr,*
error_status, num_choi, num_types, relement,
echo_area, len_dr, dr)

ginqdefchoice (*workstation_type, device_number, bufsize,*
data_size, data, error_status)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

max_choices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the maximum number of supported choices.

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of choice prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **modifiable**
mechanism: **by reference**

This argument is a pointer to the input data record.

INQUIRE DEFAULT CHOICE DEVICE DATA returns a different amount of information depending on the value contained in the first component of the data record. If you pass the value 0 as this argument and the value 4 as the *record_buffer_length* argument, then this function only returns the default

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

number of choices (it ignores the rest of the write-only arguments). This functionality allows you to check to see if your declared string buffers are large enough to hold all the default strings.

Once you obtain the default number of choices, you must initialize the arrays containing string sizes, string addresses, and strings, and then call **INQUIRE DEFAULT CHOICE DEVICE DATA** a second time. In the second call, pass the number of choices obtained in the first call to **INQUIRE DEFAULT CHOICE DEVICE DATA**, and pass the *record_buffer_length* value that specifies the whole data record. Then the function writes all the default values to its write-only arguments.

To understand the process of calling **INQUIRE DEFAULT CHOICE DEVICE DATA** twice, refer to the program example in this function description.

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you have defined an array large enough to hold all available values.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on the workstation in routine **** |

Workstation Description Table Inquiries INQUIRE DEFAULT CHOICE DEVICE DATA

Program Example

Example 11-2 illustrates the use of the function INQUIRE DEFAULT CHOICE DEVICE DATA.

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

Example 11-2: Determining the Default Choice Input Values

```
C   This program writes the return values of the function
C   GKSSINQ_DEF_CHOICE_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 3 ), NUM_CHOICES,
* LIST_PROMPT_TYPES( 10 ), NUM_PROMPT_ECHO, ERROR_STATUS,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE,
* STRING_SIZES( 10 ), STRING_POINTERS( 10 ), DEVICE_NUM,
* I, MAX_CHOICES

      CHARACTER*80 STRINGS(10)

      REAL ECHO_AREA( 4 )
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

C   First element in the data record is the number of choices.
      EQUIVALENCE( DATA_RECORD(1), NUM_CHOICES )

      CALL GKSSOPEN_GKS( 'SYS$ERROR:' )

C   Initialize the first data record component to 0. This forces
C   GKSSINQ_DEF_CHOICE_DATA to return only the number of default
C   choices.
      NUM_CHOICES = 0

C   Tell the handler the size of the record buffer (do not include
C   the array addresses in this call).
      RECORD_BUFFER_LENGTH = 4

C   Call the function to find the number of default choices.
      CALL GKSSINQ_DEF_CHOICE_DATA( GKSSK_VT240, DEVICE_NUM,
* ERROR_STATUS, MAX_CHOICES, NUM_PROMPT_ECHO,
* %DESCR( LIST_PROMPT_TYPES), ECHO_AREA, DATA_RECORD,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
* RECORD_SIZE )

C   Initialize the string pointers...
      DO 100 I = 1, NUM_CHOICES
          STRING_POINTERS( I ) = %LOC( STRINGS(I) )
          STRING_SIZES( I ) = 80
100  CONTINUE
```

(continued on next page)

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

Example 11-2 (Cont.): Determining the Default Choice Input Values

```
C      Initialize the rest of the data record...
DATA_RECORD( 2 ) = %LOC( STRING_SIZES )
DATA_RECORD( 3 ) = %LOC( STRING_POINTERS )

C      Initialize the modifiable argument (this time, you pass the
C      array addresses)...
RECORD_BUFFER_LENGTH = 12

C      You can obtain this information as long as GKS is open.
CALL GKSS$INO_DEF_CHOICE_DATA( GKSS$K_VT240, DEVICE_NUM,
* ERROR_STATUS, MAX_CHOICES, NUM_PROMPT_ECHO,
* %DESCR( LIST_PROMPT_TYPES), ECHO_AREA, DATA_RECORD,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
* RECORD_SIZE )

C      Write the returned values to the screen.
WRITE(6, *) 'The error status: ', ERROR_STATUS
WRITE(6, *) 'The maximum number of choices: ', MAX_CHOICES
WRITE(6, *) 'The number of prompt/echo types: ', NUM_PROMPT_ECHO
WRITE(6, *) 'The list of prompt/echo types: ', LIST_PROMPT_TYPES
WRITE(6, *) 'The echo area: ', ECHO_AREA
WRITE(6, *) 'The choice data record: ', DATA_RECORD
WRITE(6, *) 'The prompt/echo list return size:', PROMPT_RETURN_SIZE
WRITE(6, *) 'The data record buffer size: ',
* RECORD_BUFFER_LENGTH
WRITE(6, *) 'The data record size: ', RECORD_SIZE

C      STRINGS holds the default choice strings...
WRITE(6,*) 'The default choice strings are as follows:'
DO 200 i = 1, NUM_CHOICES
    WRITE(6,*) STRINGS(I)
200 CONTINUE

CALL GKSS$CLOSE_GKS()
END
```

Workstation Description Table Inquiries

INQUIRE DEFAULT CHOICE DEVICE DATA

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_2 RETURN
$ LINK     EXAMPLE_2 RETURN
$ RUN      EXAMPLE_2 RETURN
The error status: 0
The maximum number of choices: 47
The number of prompt/echo types: 3
The list of prompt/echo types: 1 3 --1 0
0 0 0 0 0 0
The echo area: 533.0000 799.0000 0.0000000E+00 479.0000
The choice data record: 5 1076 1116
The prompt/echo list return size: 3
The data record buffer size: 12
The data record size: 0
The default choice strings are as follows:
CHOICE1
CHOICE2
CHOICE3
CHOICE4
CHOICE5
$
```

To review the functionality of INQUIRE DEFAULT CHOICE DEVICE DATA within a larger program, refer to the choice input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT DEFERRAL STATE VALUES

INQUIRE DEFAULT DEFERRAL STATE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default deferral and regeneration modes are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning deferral, implicit regeneration, or operating states, refer to Chapter 3, Control Functions.

Syntax

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

GQDDS (*workstation_type*, *error_status*, *def_mode*, *reg_mode*)

ginqdefdeferst (*workstation_type*, *def*, *error_status*)

Workstation Description Table Inquiries

INQUIRE DEFAULT DEFERRAL STATE VALUES

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

deferral_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the default deferral mode. The argument can be any of the following values or constants:

| Value | Constant | Description |
|--------------|-----------------|--|
| 0 | GKS\$K_ASAP | Generate images as soon as possible. |
| 1 | GKS\$K_BNIG | Generate images before input is requested globally. |
| 2 | GKS\$K_BNIL | Generate images before input is requested locally. |
| 3 | GKS\$K_ASTI | Generate images some time. Exact time is not guaranteed. |

Workstation Description Table Inquiries

INQUIRE DEFAULT DEFERRAL STATE VALUES

regeneration_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the default implicit regeneration mode. The argument can be any of the following values or constants.

| Value | Constant | Description |
|--------------|-----------------------|-----------------------------------|
| 0 | GKS\$K_IRG_SUPPRESSED | Image regeneration is suppressed. |
| 1 | GKS\$K_IRG_ALLOWED | Image regeneration is allowed. |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

INQUIRE DEFAULT LOCATOR DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the locator input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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,

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

record_buffer_length,
record_size)

GQDLC (*workstation_type, device_number, element, dim_dr,*
error_status, px, py, num_types, relement, echo_area,
len_dr, dr)

ginqdefloc (*workstation_type, device_number, bufsize,*
data_size, data, error_status)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

initial_world_x
initial_world_y

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments comprise the initial starting position of the locator prompt, in world coordinates. For information concerning the DEC GKS coordinate system, refer to Chapter 6, Transformation Functions.

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of locator prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available locator prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the default locator input data record for the specified device.

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you have defined an array large enough to hold all available values.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$_ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on the workstation in routine **** |

Program Example

Example 11-3 illustrates the use of the function, INQUIRE DEFAULT LOCATOR DEVICE DATA.

Workstation Description Table Inquiries

INQUIRE DEFAULT LOCATOR DEVICE DATA

Example 11-3: Determining the Default Locator Input Values

```
C      This program writes the return values of the function
C      GKS$INQ_DEF_LOCATOR_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 1 ), DEVICE_NUM,
* LIST_PROMPT_TYPES( 7 ), NUM_PROMPT_ECHO, ERROR_STATUS,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE
      REAL ECHO_AREA( 4 ), INIT_WORLD_X, INIT_WORLD_Y
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )

C      You need to initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 4

C      You can obtain this information as long as GKS is open.
      CALL GKS$INQ_DEF_LOCATOR_DATA( GKS$K_VT240, DEVICE_NUM,
* ERROR_STATUS, INIT_WORLD_X, INIT_WORLD_Y, NUM_PROMPT_ECHO,
* %DESCR( LIST_PROMPT_TYPES ), ECHO_AREA, DATA_RECORD,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE )

C      Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The initial X value: ', INIT_WORLD_X
      WRITE(6, *) 'The initial Y value: ', INIT_WORLD_Y
      WRITE(6, *) 'The number of prompt/echo types: ', NUM_PROMPT_ECHO
      WRITE(6, *) 'The list of prompt/echo types: ', LIST_PROMPT_TYPES
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The locator data record: ', DATA_RECORD
      WRITE(6, *) 'The prompt/echo list return size: ',
* PROMPT_RETURN_SIZE
      WRITE(6, *) 'The data record buffer size: ',
* RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The data record size: ', RECORD_SIZE
      CALL GKS$CLOSE_GKS()
      END
```

Workstation Description Table Inquiries INQUIRE DEFAULT LOCATOR DEVICE DATA

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_3 RETURN
$ LINK     EXAMPLE_3 RETURN
$ RUN      EXAMPLE_3 RETURN
The error status:          0
The initial X value:      0.5000000
The initial Y value:      0.5000000
The number of prompt/echo types:          7
The list of prompt/echo types:      1          2          3          4
                                     5          6          --1
The echo area:  0.0000000E+00  479.0000  0.0000000E+00  479.0000
The locator data record:          0
The prompt/echo list return size:          7
The data record buffer size:          0
The data record size:          0
$
```

To review the functionality of INQUIRE DEFAULT LOCATOR DEVICE DATA within a larger program, refer to the locator input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT PICK DEVICE DATA

INQUIRE DEFAULT PICK DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the pick input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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*)

Workstation Description Table Inquiries

INQUIRE DEFAULT PICK DEVICE DATA

GQDPK (*workstation_type, device_number, element, dim_dr, error_status, num_types, relement, echo_area, len_dr, dr*)

ginqdefpick (*workstation_type, device_number, bufsize, data_size, data, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT PICK DEVICE DATA

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of pick prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available pick prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the default pick input data record for the specified device.

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you have defined an array large enough to hold all available values.

Workstation Description Table Inquiries

INQUIRE DEFAULT PICK DEVICE DATA

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

Workstation Description Table Inquiries INQUIRE DEFAULT PICK DEVICE DATA

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$_ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on the workstation in routine **** |

Program Example

Example 11-4 illustrates the use of the function INQUIRE DEFAULT PICK DEVICE DATA.

Example 11-4: Determining the Default Pick Input Values

```
C      This program writes the return values of the function
C      GKS$INQ_DEF_PICK_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DEVICE_NUM,
* PROMPT_ECHO TYPE( 5 ), NUM_PROMPT_ECHO,
* ERROR_STATUS, PROMPT_RETURN_SIZE,
* RECORD_BUFFER_LENGTH, RECORD_SIZE
      REAL ECHO_AREA( 4 ), DATA_RECORD( 1 )
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /
      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )

C      Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 4
```

(continued on next page)

Workstation Description Table Inquiries

INQUIRE DEFAULT PICK DEVICE DATA

Example 11-4 (Cont.): Determining the Default Pick Input Values

```

C   You can obtain this information as long as GKS is open.
    CALL GKS$INQ_DEF_PICK_DATA( GKS$K_VT240, DEVICE_NUM,
*   ERROR_STATUS, NUM_PROMPT_ECHO,
*   %DESCR( PROMPT_ECHO_TYPE ), ECHO_AREA, DATA_RECORD,
*   PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
*   RECORD_SIZE )

C   Write the returned values to the screen.
    WRITE(6, *) 'The error status: ', ERROR_STATUS
    WRITE(6, *) 'The number of prompt/echo types: ',
*   NUM_PROMPT_ECHO
    WRITE(6, *) 'The prompt/echo types: ',
*   PROMPT_ECHO_TYPE
    WRITE(6, *) 'The echo area: ', ECHO_AREA
    WRITE(6, *) 'The pick data record: ', DATA_RECORD
    WRITE(6, *) 'The prompt/echo list return size: ',
*   PROMPT_RETURN_SIZE
    WRITE(6, *) 'The data record buffer size: ',
*   RECORD_BUFFER_LENGTH
    WRITE(6, *) 'The data record size: ', RECORD_SIZE
    CALL GKS$CLOSE_GKS()
    END

```

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```

$ FORTRAN EXAMPLE_4 RETURN
$ LINK     EXAMPLE_4 RETURN
$ RUN     EXAMPLE_4 RETURN
The error status:                0
The number of prompt/echo types:                0
The prompt/echo types:          1          2          3          0
                                0
The echo area:  0.0000000E+00  479.0000          0.0000000E+00  479.0000
The pick data record:          4.790000
The prompt/echo list return size:                3
The data record buffer size:                4
The data record size:                4
$

```

To review the functionality of INQUIRE DEFAULT PICK DEVICE DATA within a larger program, refer to the pick input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

INQUIRE DEFAULT STRING DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the string input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

Syntax

GKS\$INQ_DEF_STRING_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)

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

GQDST (*workstation_type, device_number, element, dim_dr, error_status, max_buf, num_types, relement, echo_area, len_buf, len_dr, dr*)

ginqdefstring (*workstation_type, device_number, bufsize, data_size, data, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

buffer_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the maximum allowable size of the buffer, in bytes, that eventually determines the size of the input string.

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of string prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available string prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the default string input data record for the specified device.

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you had defined an array large enough to hold all available values.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$_ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on the workstation in routine **** |

Program Example

Example 11-5 illustrates the use of the function INQUIRE DEFAULT STRING DEVICE DATA.

Workstation Description Table Inquiries

INQUIRE DEFAULT STRING DEVICE DATA

Example 11-5: Determining the Default String Input Values

```
C      This program writes the return values of the function
C      GKS$INQ_DEF_STRING_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 2 ), DEVICE_NUM,
* LIST_PROMPT_TYPES( 5 ), NUM_PROMPT_ECHO, ERROR_STATUS,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE,
* BUFFER_LENGTH, CUR_POSITION
      REAL ECHO_AREA( 4 )
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      EQUIVALENCE( DATA_RECORD( 1 ), BUFFER_LENGTH )
      EQUIVALENCE( DATA_RECORD( 2 ), CUR_POSITION )

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )

C      Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 8

C      You can obtain this information as long as GKS is open.
      CALL GKS$INQ_DEF_STRING_DATA( GKS$K_VT240, DEVICE_NUM,
* ERROR_STATUS, BUFFER_LENGTH, NUM_PROMPT_ECHO,
* %DESCR( LIST_PROMPT_TYPES ), ECHO_AREA, DATA_RECORD,
* PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
* RECORD_SIZE )

C      Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The string buffer size: ', BUFFER_LENGTH
      WRITE(6, *) 'The number of prompt/echo types: ',
* NUM_PROMPT_ECHO
      WRITE(6, *) 'The list of prompt/echo types: ',
* LIST_PROMPT_TYPES
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The string data record: ', DATA_RECORD
      WRITE(6, *) 'The prompt/echo list return size: ',
* PROMPT_RETURN_SIZE
      WRITE(6, *) 'The data record buffer size: ',
* RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The data record size: ',
* RECORD_SIZE
      CALL GKS$CLOSE_GKS()
      END
```

Workstation Description Table Inquiries INQUIRE DEFAULT STRING DEVICE DATA

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_5 RETURN
$ LINK     EXAMPLE_5 RETURN
$ RUN      EXAMPLE_5 RETURN
The error status:          0
The string buffer size:   20
The number of prompt/echo types:      3
The list of prompt/echo types:  1      2      3      0
                                0
The echo area:    533.0000      799.0000      0.0000000E+00      479.0000
The string data record:      20      0
The prompt/echo list return size:      1
The data record buffer size:      8
The data record size:      8
$
```

To review the functionality of INQUIRE DEFAULT STRING DEVICE DATA within a larger program, refer to the string input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

INQUIRE DEFAULT STROKE DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the stroke input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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,

Workstation Description Table Inquiries INQUIRE DEFAULT STROKE DEVICE DATA

*record_buffer_length,
record_size)*

GQDSK (*workstation_type, device_number, element, dim_dr,
error_status, max_buf, num_types, relement, echo_area,
len_buf, len_dr, dr*)

ginqdefstroke (*workstation_type, device_number, bufsize,
data_size, data, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

buffer_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the maximum allowable size of the buffer, in bytes, that determines the maximum number of points accepted as part of the stroke. The buffer holds one point per byte.

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of stroke prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available stroke prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the default stroke input record for the specified device.

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you had defined an array large enough to hold all available values.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$_ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on the workstation in routine **** |

Program Example

Example 11-6 illustrates the use of the function INQUIRE DEFAULT STROKE DEVICE DATA.

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

Example 11-6: Determining the Default Stroke Input Values

```
C      This program writes the return values of the function
C      GK$INQ_DEF_STROKE_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 13 ), DEVICE_NUM,
*      LIST_PROMPT_TYPES( 5 ), NUM_PROMPT_ECHO, ERROR_STATUS,
*      PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
*      RECORD_SIZE, BUFFER_LENGTH
      REAL ECHO_AREA( 4 )
      DATA WS_ID / 1 //, DEVICE_NUM / 1 /

      EQUIVALENCE( DATA_RECORD( 1 ), BUFFER_LENGTH )

      CALL GK$OPEN_GKS( 'SYS$ERROR:' )

C      Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 52

C      You can obtain this information as long as GKS is open.
      CALL GK$INQ_DEF_STROKE_DATA( GK$K_VT240, DEVICE_NUM,
*      ERROR_STATUS, BUFFER_LENGTH, NUM_PROMPT_ECHO,
*      %DESCR( LIST_PROMPT_TYPES ), ECHO_AREA, DATA_RECORD,
*      PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH,
*      RECORD_SIZE )

C      Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The stroke buffer size: ', BUFFER_LENGTH
      WRITE(6, *) 'The number of prompt/echo types: ',
*      NUM_PROMPT_ECHO
      WRITE(6, *) 'The list of prompt/echo types: ',
*      LIST_PROMPT_TYPES
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The stroke data record: ', DATA_RECORD
      WRITE(6, *) 'The prompt/echo list return size: ',
*      PROMPT_RETURN_SIZE
      WRITE(6, *) 'The data record buffer size: ',
*      RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The data record size: ',
*      RECORD_SIZE
      CALL GK$CLOSE_GKS()
      END
```

Workstation Description Table Inquiries

INQUIRE DEFAULT STROKE DEVICE DATA

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_6 RETURN
$ LINK      EXAMPLE_6 RETURN
$ RUN       EXAMPLE_6 RETURN
The error status:                0
The stroke buffer size:          80
The number of prompt/echo types: 2
The list of prompt/echo types:  1      4      0      0
                                0
The echo area:  0.0000000E+00  479.0000  0.0000000E+00  479.0000
The stroke data record:        80      0 -780059640 -780059640
                                0      0      0      0
                                0      0      0
The prompt/echo list return size: 2
The data record buffer size:     20
The data record size:           20
$
```

To review the functionality of INQUIRE DEFAULT STROKE DEVICE DATA within a larger program, refer to the stroke input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

INQUIRE DEFAULT VALUATOR DEVICE DATA

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The default values for the valuator input device are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The input device exists on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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,

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

record_buffer_length,
record_size)

GQDVL (*workstation_type, device_number, element, dim_dr,*
error_status, def_value, num_types, relement,
echo_area, low_val, high_val, len_dr, dr)

ginqdefval (*workstation_type, device_number, bufsize, data_size,*
data, error_status)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

initial_value

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the default initial value.

num_prompt_echo_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of valuator prompt and echo types available on a specified workstation.

prompt_echo_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array that contains the available valuator prompt and echo types on the specified workstation.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the device coordinate values that designate the input echo area on the workstation surface, in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the default valuator input record for the specified device.

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

num_returned_prompts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of prompt and echo types actually returned to this function. Compare this number with the actual number of available prompt and echo types to see if you had defined an array large enough to hold all available values.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on the workstation in routine **** |

Program Example

Example 11-7 illustrates the use of the function INQUIRE DEFAULT VALUATOR DEVICE DATA.

Workstation Description Table Inquiries

INQUIRE DEFAULT VALUATOR DEVICE DATA

Example 11-7: Determining the Default Valuator Input Values

```
C   This program writes the return values of the function
C   GKSSINQ_DEF_VALUATOR_DATA to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DEVICE_NUM,
*   LIST_PROMPT_TYPES( 5 ), NUM_PROMPT_ECHO, ERROR_STATUS,
*   PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE
      REAL ECHO_AREA( 4 ), INIT_VALUE, DATA_RECORD( 2 )
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKSSOPEN_GKS( 'SYS$ERROR:' )

C   Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 8

C   You can obtain this information as long as GKS is open.
      CALL GKSSINQ_DEF_VALUATOR_DATA( GKSSK_VT240, DEVICE_NUM,
*   ERROR_STATUS, INIT_VALUE, NUM_PROMPT_ECHO,
*   %DESCR( LIST_PROMPT_TYPES ), ECHO_AREA, DATA_RECORD,
*   PROMPT_RETURN_SIZE, RECORD_BUFFER_LENGTH, RECORD_SIZE )

C   Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The initial value: ', INIT_VALUE
      WRITE(6, *) 'The number of prompt/echo types: ',
*   NUM_PROMPT_ECHO
      WRITE(6, *) 'The list of prompt/echo types: ',
*   LIST_PROMPT_TYPES
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The valuator data record: ', DATA_RECORD
      WRITE(6, *) 'The prompt/echo list return size: ',
*   PROMPT_RETURN_SIZE
      WRITE(6, *) 'The data record buffer size: ',
*   RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The data record size: ', RECORD_SIZE
      CALL GKSCLOSE_GKS()
      END
```

Workstation Description Table Inquiries INQUIRE DEFAULT VALUATOR DEVICE DATA

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_7 RETURN
$ LINK     EXAMPLE_7 RETURN
$ RUN      EXAMPLE_7 RETURN
The error status: 0
The initial value: 0.5000000
The number of prompt/echo types: 3
The list of prompt/echo types: 1 2 3 0
0
The echo area: 533.0000 799.0000 0.0000000E+00 479.0000
The valuator data record: 0.0000000E+00 1.000000
The prompt/echo list return size: 3
The data record buffer size: 8
The data record size: 8
$
```

To review the functionality of INQUIRE DEFAULT VALUATOR DEVICE DATA within a larger program, refer to the valuator input programs in Chapter 7, Input Functions.

Workstation Description Table Inquiries

INQUIRE DISPLAY SPACE SIZE

INQUIRE DISPLAY SPACE SIZE

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE DISPLAY SPACE SIZE returns, for a specified workstation type, a flag specifying whether the device coordinate units are in meters or in some other form of measurement, the units for the workstation-specific device coordinates, and the display surface size in raster units.

The maximum display surface size is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is *not* of category GKS\$K_WSCAT_MO or GKS\$K_WSCAT_MI (refer to Chapter 9, Metafile Functions), or of category GKS\$K_WSCAT_WISS (refer to Chapter 3, Control Functions).

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the use of this function, refer to SET WORKSTATION VIEWPORT in Chapter 6, Transformation Functions.

Syntax

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

GQDSP (*workstation_type*, *error_status*, *units*, *px*, *py*, *ras_x*,
ras_y)

Workstation Description Table Inquiries

INQUIRE DISPLAY SPACE SIZE

ginqdisplaysize (*workstation_type, dspsz, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

meters

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that specifies whether or not the device coordinate units are measured in meters or in some device-determined unit of measurement. The argument can be any of the following values or constants:

| Value | Constant | Description |
|--------------|--------------------|--------------------|
| 0 | GKS\$K_METERS | Meters |
| 1 | GKS\$K_OTHER_UNITS | Some other unit |

Workstation Description Table Inquiries

INQUIRE DISPLAY SPACE SIZE

device_coordinates_x
device_coordinates_y

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the maximum X and Y values of the workstation surface, in device coordinates.

raster_units_x
raster_units_y

data type: **integer**
access: **write-only**
mechanism: **by reference**

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

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |

Workstation Description Table Inquiries INQUIRE DISPLAY SPACE SIZE

| Error Number | Completion Status Code | Message |
|-------------------------|-------------------------------|--|
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 31 | GKS\$ _ERROR_31 | Specified workstation is of category MO in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of cate- gory INPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES returns information concerning the ability of the workstation to dynamically generate segment transformations, visibility changes, highlighting changes, priority changes, content, and the effects of a segment deletion. If the workstation can dynamically change the surface, DEC GKS generates the segment changes immediately. If the workstation cannot dynamically change the surface, DEC GKS waits until the next update of the surface to regenerate only the output primitives contained in segments (implicit regeneration).

NOTE

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

The flags determining the ability to dynamically alter segment attributes are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

Syntax

GK\$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)

GQDSGA (*workstation_type*, *error_status*, *xform*, *vis_on_off*,
vis_off_on, *highlight*, *priority*, *add_prim*, *delete*)

ginqmodsegattr (*workstation_type*, *dyn*, *error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

transformation_change

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a segment transformation change, or whether GKS must implicitly regenerate the segment at the next surface update. The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

visible_to_invisible *invisible_to_visible*

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments indicate whether DEC GKS can dynamically implement a visibility change, or whether DEC GKS must implicitly regenerate the segment at the next surface update. (Some workstations may be able to make an invisible segment visible, but may not be sophisticated enough to make a visible segment invisible, forcing the workstation to redraw what is located behind the now invisible segment.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

highlight_change

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a highlighting change, or whether GKS must implicitly regenerate the segment at the next surface update. The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

priority_change

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a priority change, or whether GKS must implicitly regenerate the segment at the next surface update. The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

add_primitives

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically add output primitives, to an open segment, or whether GKS must implicitly regenerate the segment at the next surface update. The argument can be any of the following values or constants:

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

segment_deletion

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument indicates whether GKS can dynamically delete a segment, or whether GKS must implicitly regenerate the remaining segments at the next surface update. The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

Workstation Description Table Inquiries INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

| Error Number | Completion Status Code | Message |
|-------------------------|-------------------------------|---|
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES returns information concerning the ability of the workstation to dynamically implement a change in the definition of a representation index value. If the workstation can dynamically change the surface, DEC GKS generates the attribute changes immediately. If the workstation cannot dynamically change the surface, DEC GKS waits until the next update of the surface to regenerate only the output primitives contained in segments (implicit regeneration).

NOTE

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

The flags determining the ability to dynamically alter output attributes are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning output attributes, refer to Chapter 5, Output Attribute Functions. For more information concerning implicit regeneration, dynamic alteration, and operating states, refer to Chapter 3, Control Functions.

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Syntax

GKSS\$INQ_DYN_MOD_WS_ATTB (*workstation_type*,
error_status,
polyline_representation,
polymarker_representation,
text_representation,
fill_representation,
pattern_representation,
color_representation,
workstation_transformations)

GQDWKA (*workstation_type*, *error_status*, *pl_rep*, *pm_rep*, *t_rep*,
fa_rep, *pat_rep*, *c_rep*, *ws_xforms*)

ginqmodwsattr (*workstation_type*, *dyn*, *error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

the error messages listed in the Error Messages section, and all remaining output arguments are invalid.

polyline_representation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a polyline representation index value. (For more information, refer to SET POLYLINE REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

polymarker_representation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a polymarker representation index value. (For more information, refer to SET POLYMARKER REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

text_representation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a text representation index value. (For more information, refer to SET TEXT REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

fill_representation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a fill representation index value. (For more information, refer to SET FILL AREA REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

pattern_representation

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a pattern representation index value. (For more information, refer to SET PATTERN REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

color_representation

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the definition of a color representation index value. (For more information, refer to SET COLOR REPRESENTATION in Chapter 5, Output Attribute Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

workstation_transformations

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument indicates whether GKS can dynamically implement a change in the workstation window or workstation viewport. (For more information, refer to Chapter 6, Transformation Functions.) The argument can be any of the following values or constants:

| Value | Constant | Description |
|-------|------------|--------------------------|
| 0 | GKS\$K_IRG | Implicitly regenerated. |
| 1 | GKS\$K_IMM | Dynamically implemented. |

Workstation Description Table Inquiries

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE FILL AREA FACILITIES

INQUIRE FILL AREA FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The fill area facility information is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the fill area attributes, refer to Chapter 5, Output Attribute Functions.

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*)

GQFAF (*workstation_type*, *selement*, *helement*, *error_status*,
num_int, *r_selement*, *num_hatch*, *r_helement*,
num_index)

Workstation Description Table Inquiries

INQUIRE FILL AREA FACILITIES

ginqfillfacil (*workstation_type, bufsize, fac_size, fac, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_interior_styles

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of interior styles available to workstations of the specified type.

interior_style_list

data type: **array (integer)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array whose elements correspond to the four interior fill area styles. If the graphics handler supports the style, it writes the style's constant value to the array element. If the graphics handler does

Workstation Description Table Inquiries

INQUIRE FILL AREA FACILITIES

not support the style, it writes a -1 to the array element. The possible fill area style indexes are as follows:

| Value | Constant | Description |
|--------------|-------------------------|--------------------|
| 0 | GKS\$K_INTSTYLE_HOLLOW | Hollow |
| 1 | GKS\$K_INTSTYLE_SOLID | Solid |
| 2 | GKS\$K_INTSTYLE_PATTERN | Pattern |
| 3 | GKS\$K_INTSTYLE_HATCH | Hatched |

num_hatch_styles

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of hatch styles available to workstations of the specified type.

hatch_style_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the list of hatch styles available to workstations of the specified type.

num_fill_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of predefined fill index values available on the workstations of the specified type.

Workstation Description Table Inquiries

INQUIRE FILL AREA FACILITIES

hatch_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of hatch styles returned to the hatch style list. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE GENERALIZED DRAWING PRIMITIVE

INQUIRE GENERALIZED DRAWING PRIMITIVE

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function `INQUIRE GENERALIZED DRAWING PRIMITIVE` returns the number of attribute sets, and the list of those attribute sets that are associated with the specified generalized drawing primitive (GDP) identifier for a given workstation type.

The GDP information is available when DEC GKS is in any operating state except `GKS$K_GKCL`, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category `GKS$K_WSCAT_OUTPUT` or `GKS$K_WSCAT_OUTIN`.
- The workstation supports the GDP associated with the specified identifier.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning `GENERALIZED DRAWING PRIMITIVE`, refer to Chapter 4, Output Functions. For more information concerning supported GDPs, refer to Appendix I, DEC GKS GDPs and Escapes.

Syntax

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

`GQGDP` (*workstation_id*, *gdp_id*, *error_status*, *num_atts*, *list_atts*)

`ginqgdp` (*workstation_type*, *function_id*, *fac*, *error_status*)

Workstation Description Table Inquiries

INQUIRE GENERALIZED DRAWING PRIMITIVE

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

gdp_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the GDP identifier.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_attribute_sets

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of attribute sets applicable to the specified GDP on the specified workstation type.

Workstation Description Table Inquiries

INQUIRE GENERALIZED DRAWING PRIMITIVE

attribute_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array containing the list of attribute sets associated with the specified GDP identifier. The argument can be any of the following values or constants:

| Value | Constant | Description |
|--------------|---------------------|---------------------------|
| 0 | GKS\$K_POLYLN_ATTRI | GDP polyline attributes |
| 1 | GKS\$K_POLYMR_ATTRI | GDP polymarker attributes |
| 2 | GKS\$K_TEXT_ATTRI | GDP text attributes |
| 3 | GKS\$K_FILLAR_ATTRI | GDP fill area attributes |

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of attributes returned to the attribute list. You can use this argument to see if you specified an array that was large enough to hold all the returned GDPs.

Workstation Description Table Inquiries

INQUIRE GENERALIZED DRAWING PRIMITIVE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 41 | GKS\$_ERROR_41 | Specified workstation type is not able to generate the specified generalized drawing primitive in routine **** |

Workstation Description Table Inquiries

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES returns the number of available generalized drawing primitives (GDPs) and a list of the GDP identifiers for a given workstation type. For more information concerning GDPs, refer to Chapter 4, Output Functions.

The list of available GDPs is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN (for more information concerning supported GDPs, refer to Appendix I, DEC GKS GDPs and Escapes).
- The specified workstation can generate the given GDP.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

GKS\$INQ_AVAIL_GDP (*workstation_type, error_status, num_gdps, gdp_list, return_size*)

GQEGDP (*workstation_type, element, error_status, num_gdp, relement*)

Workstation Description Table Inquiries

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

ginqavailgdp (*workstation_type, max_gdps, start, gdps, actual_gdps, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_gdps

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of different GDP types.

gdp_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array that contains the integers representing the various supported GDPs for the specified workstation. For a list of the supported GDP types, refer to Appendix I, DEC GKS GDPs and Escapes.

Workstation Description Table Inquiries

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the actual number of GDPs passed back to the array. You can check this number to see if INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVE returned fewer values than spaces allocated in the array.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES returns, for a specified workstation type, the maximum number of polyline bundles, polymarker bundles, text bundles, fill area bundles, pattern indexes, and color indexes. The maximum workstation state table size is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or of category GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the use of this function, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQLWK (*workstation_type, error_status, m_pline, m_pmark, m_text, m_fill, m_patt, m_color*)

Workstation Description Table Inquiries

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

ginqmaxwsstables (*workstation_type, tables, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

max_pline

max_pmark

max_text

max_fill_area

max_pattern

max_color

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the maximum number of bundle indexes that the workstation state list can hold for each type of bundled index.

Workstation Description Table Inquiries

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is of category MO in routine **** |

Workstation Description Table Inquiries

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The numbers of logical input devices in each class are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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*)

GQLI (*workstation_type, error_status, num_loc, num_stk, num_val, num_ch, num_pi, num_stri*)

Workstation Description Table Inquiries

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

ginqnumavailinput (*workstation_type, num, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_locator_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of locator logical input devices supported by the specified workstation type.

num_stroke_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of stroke logical input devices supported by the specified workstation type.

Workstation Description Table Inquiries

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

num_valuator_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of valuator logical input devices supported by the specified workstation type.

num_choice_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of choice logical input devices supported by the specified workstation type.

num_pick_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of pick logical input devices supported by the specified workstation type.

num_string_devices

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of string logical input devices supported by the specified workstation type.

Workstation Description Table Inquiries

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

Operating States: WSOP, WSAC, SGOP

Description

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

The number of supported segment priorities is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation identifier exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQSGP (*workstation_type*, *error_status*, *num_pri*)

ginqnumsegpri (*workstation_type*, *numpri*, *error_status*)

Workstation Description Table Inquiries

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_priorities

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of segment priorities supported on a specified workstation type. If this function writes zero (0) to this argument, the device supports an infinite number of priorities.

Workstation Description Table Inquiries

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE PATTERN FACILITIES

INQUIRE PATTERN FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The number of available pattern indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning pattern representation and the other fill area attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQPAF (*workstation_type*, *error_status*, *nindexes*)

ginqpattfacil (*workstation_type*, *bufsize*, *fac_size*, *fac*,
error_status)

Workstation Description Table Inquiries

INQUIRE PATTERN FACILITIES

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_pattern_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of predefined pattern indexes supported on the specified workstation type.

Workstation Description Table Inquiries

INQUIRE PATTERN FACILITIES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE POLYLINE FACILITIES

INQUIRE POLYLINE FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE POLYLINE FACILITIES returns the number of line types and line widths, the representation for each type and width, and the number of polyline indexes available for a specified workstation type.

The polyline facilities are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polyline attributes, refer to Chapter 5, Output Attribute Functions.

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*)

GQPLF (*workstation_type, element, error_status, num_types, relement, num_widths, nom_width, min_width, max_width, nindexes*)

Workstation Description Table Inquiries

INQUIRE POLYLINE FACILITIES

ginqlinefacil (*workstation_type, bufsize, fac_size, fac, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_line_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of available line types on the specified workstation type.

line_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing line types available on the specified workstation type. The defined values are as follows:

Workstation Description Table Inquiries

INQUIRE POLYLINE FACILITIES

| Value | Constant | Description |
|-------|-------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_LINETYPE_SOLID | Solid line |
| 2 | GKS\$K_LINETYPE_DASHED | Dashed line |
| 3 | GKS\$K_LINETYPE_DOTTED | Dotted line |
| 4 | GKS\$K_LINETYPE_DASHED_DOTTED | Solid line |
| >= 5 | | Reserved for future standardization |

num_line_widths

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is the number of line widths available on the specified workstation type.

nominal_line_width

data type: **real**
 access: **write-only**
 mechanism: **by reference**

This argument is the default line width specified in device coordinates.

line_width_min *line_width_max*

data type: **real**
 access: **write-only**
 mechanism: **by reference**

These arguments are the minimum and maximum line widths, specified in device coordinates, that the workstation type can produce.

num_indexes

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is the number of predefined polyline bundle indexes supported by the specified workstation type.

Workstation Description Table Inquiries

INQUIRE POLYLINE FACILITIES

line_type_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of line types returned to the line type list. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$_ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE POLYMARKER FACILITIES

INQUIRE POLYMARKER FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE POLYMARKER FACILITIES returns the number of marker types and marker sizes, the representation for each type and size, and the number of polymarker indexes available for a given workstation type.

The polymarker facilities are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polymarker attributes, refer to Chapter 5, Output Attribute Functions.

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*)

Workstation Description Table Inquiries INQUIRE POLYMARKER FACILITIES

GQPMF (*workstation_type, element, error_status, num_types, relement, num_sizes, nom_size, min_size, max_size, nindexes*)

ginqmarkerfacil (*workstation_type, bufsize, fac_size, fac, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_marker_types

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of marker types available on the specified workstation type.

Workstation Description Table Inquiries

INQUIRE POLYMARKER FACILITIES

marker_types

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is an array containing marker types supported by the specified workstation type. The defined values are as follows:

| Value | Constant | Description |
|--------------|----------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_MARKERTYPE_DOT | A dot (.) |
| 2 | GKS\$K_MARKERTYPE_PLUS | A plus sign (+) |
| 3 | GKS\$K_MARKERTYPE_ASTERISK | An asterisk (*) |
| 4 | GKS\$K_MARKERTYPE_CIRCLE | A circle (o) |
| 5 | GKS\$K_MARKERTYPE_DIAGONAL_CROSS | A cross (X) |
| >= 6 | | Reserved for future standardization |

num_marker_sizes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of marker sizes available on the specified workstation type.

nominal_marker_size

data type: **F_float**
access: **write-only**
mechanism: **by reference**

This argument is the default size in device coordinates.

Workstation Description Table Inquiries

INQUIRE POLYMARKER FACILITIES

marker_size_min *marker_size_max*

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the minimum and maximum marker sizes, in device coordinates, that the specified workstation type can produce.

num_marker_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of predefined marker indexes supported by the workstation type.

marker_type_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of marker types returned to the marker list. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |

Workstation Description Table Inquiries

INQUIRE POLYMARKER FACILITIES

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED COLOR REPRESENTATION

INQUIRE PREDEFINED COLOR REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE PREDEFINED COLOR REPRESENTATION returns the predefined red, green, and blue intensities associated with a specific color index for a given workstation type.

The predefined color representation for a color index value is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.
- The color index is valid.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning color representation, refer to SET COLOR REPRESENTATION in Chapter 5, Output Attribute Functions.

Syntax

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

Workstation Description Table Inquiries

INQUIRE PREDEFINED COLOR REPRESENTATION

GQPCR (*workstation_type, cindex, error_status, red_i, green_i, blue_i*)

ginqpredcolourep (*workstation_type, index, rep, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

color_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined color index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

red_intensity
green_intensity
blue_intensity

Workstation Description Table Inquiries

INQUIRE PREDEFINED COLOR REPRESENTATION

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the predefined red, green, and blue intensities that comprise the color associated with the specified color index value.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 93 | GKS\$ _ERROR_93 | Specified color index is invalid in routine **** |
| 95 | GKS\$ _ERROR_95 | A representation for the specified color index has not been predefined on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED FILL AREA REPRESENTATION

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The predefined representation for a fill index value is available when DEC GKS is in any operating state except **GKS\$K_GKCL**, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category **GKS\$K_WSCAT_OUTPUT** or **GKS\$K_WSCAT_OUTIN**.
- The fill index value is valid.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning fill representation, refer to **SET FILL AREA REPRESENTATION** in Chapter 5, Output Attribute Functions.

Syntax

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

GQPFAR (*workstation_type, index, error_status, int_style, sindex, cindex*)

ginqpredfillrep (*workstation_type, index, rep, error_status*)

Workstation Description Table Inquiries

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

fill_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined fill index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE PREDEFINED FILL AREA REPRESENTATION

interior_style

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the predefined interior style associated with the fill index value. The defined values are as follows:

| Value | Constant | Description |
|--------------|-------------------------|--------------------|
| 0 | GKS\$K_INTSTYLE_HOLLOW | Hollow interior |
| 1 | GKS\$K_INTSTYLE_SOLID | Solid interior |
| 2 | GKS\$K_INTSTYLE_PATTERN | Pattern interior |
| 3 | GKS\$K_INTSTYLE_HATCH | Hatched interior |

style_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the style index associated with the specified fill index value. For more information concerning the style index, refer to SET FILL AREA STYLE INDEX in Chapter 5, Output Attribute Functions.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the specified fill index value. For more information concerning the color index, refer to SET COLOR REPRESENTATION in Chapter 5, Output Attribute Functions.

Workstation Description Table Inquiries

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 80 | GKS\$_ERROR_80 | Fill area index is invalid in routine **** |
| 82 | GKS\$_ERROR_82 | A representation for the specified fill area index has not been predefined on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED PATTERN REPRESENTATION

INQUIRE PREDEFINED PATTERN REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The predefined representation for a pattern index value is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.
- The workstation supports pattern fill areas.
- The specified pattern index is valid.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning patterns, refer to Chapter 5, Output Attribute Functions.

Syntax

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

Workstation Description Table Inquiries INQUIRE PREDEFINED PATTERN REPRESENTATION

GQPPAR (*workstation_type, pindex, max_x_dim, max_y_dim, error_status, dim_x, dim_y, carray*)

ginqpredpatrep (*workstation_type, index, rep, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

pattern_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined pattern index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE PREDEFINED PATTERN REPRESENTATION

height
width

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the number of rows and columns contained in the color index array used to create the pattern.

color_indexes

data type: **2-D array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the two-dimensional array of color indexes that designate how DEC GKS colors the pattern.

color_columns_return_size
color_rows_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the dimensions of the color array to which GKS returned index values. You can use these values to traverse only the elements of the array that contain valid color index values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |

Workstation Description Table Inquiries INQUIRE PREDEFINED PATTERN REPRESENTATION

| Error Number | Completion Status Code | Message |
|-----------------|------------------------|--|
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 85 | GKS\$_ERROR_85 | Specified pattern index is invalid in routine **** |
| 89 | GKS\$_ERROR_89 | A representation for the specified pattern index has not been predefined on this workstation in routine **** |
| 90 | GKS\$_ERROR_90 | Interior style PATTERN is not supported on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYLINE REPRESENTATION

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The predefined representation for a polyline index value is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polyline attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQPPLR (*workstation_type*, *pindex*, *error_status*, *ltype*, *lwidth*,
cindex)

ginqpredlinerep (*workstation_type*, *index*, *rep*, *error_status*)

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

polyline_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined polyline index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYLINE REPRESENTATION

line_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the line type. The defined values are as follows:

| Value | Constant | Description |
|-------|-------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_LINETYPE_SOLID | Solid line |
| 2 | GKS\$K_LINETYPE_DASHED | Dashed line |
| 3 | GKS\$K_LINETYPE_DOTTED | Dotted line |
| 4 | GKS\$K_LINETYPE_DASHED_DOTTED | Solid line |
| >= 5 | | Reserved for future standardization |

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the given polyline index value.

line_width_scale_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the line width scale factor. DEC GKS calculates line width by multiplying the scale factor by the nominal width.

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 60 | GKS\$ _ERROR_60 | Polyline index is not valid in routine **** |
| 62 | GKS\$ _ERROR_62 | A representation for the specified polyline index has not been predefined on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The predefined representation of the polymarker index value is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polymarker attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQPPMR (*workstation_type*, *pindex*, *error_status*, *mtype*, *msize*,
cindex)

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

ginqpredmarkerrep (*workstation_type, index, rep, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

polymarker_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined polymarker index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

marker_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the marker type associated with the specified polymarker bundle index value. The defined values are as follows:

| Value | Constant | Description |
|-------|----------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_MARKERTYPE_DOT | A dot (.) |
| 2 | GKS\$K_MARKERTYPE_PLUS | A plus sign (+) |
| 3 | GKS\$K_MARKERTYPE_ASTERISK | An asterisk (*) |
| 4 | GKS\$K_MARKERTYPE_CIRCLE | A circle (o) |
| 5 | GKS\$K_MARKERTYPE_DIAGONAL_CROSS | A cross (X) |
| >= 6 | | Reserved for future standardization |

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the specified polymarker index value.

marker_size_scale_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the marker size scale factor. DEC GKS calculates the marker size by multiplying the scale factor by the nominal size.

Workstation Description Table Inquiries

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 66 | GKS\$ _ERROR_66 | Polymarker index is invalid in routine **** |
| 68 | GKS\$ _ERROR_68 | A representation for the specified polymarker index has not been predefined on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE PREDEFINED TEXT REPRESENTATION

INQUIRE PREDEFINED TEXT REPRESENTATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The predefined representation for a text index value is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the text attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

Workstation Description Table Inquiries

INQUIRE PREDEFINED TEXT REPRESENTATION

GQPTXR (*workstation_type, tindex, error_status, font, precision, exp_factor, spacing, cindex*)

ginqpredtextrep (*workstation_type, index, rep, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

text_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a predefined text index value that must be valid for the specified workstation type.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

font_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the hardware or software font number. For information concerning the hardware fonts available on your workstation, refer to the

Workstation Description Table Inquiries

INQUIRE PREDEFINED TEXT REPRESENTATION

appropriate device-specific appendix in this manual. For more information concerning the software fonts available, refer to the appropriate appendix in this manual.

precision

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the text precision. The defined values are as follows:

| Value | Constant | Description |
|--------------|------------------------------|---------------------|
| 0 | GKS\$K_TEXT_PRECISION_STRING | String precision |
| 1 | GKS\$K_TEXT_PRECISION_CHAR | Character precision |
| 2 | GKS\$K_TEXT_PRECISION_STROKE | Stroke precision |

character_expansion_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the character expansion factor. The character expansion factor multiplied by the width-to-height ratio in the original font design determines the character width. The character expansion factor does not affect the height of the characters.

character_spacing

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the character spacing. Positive values increase the space between characters. Negative values decrease the space between characters. The value 0 places the character bodies adjacent to one another.

Workstation Description Table Inquiries INQUIRE PREDEFINED TEXT REPRESENTATION

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the specified text index value.

Workstation Description Table Inquiries

INQUIRE PREDEFINED TEXT REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 72 | GKS\$_ERROR_72 | Text index is invalid in routine **** |
| 74 | GKS\$_ERROR_74 | A representation for the specified text index has not been predefined on this workstation in routine **** |

Workstation Description Table Inquiries

INQUIRE TEXT FACILITIES

INQUIRE TEXT FACILITIES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE TEXT FACILITIES returns the number and list of available fonts, a list of precisions, the number of available character heights, the minimum and maximum character heights in device coordinates, the number of available character expansion factors, the minimum and maximum character expansion factors, and the number of text indexes available for a specified workstation type.

The text facilities are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning text attributes, refer to Chapter 5, Output Attribute Functions.

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*)

Workstation Description Table Inquiries

INQUIRE TEXT FACILITIES

GQTXF (*workstation_type, element, error_status, num_font, relement_f, relement_p, num_height, min_height, max_height, num_exp, min_exp, max_exp, nindexes*)

ginqtexfacil (*workstation_type, bufsize, fac_size, fac, error_status*)

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_fonts

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of hardware and software fonts available on the specified workstation type.

Workstation Description Table Inquiries

INQUIRE TEXT FACILITIES

font_list

precision_list

data type: **array (integer)**

access: **write-only**

mechanism: **by descriptor**

These arguments are arrays containing the available hardware and software font numbers, and the available precisions. The *precision_list* argument can contain any of the following values or constants:

| Value | Constant | Description |
|-------|------------------------------|--------------------|
| 0 | GKS\$K_TEXT_PRECISION_STRING | Lowest precision |
| 1 | GKS\$K_TEXT_PRECISION_CHAR | Moderate precision |
| 2 | GKS\$K_TEXT_PRECISION_STROKE | Highest precision |

num_heights

data type: **integer**

access: **write-only**

mechanism: **by reference**

This argument is the number of character heights available for the specified workstation type.

height_min

height_max

data type: **real**

access: **write-only**

mechanism: **by reference**

These arguments are the minimum and maximum character heights available for the specified workstation type, in device coordinates. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

Workstation Description Table Inquiries

INQUIRE TEXT FACILITIES

num_character_exp

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of character expansion values available for the specified workstation type. Character expansion values affect the character width. For more information, refer to the text attributes section in Chapter 5, Output Attribute Functions.

character_exp_min

character_exp_max

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the minimum and maximum character expansion values available for the specified workstation types.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of predefined index values associated with the specified workstation type.

precision_return_size

font_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the number of the elements in the precision and font arrays. You can use these values to make sure that you declared arrays large enough to hold all the font and precision types.

Workstation Description Table Inquiries

INQUIRE TEXT FACILITIES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -33 | DECGKS\$ _ERROR_NEG_33 | Array descriptor is not acceptable in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$ _ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$ _ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Workstation Description Table Inquiries

INQUIRE WORKSTATION CATEGORY

INQUIRE WORKSTATION CATEGORY

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The workstation category is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the specified workstation identifier exists and is valid. If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning workstation categories and operating states, refer to Chapter 3, Control Functions.

Syntax

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

GQWKCA (*workstation_type*, *error_status*, *category*)

ginqwscategory (*workstation_type*, *cat*, *error_status*)

Workstation Description Table Inquiries

INQUIRE WORKSTATION CATEGORY

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

workstation_category

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the category of the specified workstation type. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------|---|
| 0 | GKS\$K_WSCAT_OUTPUT | Output category |
| 1 | GKS\$K_WSCAT_INPUT | Input category |
| 2 | GKS\$K_WSCAT_OUTIN | Output/Input category |
| 3 | GKS\$K_WSCAT_WISS | Workstation independent segment storage |
| 4 | GKS\$K_WSCAT_MO | Metafile Output category |
| 5 | GKS\$K_WSCAT_MI | Metafile Input category |

Workstation Description Table Inquiries

INQUIRE WORKSTATION CATEGORY

For more information concerning segments, refer to Chapter 8, Segment Functions. For more information concerning metafiles, refer to Chapter 9, Metafile Functions.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |

Workstation Description Table Inquiries

INQUIRE WORKSTATION CLASSIFICATION

INQUIRE WORKSTATION CLASSIFICATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE WORKSTATION CLASSIFICATION returns the type of display surface hardware for a specified workstation type.

The workstation classification is available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the following conditions exist:

- The specified workstation type exists and is valid.
- The workstation is *not* of category GKS\$K_WSCAT_WISS, GKS\$K_WSCAT_MO, or GKS\$K_WSCAT_MI.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning workstation categories and DEC GKS operating states, refer to Chapter 3, Control Functions.

Syntax

GKS\$WS_CLASSIFICATION (*workstation_type*, *error_status*,
classification)

GQWKCL (*workstation_type*, *error_status*, *class*)

ginqwsclass (*workstation_type*, *class*, *error_status*)

Workstation Description Table Inquiries

INQUIRE WORKSTATION CLASSIFICATION

Arguments

workstation_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that specifies the workstation type. For a list of the supported workstation types, refer to the appropriate appendix in this manual.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

classification

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the classification of the device associated with the specified workstation type. You can use the workstation classification to determine the validity of other GKS return values. For instance, if you are working on a device other than one which uses raster units to define pixel dimensions, the function INQUIRE DISPLAY SPACE SIZE will not return valid values to the raster unit arguments.

Workstation Description Table Inquiries INQUIRE WORKSTATION CLASSIFICATION

The defined values are as follows:

| Value | Constant | Description |
|-------|-----------------------|---------------|
| 0 | GKS\$K_WSCLASS_VECTOR | Vector device |
| 1 | GKS\$K_WSCLASS_RASTER | Raster device |
| 2 | GKS\$K_WSCLASS_OTHERD | Other device |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 31 | GKS\$_ERROR_31 | Specified workstation is of category MO in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified Workstation Independent Segment Storage in routine **** |

GKS State List Inquiries

GKS State List Inquiries

This section describes the DEC GKS state list inquiries. (For more information concerning the DEC GKS state list, refer to Chapter 3, Control Functions.) You use these functions if you are not aware of the current DEC GKS operating state, of the current normalization transformation number, of the current individual output attribute values, of the current clipping indicator, or of the list of currently open or active workstations.

INQUIRE CLIPPING

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE CLIPPING returns the current value of the world viewport clipping flag.

The clipping indicator is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning clipping, refer to Chapter 6, Transformation Functions.

Syntax

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

GQCLIP (*error_status, cflag, crec*)

ginqclip (*clipping, error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

GKS State List Inquiries

INQUIRE CLIPPING

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

clipping_indicator

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current setting of the clipping flag. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|--------------------|
| 0 | GKS\$K_NOCLIP | Clipping is off. |
| 1 | GKS\$K_CLIP | Clipping is on. |

clipping_rectangle

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is the 4-element array that contains the dimensions of the current clipping rectangle, in normalized device coordinates. DEC GKS stores the minimum X value in the first element, the maximum X value in the second element, the minimum Y value in the third element, and the maximum Y value in the last element of the array.

GKS State List Inquiries INQUIRE CLIPPING

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

FORTRAN Functions: GQLN, GQLWSC, GQPLCI, GQMK, GQMKSC, GQPMCI, GQTXFP, GQCHXP, GQCHSP, GQTXCI, GQFAIS, GQFASI, GQFACI, GQASF

C Functions: ginqindivattr, ginqlinetype, ginqlinewidth, ginqlinecolour, ginqmarkertype, ginqmarkersize, ginqmarkercolour, ginqtextfontprec, ginqcharexpan, ginqcharspace, ginqtextcolourind, ginqfillintstype, ginqfillstyleind, ginqfillcolourind, ginqasf

Description

The function INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES returns the current values for each of the nongeometric output attributes.

The current individual output attributes are available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning output attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

GKS\$INQ_INDIV_ATT (*error_status, polyline_type, polyline_width, polyline_color_index, polymarker_type, polymarker_size, polymarker_color_index, text_font, text_precision, character_expansion_factor,*

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

*character_spacing, text_color_index,
interior_style, style_index,
fill_color_index, aspect_source_flags)*

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

polyline_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current line type. The defined values are as follows:

| Value | Constant | Description |
|-------|-------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_LINETYPE_SOLID | Solid line |
| 2 | GKS\$K_LINETYPE_DASHED | Dashed line |
| 3 | GKS\$K_LINETYPE_DOTTED | Dotted line |
| 4 | GKS\$K_LINETYPE_DASHED_DOTTED | Solid line |
| >= 5 | | Reserved for future standardization |

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

polyline_width

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current line width scale factor. DEC GKS calculates line width by multiplying the scale factor by the nominal width.

polyline_color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current polyline color index.

polymarker_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current marker type. The defined values are as follows:

| Value | Constant | Description |
|--------------|----------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_MARKERTYPE_DOT | A dot (.) |
| 2 | GKS\$K_MARKERTYPE_PLUS | A plus sign (+) |
| 3 | GKS\$K_MARKERTYPE_ASTERISK | An asterisk (*) |
| 4 | GKS\$K_MARKERTYPE_CIRCLE | A circle (o) |
| 5 | GKS\$K_MARKERTYPE_DIAGONAL_CROSS | A cross (X) |
| >= 6 | | Reserved for future standardization |

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

polymarker_size

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current marker size scale factor. DEC GKS calculates the marker size by multiplying the scale factor by the nominal size.

polymarker_color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current polymarker color index.

text_font

text_precision

data type: **integer**
access: **write-only**
mechanism: **by reference**

The first argument is the current hardware or software font number. For information concerning the hardware fonts available on your workstation, refer to the appropriate device-specific appendix in this manual. For more information concerning the software fonts available, refer to the appropriate appendix in this manual.

The second argument is the current text precision. The defined values are as follows:

| Value | Constant | Description |
|--------------|------------------------------|---------------------|
| 0 | GKS\$K_TEXT_PRECISION_STRING | String precision |
| 1 | GKS\$K_TEXT_PRECISION_CHAR | Character precision |
| 2 | GKS\$K_TEXT_PRECISION_STROKE | Stroke precision |

character_expansion_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character expansion factor. The character expansion factor multiplied by the width to height ratio in the original font

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

design determines the character width. The character expansion factor does not affect the height of the characters.

character_spacing

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character spacing. Positive values increase the space between characters. Negative values decrease the space between characters. The value 0 places the character bodies adjacent to one another.

text_color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current text color index.

interior_style

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current fill area interior style. The defined values are as follows:

| Value | Constant | Description |
|--------------|-------------------------|--------------------|
| 0 | GKS\$K_INTSTYLE_HOLLOW | Hollow interior |
| 1 | GKS\$K_INTSTYLE_SOLID | Solid interior |
| 2 | GKS\$K_INTSTYLE_PATTERN | Pattern interior |
| 3 | GKS\$K_INTSTYLE_HATCH | Hatched interior |

style_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current style index. For more information concerning the style index, refer to SET FILL AREA STYLE INDEX in Chapter 5, Output Attribute Functions.

GKS State List Inquiries INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

fill_color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current fill area color index. For more information concerning the color index, refer to SET COLOR REPRESENTATION in Chapter 5, Output Attribute Functions.

aspect_source_flags

data type: **array (integer)**
access: **write-only**
mechanism: **by reference**

This argument is a 13-element array containing the aspect source flag for each of the nongeometric output attributes. The aspect source flag determines whether DEC GKS uses the individual or bundled attribute value for each of the nongeometric output attributes. (For detailed information concerning the aspect source flags, refer to Chapter 5, Output Attribute Functions.)

The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--------------------------------------|
| 0 | GKS\$K_ASF_BUNDLED | Use the bundled attribute values. |
| 1 | GKS\$K_ASF_INDIVIDUAL | Use the individual attribute values. |

GKS State List Inquiries

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The current normalization transformation number is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning normalization transformations, refer to Chapter 6, Transformation Functions.

Syntax

GKS\$INQ_CURRENT_XFORMNO (*error_status*,
transformation_number)

GQCNTN (*error_status*, *xform*)

ginqcurtrannum (*tran*, *error_status*)

GKS State List Inquiries

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

transformation_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of the normalization transformation currently in effect.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The current bundle indexes and geometric attributes are available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning output attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

GKS\$INQ_PRIM_ATT (*error_status, list_bundle_indexes, character_height, character_up_vector, character_width, text_base_vector, character_path, character_alignment, pattern_width, pattern_height, pattern_reference_point*)

ginqprimattr (*primattr, error_status*)

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

list_bundle_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the current bundle indexes. The order of the bundle indexes is as follows:

1. Polyline index
2. Polymarker index
3. Text index
4. Fill area index

character_height

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character height specified by a world coordinate value.

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

character_up_vector

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is an array with two elements that contains the X and Y world coordinates comprising the current character-up vector. The character-up vector, in conjunction with the text string starting point, establishes an upward direction for the characters in a text string. For more information, refer to Chapter 5, Output Attribute Functions.

character_width

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character width in world coordinate units.

text_base_vector

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is an array with two elements containing the text base vectors. Using the starting point and the base vectors, you can calculate the line on which the text extent rectangle is positioned.

character_path

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current character path. The character path determines in which direction along the imaginary text line GKS writes the characters. The defined values are as follows:

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

| Value | Constant | Description |
|-------|------------------------|--|
| 0 | GKS\$K_TEXT_PATH_RIGHT | From left to right |
| 1 | GKS\$K_TEXT_PATH_LEFT | From right to left |
| 2 | GKS\$K_TEXT_PATH_UP | From the bottom to the top along the character-up vector |
| 3 | GKS\$K_TEXT_PATH_DOWN | From the top to the bottom along the character-up vector |

character_alignment

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is a 2-element array containing the horizontal (in the first element) and vertical values of the current character alignment (in the second element). The character alignment designates how GKS positions the text extent rectangle along the imaginary text line. The defined horizontal values are as follows:

| Value | Constant | Description |
|-------|---------------------------|---------------|
| 0 | GKS\$K_TEXT_HALIGN_NORMAL | Default value |
| 1 | GKS\$K_TEXT_HALIGN_LEFT | Left |
| 2 | GKS\$K_TEXT_HALIGN_CENTER | Center |
| 3 | GKS\$K_TEXT_HALIGN_RIGHT | Right |

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

The defined vertical values are as follows:

| Value | Constant | Description |
|--------------|---------------------------|--------------------|
| 0 | GKS\$K_TEXT_VALIGN_NORMAL | Default value |
| 1 | GKS\$K_TEXT_VALIGN_TOP | Top |
| 2 | GKS\$K_TEXT_VALIGN_CAP | Cap |
| 3 | GKS\$K_TEXT_VALIGN_HALF | Half |
| 4 | GKS\$K_TEXT_VALIGN_BASE | Base |
| 5 | GKS\$K_TEXT_VALIGN_BOTTOM | Bottom |

For more information, refer to the figures in Chapter 5, Output Attribute Functions.

pattern_width

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 2-element array containing the pattern width vector. The first element contains the X vector in world coordinates and the second element contains the Y vector in world coordinates.

pattern_height

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 2-element array containing the pattern height vector. The first element contains the X vector in world coordinates and the second element contains the Y vector in world coordinates.

GKS State List Inquiries

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

pattern_reference_point

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 2-element array that contains the world coordinate values designating the pattern reference point. None of the DEC GKS supported workstations support this feature, so the returned values will always be zero.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE INPUT QUEUE OVERFLOW

INQUIRE INPUT QUEUE OVERFLOW

Operating States: WSOP, WSAC, SGOP

Description

If the input queue is overflowed, and if information about the cause of the overflow is available, the function INQUIRE INPUT QUEUE OVERFLOW returns a zero (0) to the error status argument and writes valid values to its remaining arguments. Otherwise, this function returns an error to the error status argument that explains why information is not available.

Syntax

GK\$INQ_INPUT_QUEUE_OVERFLOW (*error_status*,
workstation_id,
input_class,
device_number)

GQIQOV (*error_status*, *workstation_id*, *in_class*, *device_number*)

ginqinputoverflow (*overflow*, *error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, the input queue overflowed and all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are

GKS State List Inquiries

INQUIRE INPUT QUEUE OVERFLOW

invalid; the error message explains why information about the overflow is not available.

workstation_id

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the workstation identifier of the workstation whose input device caused the queue to overflow.

input_class

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the input class of the device that generated the event that caused the input queue to overflow.

device_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of the input device that generated the event that caused the input queue to overflow.

GKS State List Inquiries

INQUIRE INPUT QUEUE OVERFLOW

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 7 | DECGKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 148 | DECGKS\$ _ERROR_NEG_148 | Input queue has not overflowed since GKS was opened or the last invocation of INQUIRE INPUT QUEUE OVERFLOW in routine **** |
| 149 | GKS\$ _ERROR_149 | Input queue has overflowed, but associated workstation has been closed in routine **** |

GKS State List Inquiries

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The list of normalization transformations is available when DEC GKS is in any operating state except GKS\$K_GKCL. If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning transformations, refer to Chapter 6, Transformation Functions.

Syntax

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

GQENTN (*element, error_status, num_xforms, relement*)

ginqnrannum (*max_ntrans, start, ntrans, actual_ntrans, error_status*)

GKS State List Inquiries

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_transformations

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of currently defined normalization transformations.

list_transformations

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the list of all the currently defined normalization transformation numbers in order of input viewport priority.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of normalization transformation numbers returned to the list. By comparing this argument to the number of returned transformations, you can determine if you defined an array large enough to hold all the returned values.

GKS State List Inquiries

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE MORE SIMULTANEOUS EVENTS

INQUIRE MORE SIMULTANEOUS EVENTS

Operating States: WSOP, WSAC, SGOP

Description

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

Syntax

GKS\$INQ_MORE_SIMUL_EVENTS (*error_status*,
more_events_flag)

GQSIM (*error_status*, *flag*)

ginqmoreevents (*events*, *error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to the error message listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS State List Inquiries

INQUIRE MORE SIMULTANEOUS EVENTS

more_events_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that specifies whether there exist more simultaneously generated events on the event input queue. This argument can be either of the following values:

| Value | Constant | Description |
|--------------|----------------------|---|
| 0 | GKS\$K_NOMORE_EVENTS | There are no more simultaneously generated events on the queue. |
| 1 | GKS\$K_MORE_EVENTS | There are more simultaneously generated events on the queue. |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to the number in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 7 | DECGKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

INQUIRE NAME OF OPEN SEGMENT

Operating States: SGOP

Description

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

The name of the open segment is available only when DEC GKS is in the operating state GKS\$K_SGOP. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

GKS\$INQ_NAME_OPEN_SEG (*error_status, segment_name*)

GQOPSG (*error_status, segment_name*)

ginqnameopenseg (*segment_name error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS State List Inquiries

INQUIRE NAME OF OPEN SEGMENT

segment_name

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the integer identifier known as the segment name.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 4 | GKS\$_ERROR_4 | GKS not in proper state: GKS shall be in the state SGOP in routine **** |

GKS State List Inquiries

INQUIRE NORMALIZATION TRANSFORMATION

INQUIRE NORMALIZATION TRANSFORMATION

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The world window and viewport values are available when DEC GKS is in any operating state except GKS\$K_GKCL, and if the given normalization transformation number is valid. If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning transformations, refer to Chapter 6, Transformation Functions.

Syntax

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

GQNT (*xform, error_status, window, vport*)

ginqntran (*num, tran, error_status*)

GKS State List Inquiries

INQUIRE NORMALIZATION TRANSFORMATION

Arguments

transformation_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the specified normalization transformation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

world_window_boundaries

world_viewport_boundaries

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

These arguments are 4-element arrays containing coordinate points in the order XMIN, XMAX, YMIN, YMAX. These arguments are the boundaries of the world window and world viewport, in world coordinates and normalized device coordinates, respectively, associated with the specified normalization transformation number.

GKS State List Inquiries

INQUIRE NORMALIZATION TRANSFORMATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 50 | GKS\$_ERROR_50 | Transformation number is invalid in routine **** |

GKS State List Inquiries

INQUIRE OPERATING STATE VALUE

INQUIRE OPERATING STATE VALUE

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

Description

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

The DEC GKS operating state is always available to this inquiry function.

Syntax

GKS\$INQ_OPERATING_STATE (*operating_state*)

GQOPS (*op_state*)

ginqopst (*state*)

GKS State List Inquiries

INQUIRE OPERATING STATE VALUE

Arguments

operating_state

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current GKS operating state. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|-------------------------------------|
| 0 | GKS\$K_GKCL | GKS is closed. |
| 1 | GKS\$K_GKOP | GKS is open. |
| 2 | GKS\$K_WSOP | At least one workstation is open. |
| 3 | GKS\$K_WSAC | At least one workstation is active. |
| 4 | GKS\$K_SGOP | A segment is being created. |

Error Messages

This inquiry function never returns an error status.

GKS State List Inquiries

INQUIRE PICK IDENTIFIER VALUE

INQUIRE PICK IDENTIFIER VALUE

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE PICK IDENTIFIER VALUE returns the current pick identifier.

The current pick identifier is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning pick identification, refer to Chapter 8, Segment Functions.

Syntax

GKS\$INQ_PICK_ID (*error_status, pick_identifier*)

GQPKID (*error_status, pick_id*)

ginqcurpickid (*pickid, error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS State List Inquiries

INQUIRE PICK IDENTIFIER VALUE

pick_identifier

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current integer value that you use to identify a portion of a segment.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE SET OF ACTIVE WORKSTATIONS

INQUIRE SET OF ACTIVE WORKSTATIONS

Operating States: GKOP, WSOP, WSAC, SGOP

Description

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

The list of active workstations is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

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

GQACWK (*member, error_status, num_active, rmember*)

ginqactivews (*max_ids, start, wsids, actual_ids, error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS State List Inquiries

INQUIRE SET OF ACTIVE WORKSTATIONS

num_workstations

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstations currently active.

workstation_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing the identifiers associated with the currently active workstations.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstation identifiers returned to the active workstation list. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

GKS State List Inquiries

INQUIRE SET OF ACTIVE WORKSTATIONS

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$_ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE SET OF OPEN WORKSTATIONS

INQUIRE SET OF OPEN WORKSTATIONS

Operating States: GKOP, WSOP, WSAC, SGOP

Description

The function INQUIRE SET OF OPEN WORKSTATIONS returns the current set of identifiers associated with open workstations.

The current list of open workstations is available when DEC GKS is in any operating state except GKS\$K_GKCL. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Syntax

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

GQOPWK (*member, error_status, num_open, rmember*)

ginqopenws (*max_ids, start, wsids, actual_ids, error_status*)

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

GKS State List Inquiries

INQUIRE SET OF OPEN WORKSTATIONS

num_open_workstations

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstations currently open.

workstation_list

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing the identifiers associated with the currently open workstations.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstation identifiers returned to the open workstation list. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

GKS State List Inquiries

INQUIRE SET OF OPEN WORKSTATIONS

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 8 | GKS\$ _ERROR_8 | GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

GKS State List Inquiries

INQUIRE SET OF SEGMENT NAMES IN USE

INQUIRE SET OF SEGMENT NAMES IN USE

Operating States: WSOP, WSAC, SGOP

Description

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

The list of segment names is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP. If this condition is not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQSGUS (*member, error_status, num_seg, rmember*)

ginqsegnames (*max_segnames, start, segnames,*
actual_segnames, error_status)

GKS State List Inquiries

INQUIRE SET OF SEGMENT NAMES IN USE

Arguments

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_segments

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of currently existing segments.

list_segments

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing segment names that correspond to all the currently existing segments.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of segment names returned to the list of stored segments. By comparing this argument to the actual list, you can determine if you defined an array large enough to hold all the returned values.

GKS State List Inquiries

INQUIRE SET OF SEGMENT NAMES IN USE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

Workstation State List Inquiries

This section describes the workstation state list inquiries. (For more information concerning the workstation state list, refer to Chapter 3, Control Functions.) You use these functions if you need information about the state of a single workstation, which is identified by a numeric workstation identifier, or if you are not aware of the current workstation transformation, the locator device state, the current segment priority, or the workstation update state.

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

INQUIRE CHOICE DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE CHOICE DEVICE STATE returns initialization values for the choice logical input device, and the input operating mode.

The choice logical input device state is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The device supports the choice logical input device.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

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*)

Workstation State List Inquiries INQUIRE CHOICE DEVICE STATE

GQCHS (*workstation_id, device_number, dim_dr, error_status, operating_mode, echo_flag, in_status, in_choice, p_e_type, echo_area, len_dr, dr*)

ginqchoicest (*workstation_id, device_number, bufsize, state_size, state, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

operating_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether or not input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

initial_choice_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument determines if the user can return a measure value of "No Choice." If No Choice can be returned, then the user can press RETURN as soon as the menu appears, without returning the value corresponding with the initial choice. This action returns the value 0 as the logical input device's measure. If the user cannot return No Choice, then pressing

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

RETURN when the menu appears returns the value of the highlighted initial choice.

The defined values are as follows:

| Value | Constant | Description |
|-------|------------------------|----------------------------|
| 1 | GKS\$K_STATUS_OK | Return the initial number. |
| 2 | GKS\$K_STATUS_NOCHOICE | Return No Choice. |

initial_choice_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current choice measure that represents one of the current choices, expressed as an integer.

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing coordinates in the order XMIN, XMAX, YMIN, YMAX. The points designate the input echo area on the workstation surface. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

INQUIRE CHOICE DEVICE STATE returns a different amount of information depending on the value contained in the first component of the data record. If you pass the address of an integer with the value 0 as this

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

argument, and the value 4 as the *record_buffer_length* argument, then this function only returns the default number of choices. This functionality allows you to see if your declared string buffers are large enough to hold all the current strings.

Once you obtain the current number of choices, you must initialize the arrays containing string sizes, string addresses, and strings, and then call INQUIRE CHOICE DEVICE STATE a second time. In the second call, pass the number of choices obtained in the first call to INQUIRE CHOICE DEVICE STATE and pass the *record_buffer_length* value that specifies the whole data record. Then the function writes all the current values to its write-only arguments.

To understand the process of calling INQUIRE CHOICE DEVICE STATE twice, refer to the program example in this function description.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on workstation in routine **** |

Program Example

Example 11-8 illustrates the use of the function INQUIRE CHOICE DEVICE STATE.

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

Example 11-8: Determining the State of the Choice Logical Input Device

```
C      This program writes the return values of the function
C      GKS$INQ_CHOICE_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 3 ), NUM_CHOICES,
* PROMPT_ECHO_TYPE, ERROR_STATUS, DEVICE_NUM,
* INPUT_MODE, ECHO_FLAG, RECORD_BUFFER_LENGTH,
* RECORD_SIZE, STRING_SIZES( 10 ),
* STRING_POINTERS( 10 ), INIT_STATUS, INIT_CHOICE,
* I

      CHARACTER*80 STRINGS(10)

      REAL ECHO_AREA( 4 )
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

C      First element in the data record is the number of choices.
      EQUIVALENCE( DATA_RECORD( 1 ), NUM_CHOICES )

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )

C      Specifying zero as this argument forces GKS$INQ_CHOICE_STATE to
C      only return the current number of choices.
      NUM_CHOICES = 0

C      Tell the handler the size of the record buffer (do not include
C      the array addresses in this call).
      RECORD_BUFFER_LENGTH = 4

C      You can obtain this information as long as the specified
C      workstation is open.
      CALL GKS$INQ_CHOICE_STATE( WS_ID, DEVICE_NUM,
* ERROR_STATUS, INPUT_MODE, ECHO_FLAG, INIT_STATUS,
* INIT_CHOICE, PROMPT_ECHO_TYPE, ECHO_AREA,
* DATA_RECORD, RECORD_BUFFER_LENGTH, RECORD_SIZE )

C      Tell the handler where to write current addresses...
      DATA_RECORD( 2 ) = %LOC( STRING_SIZES )
      DATA_RECORD( 3 ) = %LOC( STRING_POINTERS )

C      Initialize the string pointers...
      DO 100 I = 1, NUM_CHOICES
          STRING_POINTERS( I ) = %LOC( STRINGS(I) )
          STRING_SIZES( I ) = 80
100  CONTINUE
```

(continued on next page)

Workstation State List Inquiries INQUIRE CHOICE DEVICE STATE

Example 11-8 (Cont.): Determining the State of the Choice Logical Input Device

```
C      Initialize the size of the data record...
      RECORD_BUFFER_LENGTH = 12

C      You can obtain this information as long as the specified
C      workstation is open.
      CALL GKS$INO_CHOICE_STATE( WS_ID, DEVICE_NUM,
*     ERROR_STATUS, INPUT_MODE, ECHO_FLAG, INIT_STATUS,
*     INIT_CHOICE, PROMPT_ECHO_TYPE, ECHO_AREA,
*     DATA_RECORD, RECORD_BUFFER_LENGTH, RECORD_SIZE )

C      Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The input operating mode: ', INPUT_MODE
      WRITE(6, *) 'The echo flag: ', ECHO_FLAG
      WRITE(6, *) 'The initial choice status: ', INIT_STATUS
      WRITE(6, *) 'The initial choice value: ', INIT_CHOICE
      WRITE(6, *) 'The prompt and echo type: ',
*     PROMPT_ECHO_TYPE
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The data record: ', DATA_RECORD
      WRITE(6, *) 'The maximum data length: ',
*     RECORD_BUFFER_LENGTH
      WRITE(6, *) 'Size of returned data record: ', RECORD_SIZE

C      STRINGS holds the current choice strings...
      WRITE(6,*) 'The current choice strings are as follows:'
      DO 200 i = 1, NUM_CHOICES
        WRITE(6,*) STRINGS(I)
200    CONTINUE

      CALL GKS$CLOSE_WS( WS_ID )
      CALL GKS$CLOSE_GKS()
      END
```

Workstation State List Inquiries

INQUIRE CHOICE DEVICE STATE

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_8 
$ LINK      EXAMPLE_8 
$ RUN       EXAMPLE_8 
The error status:                0
The input operating mode:        0
The echo flag:                   1
The initial choice status:       1
The initial choice value:        1
The prompt and echo type:        1
The echo area:   533.0000      799.0000      0.0000000E+00      479.0000
The data record:   5          1036          1076
The maximum data length:         12
Size of returned data record:    12
The current choice strings are as follows:
CHOICE1
CHOICE2
CHOICE3
CHOICE4
CHOICE5
$
```

Workstation State List Inquiries

INQUIRE COLOR REPRESENTATION

INQUIRE COLOR REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE COLOR REPRESENTATION returns the red, green, and blue intensities associated with a given color index, on a specified workstation.

The color representation is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.
- The color index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning color representation, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQCR (*workstation_id, cindex, type, error_status, red_i, green_i, blue_i*)

Workstation State List Inquiries

INQUIRE COLOR REPRESENTATION

ginqcolourrep (*workstation_id, index, type, rep, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

color_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is a color index value defined on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

Workstation State List Inquiries

INQUIRE COLOR REPRESENTATION

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

red_intensity
green_intensity
blue_intensity

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the red, green, and blue intensities associated with the specified color index.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -17 | DECGKS\$ _ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

Workstation State List Inquiries

INQUIRE COLOR REPRESENTATION

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 93 | GKS\$_ERROR_93 | Color index is invalid in routine **** |
| 94 | GKS\$_ERROR_94 | A representation for the specified color index has not been defined on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE FILL AREA REPRESENTATION

INQUIRE FILL AREA REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

The function **INQUIRE FILL AREA REPRESENTATION** returns the values associated with the given fill area index, on the specified workstation.

The fill area representation is available when DEC GKS is in any operating state except **GKS\$K_GKCL** or **GKS\$KOP**, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is *not* of category **GKS\$K_WSCAT_MI**, **GKS\$K_WSCAT_INPUT**, or **GKS\$K_WSCAT_WISS**.
- The fill area index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the fill area representation, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQFAR (*workstation_id, findex, type, error_status, int_style, sindex, cindex*)

ginqfillrep (*workstation_id, index, type, rep, error_status*)

Workstation State List Inquiries

INQUIRE FILL AREA REPRESENTATION

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

fill_area_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the defined fill area index on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

Workstation State List Inquiries INQUIRE FILL AREA REPRESENTATION

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

interior_style

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the interior style associated with the specified fill area index. The defined values are as follows:

| Value | Constant | Description |
|-------|-------------------------|-------------|
| 0 | GKS\$K_INTSTYLE_HOLLOW | Hollow |
| 1 | GKS\$K_INTSTYLE_SOLID | Solid |
| 2 | GKS\$K_INTSTYLE_PATTERN | Pattern |
| 3 | GKS\$K_INTSTYLE_HATCH | Hatched |

style_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the style index value associated with the specified fill area index value.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index value associated with the specified fill area index value.

Workstation State List Inquiries

INQUIRE FILL AREA REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 80 | GKS\$_ERROR_80 | Fill area index is invalid in routine **** |
| 81 | GKS\$_ERROR_81 | A representation for the specified fill area index has not been defined on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF COLOR INDICES

INQUIRE LIST OF COLOR INDICES

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE LIST OF COLOR INDICES returns the number and the list of defined color indexes.

The list of color indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning color indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQECI (*workstation_id, element, error_status, num_color, relement*)

ginqcolourindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF COLOR INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of currently defined color indexes.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array that contains the currently defined color index values.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the color index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Workstation State List Inquiries

INQUIRE LIST OF COLOR INDICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF FILL AREA INDICES

INQUIRE LIST OF FILL AREA INDICES

Operating States: WSOP, WSAC, SGOP

Description

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

The list of fill area indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the fill area indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQEFAI (*workstation_id, element, error_status, num_fill, relement*)

gingfillindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF FILL AREA INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of defined fill area index values for the specified workstation.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined fill area index values.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the fill area index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Workstation State List Inquiries

INQUIRE LIST OF FILL AREA INDICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF PATTERN INDICES

INQUIRE LIST OF PATTERN INDICES

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE LIST OF PATTERN INDICES returns the number and the list of defined pattern indexes on the specified workstation.

The list pattern indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning patterns, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQEPAI (*workstation_id, element, error_status, num_patt, relement*)

ginqpatindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF PATTERN INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of defined pattern index values for the specified workstation.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined pattern index values.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the pattern index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Workstation State List Inquiries

INQUIRE LIST OF PATTERN INDICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF POLYLINE INDICES

INQUIRE LIST OF POLYLINE INDICES

Operating States: WSOP, WSAC, SGOP

Description

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

The list of polyline indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning polyline indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQEPLI (*workstation_id, element, error_status, num_line, relement*)

ginplineindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF POLYLINE INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of defined polyline index values for the specified workstation.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined polyline index values.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the polyline index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Workstation State List Inquiries

INQUIRE LIST OF POLYLINE INDICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF POLYMARKER INDICES

INQUIRE LIST OF POLYMARKER INDICES

Operating States: WSOP, WSAC, SGOP

Description

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

The list of polymarker indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning polymarker indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQEPMI (*workstation_id, element, error_status, num_mark, relement*)

ginqmarkerindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF POLYMARKER INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of defined polymarker index values for the specified workstation.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined polymarker index values.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the polymarker index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Workstation State List Inquiries

INQUIRE LIST OF POLYMARKER INDICES

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LIST OF TEXT INDICES

INQUIRE LIST OF TEXT INDICES

Operating States: WSOP, WSAC, SGOP

Description

The list of available text indexes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning text indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQETXI (*workstation_id, element, error_status, num_text, relement*)

ginqtextindices (*workstation_id, max_indices, start, indices, actual_indices, error_status*)

Workstation State List Inquiries

INQUIRE LIST OF TEXT INDICES

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_indexes

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of text index values for the specified workstation.

list_indexes

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined text index values.

Workstation State List Inquiries

INQUIRE LIST OF TEXT INDICES

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of indexes returned to the text index list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

INQUIRE LOCATOR DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE LOCATOR DEVICE STATE returns the initialization values for the specified locator logical input device, and the current input operating mode.

The locator logical input values are available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The locator logical input device is present on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

Syntax

GKS\$INQ_LOCATOR_STATE (*workstation_id, device_number, value_type, error_status, operating_mode, echo_flag, transformation_number, world_locator_x, world_locator_y, prompt_echo_type, echo_area,*

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

*data_record, record_buffer_length,
record_size)*

GQLCS (*workstation_id, device_number, type, dim_dr,
error_status, operating_mode, echo_flag, in_xform,
in_px, in_py, p_e_type, echo_area, len_dr, dr*)

ginqlocst (*workstation_id, device_number, workstation_type,
bufsize, state_size, state, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

Workstation State List Inquiries INQUIRE LOCATOR DEVICE STATE

| Value | Constant | Description |
|-------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

error_status

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

operating_mode

data type: **integer**
 access: **write-only**
 mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|-------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|-------|---------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

transformation_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the normalization transformation used to translate the initial input data point to device coordinates.

world_locator_x *world_locator_y*

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments designate the initial locator position, in world coordinates.

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing echo area device coordinate points in the following order XMIN, XMAX, YMIN, YMAX. For more

Workstation State List Inquiries INQUIRE LOCATOR DEVICE STATE

information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the current locator input data record for the specified device.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 38 | GKS\$_ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$_ERROR_140 | Specified input device is not present on workstation in routine **** |

Program Example

Example 11-9 illustrates the use of the function GKS\$INQ_LOCATOR_STATE.

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

Example 11-9: Determining the Current Locator State

```
C   This program writes the return values of the function
C   GKS$INQ_LOCATOR_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 1 ), PROMPT_ECHO_TYPE,
*   ERROR_STATUS, INPUT_MODE, ECHO_FLAG, TRANSFRM_NUMBER,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE, DEVICE_NUM
      REAL ECHO_AREA( 4 ), WORLD_X, WORLD_Y
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )

C   Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 4

C   You can obtain this information as long as the specified
C   workstation is open.
      CALL GKS$INQ_LOCATOR_STATE( WS_ID, DEVICE_NUM,
*   GKS$K_VALUE_REALIZED, ERROR_STATUS, INPUT_MODE,
*   ECHO_FLAG, TRANSFRM_NUMBER, WORLD_X, WORLD_Y,
*   PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE )

C   Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The input operating mode: ', INPUT_MODE
      WRITE(6, *) 'The echo flag: ', ECHO_FLAG
      WRITE(6, *) 'The transformation number: ',
*   TRANSFRM_NUMBER
      WRITE(6, *) 'The X world coordinate: ', WORLD_X
      WRITE(6, *) 'The Y world coordinate: ', WORLD_Y
      WRITE(6, *) 'The prompt and echo type: ',
*   PROMPT_ECHO_TYPE
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The data record: ', DATA_RECORD
      WRITE(6, *) 'The record buffer length: ',
*   RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The record size: ', RECORD_SIZE

      CALL GKS$CLOSE_WS( WS_ID )
      CALL GKS$CLOSE_GKS()
      END
```

Workstation State List Inquiries

INQUIRE LOCATOR DEVICE STATE

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_9 RETURN
$ LINK     EXAMPLE_9 RETURN
$ RUN      EXAMPLE_9 RETURN
The error status: 0
The input operating mode: 0
The echo flag: 1
The transformation number: 0
The X world coordinate: 0.5000000
The Y world coordinate: 0.5000000
The prompt and echo type: 1
The echo area: 0.0000000E+00 479.0000 0.0000000E+00 479.0000
The data record: 0
The record buffer length: 0
The record size: 0
$
```


Workstation State List Inquiries

INQUIRE PATTERN REPRESENTATION

INQUIRE PATTERN REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE PATTERN REPRESENTATION returns the values associated with a defined pattern index on a specified workstation.

The pattern representation is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.
- The pattern index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning patterns, refer to Chapter 5, Output Attribute Functions.

Syntax

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

Workstation State List Inquiries

INQUIRE PATTERN REPRESENTATION

GQPAR (*workstation_id, pindex, type, max_x_dim, max_y_dim, error_status, x_dim, y_dim, carry*)

ginqpatrep (*workstation_id, index, type, rep, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

pattern_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the defined pattern index on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

Workstation State List Inquiries

INQUIRE PATTERN REPRESENTATION

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

pattern_width ***pattern_height***

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the number of columns (width) and rows (height), within the color index array, for use when you create the pattern.

list_color_indexes

data type: **2-D array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the two-dimensional array containing the list of color indexes to use to create the pattern.

color_columns_return_size ***color_rows_return_size***

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the dimensions of the elements in the color array to which DEC GKS returned index values. You can use these values to traverse only the elements of the array that contain valid color index values.

Workstation State List Inquiries

INQUIRE PATTERN REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| -17 | DECGKS\$_ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 85 | GKS\$_ERROR_85 | Specified pattern index is invalid in routine **** |
| 88 | GKS\$_ERROR_88 | A representation for the specified pattern index has not been defined on this workstation in routine **** |
| 90 | GKS\$_ERROR_90 | Interior style PATTERN is not supported on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

INQUIRE PICK DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE PICK DEVICE STATE returns the initialization values for the specified pick logical input device, and the current input operating mode.

The pick logical input initialization values are available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The pick logical input device is present on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

NOTE

The DEC GKS FORTRAN binding does not return the data record for this function. This restriction conforms with the GKS Standard. Use the GKS\$ function with FORTRAN if you want the data record returned.

For more information concerning pick input, refer to Chapter 7, Input Functions.

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

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*)

GQPKS (*workstation_id, device_number, type, dim_dr, error_status, operating_mode, echo_flag, in_status, in_seg, in_pick_id, p_e_type, echo_area, len_dr, dr*)

ginqpickst (*workstation_id, device_number, type, bufsize, state_size, state, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

operating_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

| Value | Constant | Description |
|-------|-------------------------|------------------|
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|-------|---------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

initial_pick_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument determines whether the user can return a measure value of "No Pick." If the user can return No Pick, then the user can trigger the end of input without returning the value corresponding to the initial segment. This action returns the value 0 as the logical input device's measure. If the user cannot return No Pick, then triggering the end of input as soon as the initial picked segment is highlighted returns the identifier associated with that segment.

The defined values are as follows:

| Value | Constant | Description |
|-------|----------------------|-----------------------------|
| 1 | GKS\$K_STATUS_OK | Return the initial measure. |
| 2 | GKS\$K_STATUS_NOPICK | Return No Pick. |

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

initial_segment

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the name of the segment that is initially highlighted as soon as you request the pick logical input device.

initial_pick_id

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the pick identifier that is associated with a portion of the initially highlighted segment. For more information concerning pick identifiers, refer to Chapter 7, Input Functions, or Chapter 8, Segment Functions.

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument contains coordinate values in the following order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the current pick input data record for the specified device.

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -17 | DECGKS\$_ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |

Workstation State List Inquiries INQUIRE PICK DEVICE STATE

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 37 | GKS\$ _ERROR_37 | Specified workstation is not of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on workstation in routine **** |

Program Example

Example 11-10 illustrates the use of the function GKS\$INQ_PICK_STATE.

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

Example 11-10: Determining the Values Associated with the Current Pick State

```
C   This program writes the return values of the function
C   GKSSINQ_PICK_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, INITIAL_STATUS, SEGMENT,
* PICK_ID, PROMPT_ECHO_TYPE, ERROR_STATUS, DEVICE_NUM,
* INPUT_MODE, ECHO_FLAG, RECORD_BUFFER_LENGTH,
* RECORD_SIZE
      REAL ECHO_AREA(4), DATA_RECORD( 1 )

      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKSSOPEN_GKS( 'SYS$ERROR:' )
      CALL GKSSOPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )

C   Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 4

C   You can obtain this information as long as the specified
C   workstation is open.
      CALL GKSSINQ_PICK_STATE( WS_ID, DEVICE_NUM,
* GKS$K_VALUE_REALIZED, ERROR_STATUS, INPUT_MODE,
* ECHO_FLAG, INITIAL_STATUS, SEGMENT, PICK_ID,
* PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
* RECORD_BUFFER_LENGTH, RECORD_SIZE )

C   Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The input operating mode: ', INPUT_MODE
      WRITE(6, *) 'The echo flag: ', ECHO_FLAG
      WRITE(6, *) 'The initial pick status: ', INITIAL_STATUS
      WRITE(6, *) 'The picked segment identifier: ',
* SEGMENT
      WRITE(6, *) 'The initial pick identifier: ',
* PICK_ID
      WRITE(6, *) 'The prompt and echo type: ',
* PROMPT_ECHO_TYPE
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The data record: ', DATA_RECORD
```

(continued on next page)

Workstation State List Inquiries

INQUIRE PICK DEVICE STATE

Example 11-10 (Cont.): Determining the Values Associated with the Current Pick State

```
WRITE(6, *) 'The record buffer length: ',  
* RECORD_BUFFER_LENGTH  
WRITE(6, *) 'The record size: ', RECORD_SIZE  
  
CALL GKS$CLOSE_WS( WS_ID )  
CALL GKS$CLOSE_GKS()  
END
```

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_10 RETURN  
$ LINK EXAMPLE_10 RETURN  
$ RUN EXAMPLE_10 RETURN  
The error status: 0  
The input operating mode: 0  
The echo flag: 1  
The initial pick status: 2  
The picked segment identifier: 1  
The initial pick identifier: 1  
The prompt and echo type: 1  
The echo area: 0.000000E+00 479.0000 0.000000E+00 479.0000  
The data record: 4.790000  
The record buffer length: 4  
The record size: 4  
$
```

Workstation State List Inquiries

INQUIRE POLYLINE REPRESENTATION

INQUIRE POLYLINE REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

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

The polyline index values are available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.
- The polyline index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polyline attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQPLR (*workstation_id, pindex, type, error_status, ltype, lwidth, cindex*)

ginqlinerep (*workstation_id, index, type, rep, error_status*)

Workstation State List Inquiries

INQUIRE POLYLINE REPRESENTATION

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

polyline_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the defined polyline index on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

Workstation State List Inquiries

INQUIRE POLYLINE REPRESENTATION

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

line_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the line type associated with the specified polyline bundle index. The defined values are as follows:

| Value | Constant | Description |
|-------|-------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_LINETYPE_SOLID | Solid line |
| 2 | GKS\$K_LINETYPE_DASHED | Dashed line |
| 3 | GKS\$K_LINETYPE_DOTTED | Dotted line |
| 4 | GKS\$K_LINETYPE_DASHED_DOTTED | Solid line |
| >= 5 | | Reserved for future standardization |

line_width_scale_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the line width scale factor associated with the specified polyline bundle index. DEC GKS calculates line width by multiplying the scale factor by the nominal width.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the given polyline index value.

Workstation State List Inquiries

INQUIRE POLYLINE REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -17 | DECGKS\$ _ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 60 | GKS\$ _ERROR_60 | Polyline index is invalid in routine **** |
| 61 | GKS\$ _ERROR_61 | A representation for the specified polyline index has not been defined on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE POLYMARKER REPRESENTATION

INQUIRE POLYMARKER REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

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

The polymarker index values are available when DEC GKS is in any operating state except **GKS\$K_GKCL** or **GKS\$K_GKOP**, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is *not* of category **GKS\$K_WSCAT_MI**, **GKS\$K_WSCAT_INPUT**, or **GKS\$K_WSCAT_WISS**.
- The polymarker index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the polymarker attributes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQPMR (*workstation_id, pindex, type, error_status, mtype, msize, cindex*)

ginqmarkerrep (*workstation_id, index, type, rep, error_status*)

Workstation State List Inquiries

INQUIRE POLYMARKER REPRESENTATION

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

polymarker_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the defined polymarker index on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to

Workstation State List Inquiries

INQUIRE POLYMARKER REPRESENTATION

one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

marker_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the marker type associated with the specified polymarker bundle index value. The defined values are as follows:

| Value | Constant | Description |
|-------|----------------------------------|--|
| <= 0 | | Reserved for implementation-specific use |
| 1 | GKS\$K_MARKERTYPE_DOT | A dot (.) |
| 2 | GKS\$K_MARKERTYPE_PLUS | A plus sign (+) |
| 3 | GKS\$K_MARKERTYPE_ASTERISK | An asterisk (*) |
| 4 | GKS\$K_MARKERTYPE_CIRCLE | A circle (o) |
| 5 | GKS\$K_MARKERTYPE_DIAGONAL_CROSS | A cross (X) |
| >= 6 | | Reserved for future standardization |

marker_size_scale_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the marker size scale factor associated with the polymarker bundle index. DEC GKS calculates the marker size by multiplying the scale factor by the nominal size.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the specified polymarker index value.

Workstation State List Inquiries

INQUIRE POLYMARKER REPRESENTATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -17 | DECGKS\$ _ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 66 | GKS\$ _ERROR_66 | Polymarker index is invalid in routine **** |
| 67 | GKS\$ _ERROR_67 | A representation for the specified polymarker index has not been defined on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

Operating States: WSOP, WSAC, SGOP

Description

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

The list segment names are available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI or GKS\$K_WSCAT_INPUT.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQSGWK (*workstation_id, member, error_status, num_names, rmember*)

Workstation State List Inquiries

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

ginqsegnamesws (*workstation_id, max_segnames, start, segnames, actual_segnames, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_segment_names

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of defined segment names for the specified workstation.

list_segment_names

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing defined segment names.

Workstation State List Inquiries

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of names returned to the segment list. You can use this argument to see if you specified an array that was large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

INQUIRE STRING DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function **INQUIRE STRING DEVICE STATE** returns the initialization values for the specified string logical input device and the current input operating mode.

The string logical device state is available when DEC GKS is in any operating state except **GKS\$K_GKCL** or **GKS\$K_GKOP**, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category **GKS\$K_WSCAT_INPUT** or **GKS\$K_WSCAT_OUTIN**.
- The string logical input device is present on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the string logical input device, refer to Chapter 7, Input Functions.

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,*

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

*data_record, record_buffer_length,
record_size)*

GQSTS (*workstation_id, device_number, dim_dr, error_status,
operating_mode, echo_flag, num_chars, in_string,
p_e_type, echo_area, buf_size, i_cur_pos, len_dr, dr*)

ginqstringst (*workstation_id, device_number, bufsize, state_size,
state, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

operating_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|-------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|-------|---------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

default_string

data type: **string**
access: **write-only**
mechanism: **by descriptor**

This argument is the default input string value.

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

string_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the return size, in bytes, of the default string value.

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This 4-element array contains coordinate values in the order XMIN, XMAX, YMIN, YMAX. This argument is an array containing the device coordinate values that designate the input echo area on the workstation surface. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the current string input data record for the specified device.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on workstation in routine **** |

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

Program Example

Example 11-11 illustrates the use of the function INQUIRE STRING DEVICE STATE.

Example 11-11: Determining the Initial String Logical Input Device Values

```
C   This program writes the return values of the function
C   GKS$INQ_STRING_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 2 ), PROMPT_ECHO_TYPE,
*   ERROR_STATUS, INPUT_MODE, ECHO_FLAG,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE, INPUT_STATUS,
*   DEVICE_NUM, STRING_SIZE
      REAL ECHO_AREA( 4 )
      CHARACTER*80 INITIAL_STRING
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )

C   Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 8

C   You can obtain this information as long as the specified
C   workstation is open.
      CALL GKS$INQ_STRING_STATE( WS_ID, DEVICE_NUM,
*   ERROR_STATUS, INPUT_MODE,
*   ECHO_FLAG, %DESCR( INITIAL_STRING ), STRING_SIZE,
*   PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE )

C   Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The input mode: ', INPUT_MODE
      WRITE(6, *) 'The echo flag: ', ECHO_FLAG
      WRITE(6, *) 'The initial string: ', INITIAL_STRING
      WRITE(6, *) 'The initial string size: ', STRING_SIZE
      WRITE(6, *) 'The prompt and echo type: ',
*   PROMPT_ECHO_TYPE
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The data record: ', DATA_RECORD
```

(continued on next page)

Workstation State List Inquiries

INQUIRE STRING DEVICE STATE

Example 11-11 (Cont.): Determining the Initial String Logical Input Device Values

```
WRITE(6, *) 'The record buffer length: ',  
* RECORD_BUFFER_LENGTH  
WRITE(6, *) 'The record size: ', RECORD_SIZE  
  
CALL GKS$CLOSE_WS( WS_ID )  
CALL GKS$CLOSE_GKS()  
END
```

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_11 RETURN  
$ LINK     EXAMPLE_11 RETURN  
$ RUN     EXAMPLE_11 RETURN  
The error status:          0  
The input mode:           0  
The echo flag:            1  
The initial string:  
  
The initial string size:          0  
The prompt and echo type:        1  
The echo area: 533.0000 799.0000 0.0000000E+00 479.0000  
The data record:                20 0  
The record buffer length:        8  
The record size:                 8  
$
```

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

INQUIRE STROKE DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE STROKE DEVICE STATE returns the initialization values for the specified stroke logical input device and the current input operating mode.

The stroke logical device state is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation open.
- The workstation is of category GKS\$K_WSCAT_INPUT or GKS\$K_WSCAT_OUTIN.
- The stroke logical input device is present on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the stroke logical input device, refer to Chapter 7, Input Functions.

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,*

Workstation State List Inquiries INQUIRE STROKE DEVICE STATE

*echo_area, data_record,
record_buffer_length, record_size)*

GQSKS (*workstation_id, device_number, type, max_pts, dim_dr,
error_status, operating_mode, echo_flag, xform,
num_pts, px, py, p_e_type, echo_area, buf_size, len_dr,
dr*)

gingstrokest (*workstation_id, device_number, type, bufsize,
state_size, state, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

num_elements

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument contains the number of elements in the declared array buffer. On output, this argument contains the number of elements containing returned stroke points.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

operating_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

transformation_number

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the normalization transformation number used to translate the points in the initial stroke from world coordinates to device coordinates. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

total_points

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total number of world coordinate points in the initial stroke. If *total_points* is more than *num_elements*, DEC GKS truncated the stroke point list so that it fits into your declared buffer.

world_x_points

world_y_points

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

These arguments are the world coordinate points that comprise the initial stroke.

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing echo area device coordinate points in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the current stroke input data record for the specified device.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -17 | DECGKS\$ _ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on workstation in routine **** |

Program Example

Example 11-12 illustrates the use of the function INQUIRE STROKE DEVICE STATE.

Workstation State List Inquiries INQUIRE STROKE DEVICE STATE

Example 11-12: Determining the Initial Stroke Logical Input Device Values

```
C   This program writes the return values of the function
C   GKS$INQ_STROKE_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, DATA_RECORD( 13 ), BUFFER_SIZE,
*   DIMENSION, PROMPT_ECHO_TYPE, ERROR_STATUS,
*   TRANSFRM, NUM_POINTS, INPUT_MODE, ECHO_FLAG,
*   INPUT_STATUS, DEVICE_NUM, RET_SIZE_X, RET_SIZE_Y,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE, EDIT_POSITION,
*   ATT_FLAG
      REAL ECHO_AREA( 4 ), STROKE_X( 5 ),
*   STROKE_Y( 5 ), X_INT, Y_INT, TIME_INT
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /, DIMENSION / 5 /

C   Clarify the components of the data record...
      EQUIVALENCE( DATA_RECORD( 1 ), BUFFER_SIZE)
      EQUIVALENCE( DATA_RECORD( 2 ), EDIT_POSITION)
      EQUIVALENCE( DATA_RECORD( 3 ), X_INT)
      EQUIVALENCE( DATA_RECORD( 4 ), Y_INT)
      EQUIVALENCE( DATA_RECORD( 5 ), TIME_INT)
      EQUIVALENCE( DATA_RECORD( 6 ), ATT_FLAG)

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )
C   Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 52

C   You can obtain this information as long as the specified
C   workstation is open.
      CALL GKS$INQ_STROKE_STATE( WS_ID, DEVICE_NUM,
*   GKS$K_VALUE_REALIZED, DIMENSION, ERROR_STATUS,
*   INPUT_MODE, ECHO_FLAG, TRANSFRM, NUM_POINTS, STROKE_X,
*   STROKE_Y, PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
*   RECORD_BUFFER_LENGTH, RECORD_SIZE )
```

(continued on next page)

Workstation State List Inquiries

INQUIRE STROKE DEVICE STATE

Example 11-12 (Cont.): Determining the Initial Stroke Logical Input Device Values

```
C      Write the returned values to the screen.
WRITE(6, *) 'The error status: ', ERROR_STATUS
WRITE(6, *) 'The input mode: ', INPUT_MODE
WRITE(6, *) 'The echo flag: ', ECHO_FLAG
WRITE(6, *) 'The transformation number: ', TRANSFORM
WRITE(6, *) 'The number of points: ', NUM_POINTS
WRITE(6, *) 'The X values of the initial'
WRITE(6, *) 'stroke: ', STROKE_X
WRITE(6, *) 'The Y values of the initial'
WRITE(6, *) 'stroke: ', STROKE_Y
WRITE(6, *) 'The prompt and echo type: ',
* PROMPT_ECHO_TYPE
WRITE(6, *) 'The echo area: ', ECHO_AREA
WRITE(6, *) 'The data record: ', DATA_RECORD
WRITE(6, *) 'The maximum data length: ',
* RECORD_BUFFER_LENGTH
WRITE(6, *) 'The data return size: ',
* RECORD_SIZE

CALL GKSSCLOSE_WS( WS_ID )
CALL GKSSCLOSE_GKS()
END
```

Workstation State List Inquiries INQUIRE STROKE DEVICE STATE

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_12 RETURN
$ LINK     EXAMPLE_12 RETURN
$ RUN      EXAMPLE_12 RETURN
The error status:          0
The input mode:           0
The echo flag:            1
The transformation number:          0
The number of points:      0
The X values of the initial
stroke:  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
0.0000000E+00
The Y values of the initial
stroke:  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
0.0000000E+00
The prompt and echo type:          1
The echo area:  0.0000000E+00  479.0000  0.0000000E+00  479.0000
The data record:          80          0          0          0          0
0          0          0          0          0
The maximum data length:          20
The data return size:          20
$
```

Workstation State List Inquiries

INQUIRE TEXT EXTENT

INQUIRE TEXT EXTENT

Operating States: WSOP, WSAC, SGOP

Description

The text extent information is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.
- The string is valid.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning text attributes, refer to Chapter 5, Output Attribute Functions.

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*)

GQTX (*workstation_id, px, py, cstring, error_status, con_pt_x, con_pt_y, ext_x, ext_y*)

Workstation State List Inquiries

INQUIRE TEXT EXTENT

ginqtextextent (*workstation_id, position, string, extent, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

string_position_x ***string_position_y***

data type: **real**
access: **read-only**
mechanism: **by reference**

These arguments are the X and Y world coordinate points that designate the starting point of the specified string.

string

data type: **string**
access: **read-only**
mechanism: **by descriptor**

This argument is the output text string about which you need information.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation State List Inquiries

INQUIRE TEXT EXTENT

concatenation_x
concatenation_y

data type: **real**
access: **write-only**
mechanism: **by reference**

These arguments are the X and Y world coordinate points that you can use as a starting point for a new output string or as a concatenation point at the end of the specified string.

extent_rectangle_x
extent_rectangle_y

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

These arguments are 4-element arrays containing the four world coordinate X and Y values comprising the text extent rectangle. Point order starts with the lower left corner and moves in a counter-clockwise direction. DEC GKS computes the text extent rectangle using the current values for the text font and precision, the character expansion factor, the character-up vector, the character spacing, text path, text alignment, and character width. The extent rectangle encloses the character bodies of the specified string.

Workstation State List Inquiries

INQUIRE TEXT EXTENT

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 38 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 101 | GKS\$ _ERROR_101 | Invalid code in string in routine **** |

Workstation State List Inquiries

INQUIRE TEXT REPRESENTATION

INQUIRE TEXT REPRESENTATION

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE TEXT REPRESENTATION returns the values currently associated with the specified text index value.

The current text representation values are available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.
- The text index is valid and defined.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning text indexes, refer to Chapter 5, Output Attribute Functions.

Syntax

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

GQTXR (*workstation_id, tindex, type, error_status, font, precision, ex_fac, spacing, cindex*)

Workstation State List Inquiries

INQUIRE TEXT REPRESENTATION

ginqtextrep (*workstation_id, index, type, rep, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

text_index

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the defined text index on the specified workstation.

value_type

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument specifies the type of values you want this function to return. This function either returns the exact workstation state list values as they are set, or it returns the values that the DEC GKS device handler is capable of implementing. (See Section 11.1.2 for more information concerning this argument.) The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------------|--|
| 0 | GKS\$K_VALUE_SET | Use the exact state list values. |
| 1 | GKS\$K_VALUE_REALIZED | Use the values approximated by the graphics handler. |

Workstation State List Inquiries

INQUIRE TEXT REPRESENTATION

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

text_font

text_precision

data type: **integer**
access: **write-only**
mechanism: **by reference**

The first argument is the current hardware or software font number associated with the specified text bundle index. For information concerning the hardware fonts available on your workstation, refer to the appropriate device-specific appendix in this manual. For more information concerning the software fonts available, refer to the appropriate appendix in this manual.

The second argument is the current text precision associated with the specified text bundle index. The defined values are as follows:

| Value | Constant | Description |
|--------------|------------------------------|---------------------|
| 0 | GKS\$K_TEXT_PRECISION_STRING | String precision |
| 1 | GKS\$K_TEXT_PRECISION_CHAR | Character precision |
| 2 | GKS\$K_TEXT_PRECISION_STROKE | Stroke precision |

character_expansion_factor

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character expansion factor associated with the specified text bundle index. The character expansion factor multiplied by the width-to-height ratio in the original font design determines the character

Workstation State List Inquiries

INQUIRE TEXT REPRESENTATION

width. The character expansion factor does not affect the height of the characters.

character_spacing

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the current character spacing associated with the specified text bundle index. Positive values increase the space between characters. Negative values decrease the space between characters. The value 0 places the character bodies adjacent to one another.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index associated with the specified text index value.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -17 | DECGKS\$_ERROR_NEG_17 | Inquired device values not set or realized in routine **** |
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

Workstation State List Inquiries

INQUIRE TEXT REPRESENTATION

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 72 | GKS\$_ERROR_72 | Text index is invalid in routine **** |
| 73 | GKS\$_ERROR_73 | A representation for the specified text index has not been defined on this workstation in routine **** |

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

INQUIRE VALUATOR DEVICE STATE

Operating States: WSOP, WSAC, SGOP

Description

The function `INQUIRE VALUATOR DEVICE STATE` returns the initialization values for the specified valuator logical input device, and the current input operating mode.

The valuator device state is available when DEC GKS is in any operating state except `GKS$K_GKCL` or `GKS$K_GKOP`, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category `GKS$K_WSCAT_OUTPUT` or `GKS$K_WSCAT_OUTIN`.
- The valuator logical input device is present on the specified workstation.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning input, refer to Chapter 7, Input Functions.

Syntax

`GKS$INQ_VALUATOR_STATE` (*workstation_id, device_number, error_status, operating_mode, echo_flag, default_value, prompt_echo_type, echo_area, data_record,*

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

record_buffer_length,
record_size)

GQVLS (*workstation_id, dev_num, dim_dr, error_status,*
operating_mode, echo_flag, in_value, p_e_type,
echo_area, low_val, high_val,
len_dr, dr)

ginqvalst (*workstation_id, device_number, bufsize, state_size,*
state, error_status)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

device_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the device number that differentiates between logical input devices of the same class, operating on the same workstation. For more information, refer to Chapter 7, Input Functions.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

operating_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current input operating mode for the specified logical input device. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------------|--------------------|
| 0 | GKS\$K_INPUT_MODE_REQUEST | Request input mode |
| 1 | GKS\$K_INPUT_MODE_SAMPLE | Sample input mode |
| 2 | GKS\$K_INPUT_MODE_EVENT | Event input mode |

For more information concerning the input operating modes, refer to Chapter 7, Input Functions.

echo_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the echo flag specifying whether input is echoed on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|--------------------|
| 0 | GKS\$K_NOECHO | Do not echo input. |
| 1 | GKS\$K_ECHO | Echo input. |

default_value

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument is the default real value of the valuator input device.

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

prompt_echo_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current prompt and echo type value.

echo_area

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 4-element array containing the echo area device coordinate points in the order XMIN, XMAX, YMIN, YMAX. For more information concerning the DEC GKS coordinate systems, refer to Chapter 6, Transformation Functions.

data_record

data type: **address (record)**
access: **write-only**
mechanism: **by reference**

This argument is a pointer to the current valuator input data record for the specified device.

record_buffer_length

data type: **integer**
access: **modifiable**
mechanism: **by reference**

On input, this argument should contain the size, in bytes, of the data record buffer you passed as the argument *data_record*. On output, the graphics handler writes the amount of the buffer, in bytes, filled by the written data record. If the argument *record_size* is larger than *record_buffer_length* after the function call, then you know that the graphics handler truncated the data record when writing it to the buffer and data was lost.

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

record_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the total size, in bytes, of the data record.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 38 | GKS\$ _ERROR_38 | Specified workstation is neither of category INPUT nor of category OUTIN in routine **** |
| 140 | GKS\$ _ERROR_140 | Specified input device is not present on workstation in routine **** |

Program Example

Example 11-13 illustrates the use of the function INQUIRE VALUATOR DEVICE STATE.

Workstation State List Inquiries

INQUIRE VALUATOR DEVICE STATE

Example 11-13: Determining the Current Valuator State

```
C      This program writes the return values of the function
C      GKS$INQ_VALUATOR_STATE to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, PROMPT_ECHO_TYPE, ERROR_STATUS,
* INPUT_MODE, ECHO_FLAG, INPUT_STATUS, DEVICE_NUM,
* RECORD_BUFFER_LENGTH, RECORD_SIZE
      REAL ECHO_AREA( 4 ), DATA_RECORD( 2 ), UPPER_LIMIT,
* LOWER_LIMIT, VALUE
      DATA WS_ID / 1 /, DEVICE_NUM / 1 /

C      The elements in the data record are the upper and lower limits.
      EQUIVALENCE( DATA_RECORD( 1 ), LOWER_LIMIT )
      EQUIVALENCE( DATA_RECORD( 2 ), UPPER_LIMIT )

      CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
      CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )

C      Initialize the modifiable argument...
      RECORD_BUFFER_LENGTH = 8

C      You can obtain this information as long as the specified
C      workstation is open.
      CALL GKS$INQ_VALUATOR_STATE( WS_ID, DEVICE_NUM,
* ERROR_STATUS, INPUT_MODE, ECHO_FLAG, VALUE,
* PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,
* RECORD_BUFFER_LENGTH, RECORD_SIZE )

C      Write the returned values to the screen.
      WRITE(6, *) 'The error status: ', ERROR_STATUS
      WRITE(6, *) 'The input operating mode: ', INPUT_MODE
      WRITE(6, *) 'The echo flag: ', ECHO_FLAG
      WRITE(6, *) 'The initial value: ', VALUE
      WRITE(6, *) 'The prompt and echo type: ',
* PROMPT_ECHO_TYPE
      WRITE(6, *) 'The echo area: ', ECHO_AREA
      WRITE(6, *) 'The data record: ', DATA_RECORD
      WRITE(6, *) 'The maximum data length: ',
* RECORD_BUFFER_LENGTH
      WRITE(6, *) 'The return size: ', RECORD_SIZE

      CALL GKS$CLOSE_WS( WS_ID )
      CALL GKS$CLOSE_GKS()
      END
```

Workstation State List Inquiries INQUIRE VALUATOR DEVICE STATE

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN EXAMPLE_13 RETURN
$ LINK     EXAMPLE_13 RETURN
$ RUN      EXAMPLE_13 RETURN
The error status:          0
The input operating mode:          0
The echo flag:              1
The initial value:    0.5000000
The prompt and echo type:          1
The echo area:    533.0000      799.0000      0.0000000E+00      479.0000
The data record:    0.0000000E+00      1.000000
The maximum data length:          8
The return size:          8
$
```

Workstation State List Inquiries

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES returns the current deferral state, implicit regeneration mode, workstation surface status, and whether a new frame is necessary to update the screen.

The deferral and update information is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI, GKS\$K_WSCAT_INPUT, or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning surface update or operating states, refer to Chapter 3, Control Functions.

Syntax

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

Workstation State List Inquiries

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

GQWKDU (*workstation_id, error_status, def_mode, reg_mode, surface, new_frame*)

ginqwsdeferupdatest (*workstation_id, du, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

deferral_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current deferral mode associated with the specified workstation. The defined values are as follows:

| Value | Constant | Description |
|--------------|-----------------|---|
| 0 | GKS\$K_ASAP | Generate images as soon as possible. |
| 1 | GKS\$K_BNIG | Generate images before input is requested globally. |

Workstation State List Inquiries

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

| Value | Constant | Description |
|-------|-------------|--|
| 2 | GKS\$K_BNIL | Generate images before input is requested locally. |
| 3 | GKS\$K_ASTI | Generate images some time. Exact time is not guaranteed. |

regeneration_mode

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the current implicit regeneration mode associated with the specified workstation. The defined values are as follows:

| Value | Constant | Description |
|-------|-----------------------|-----------------------------------|
| 0 | GKS\$K_IRG_SUPPRESSED | Image regeneration is suppressed. |
| 1 | GKS\$K_IRG_ALLOWED | Image regeneration is allowed. |

surface_empty

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that specifies whether the workstation surface is empty (refer to Chapter 3, Control Functions). The defined values are as follows:

| Value | Constant | Description |
|-------|-----------------|-------------------------|
| 0 | GKS\$K_EMPTY | Surface is "empty." |
| 1 | GKS\$K_NOTEMPTY | Surface is "not empty." |

Workstation State List Inquiries

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

new_frame_necessary

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that specifies whether DEC GKS needs to clear the surface before making the next update to the screen. The defined values are as follows:

| Value | Constant | Description |
|--------------|------------------------------|--------------------------------------|
| 0 | GKS\$K_NEWFRAME_NOTNECESSARY | Do not clear surface at next update. |
| 1 | GKS\$K_NEWFRAME_NECESSARY | Clear the surface at next update. |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |

Workstation State List Inquiries

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Workstation State List Inquiries

INQUIRE WORKSTATION CONNECTION AND TYPE

INQUIRE WORKSTATION CONNECTION AND TYPE

Operating States: WSOP, WSAC, SGOP

Description

The function **INQUIRE WORKSTATION CONNECTION AND TYPE** returns the logical name associated with the physical device connection running from the host computer to the workstation, and returns the type of workstation with which you are working.

The workstation connection and type are available when DEC GKS is in any operating state except **GKS\$K_GKCL** or **GKS\$K_GKOP**, if the specified workstation identifier is valid, and if the associated workstation is open.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning workstation connections, workstation types, and operating states, refer to Chapter 3, Control Functions.

Syntax

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

GQWKC (*workstation_id, error_status, con_id, workstation_type*)

ginqwsconntype (*workstation_id, bufsize, ct_size, ct, error_status*)

Workstation State List Inquiries

INQUIRE WORKSTATION CONNECTION AND TYPE

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

connection_logical_name

data type: **string**
access: **write-only**
mechanism: **by descriptor**

This argument is the logical name associated with the physical device connection running from the host computer to the workstation.

workstation_type

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the integer value that is associated with the open workstation. For the list of all DEC GKS valid workstation types, refer to the appropriate appendix in this manual.

Workstation State List Inquiries

INQUIRE WORKSTATION CONNECTION AND TYPE

logical_return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the return size, in bytes, of the string specifying the connection logical name.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| -68 | DECGKS\$ _ERROR_NEG_68 | Invalid descriptor **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |

Workstation State List Inquiries

INQUIRE WORKSTATION STATE

INQUIRE WORKSTATION STATE

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE WORKSTATION STATE returns the state of the active or inactive workstation.

The state of the workstation is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI or GKS\$K_WSCAT_INPUT.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning open workstations or operating states, refer to Chapter 3, Control Functions.

Syntax

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

GQWKS (*workstation_id*, *error_status*, *state*)

ginqwsst (*workstation_id*, *state*, *error_status*)

Workstation State List Inquiries

INQUIRE WORKSTATION STATE

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

workstation_state

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument specifies whether the currently open workstation is active. The defined values are as follows:

| Value | Constant | Description |
|--------------|--------------------|----------------------------|
| 0 | GKS\$K_WS_INACTIVE | Workstation is not active. |
| 1 | GKS\$K_WS_ACTIVE | Workstation is active. |

Workstation State List Inquiries

INQUIRE WORKSTATION STATE

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |

Workstation State List Inquiries

INQUIRE WORKSTATION TRANSFORMATION

INQUIRE WORKSTATION TRANSFORMATION

Operating States: WSOP, WSAC, SGOP

Description

The function INQUIRE WORKSTATION TRANSFORMATION returns the flag that determines whether or not a workstation transformation is pending, the current workstation window and viewport, and the pending workstation window and viewport.

The workstation transformation information is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is *not* of category GKS\$K_WSCAT_MI or GKS\$K_WSCAT_WISS.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning workstation transformations, refer to Chapter 6, Transformation Functions.

Syntax

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

GQWKT (*workstation_id, error_status, state, r_win, c_win, r_view, c_view*)

Workstation State List Inquiries

INQUIRE WORKSTATION TRANSFORMATION

ginqwstran (*workstation_id, wstran, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

transformation_pending

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that designates whether a workstation transformation is pending. The defined values are as follows:

| Value | Constant | Description |
|--------------|-------------------|--|
| 0 | GKS\$K_NOTPENDING | A workstation transformation is not pending. |
| 1 | GKS\$K_PENDING | A workstation transformation is pending. |

Workstation State List Inquiries

INQUIRE WORKSTATION TRANSFORMATION

requested_window
current_window

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

These arguments are 4-element arrays that contain the requested and current workstation window dimensions, in normalized device coordinates. DEC GKS stores the dimensions in the following order:

1. X minimum value
2. X maximum value
3. Y minimum value
4. Y maximum value

requested_viewport
current_viewport

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

These arguments are 4-element arrays that contain the requested and current workstation viewport dimensions, in device coordinates. GKS stores the dimensions in the following order:

1. X minimum value
2. X maximum value
3. Y minimum value
4. Y maximum value

Workstation State List Inquiries

INQUIRE WORKSTATION TRANSFORMATION

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Segment State List Inquiries

This section describes the segment state list inquiries. (For more information concerning the segment state list, refer to Chapter 3, Control Functions, and to Chapter 8, Segment Functions.) You use these functions if you need information about the state of a single segment, which is identified by a numeric segment name, or if you are not aware of the current list of segment attributes or the set of workstations associated with a segment.

Segment State List Inquiries

INQUIRE SEGMENT ATTRIBUTES

INQUIRE SEGMENT ATTRIBUTES

Operating States: WSOP, WSAC, SGOP

Description

The function INQ SEGMENT ATTRIBUTES returns the segment transformation matrix, visibility, highlighting, priority, and detectability.

The list of segment attributes is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the segment exists and its name is valid. If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQSGA (*segment_name, error_status, matrix, visible, highlight, priority, detect*)

ginqsegattr (*segment_name segattr, error_status*)

Arguments

segment_name

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an existing segment.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

transformation_matrix

data type: **array (real)**
access: **write-only**
mechanism: **by reference**

This argument is a 6-element array containing the translation, scaling, and rotation components of the segment transformation matrix. For more information concerning the transformation matrix, refer to ACCUMULATE TRANSFORMATION MATRIX, EVALUATE TRANSFORMATION MATRIX, and SET SEGMENT TRANSFORMATION in Chapter 8, Segment Functions.

Segment State List Inquiries

INQUIRE SEGMENT ATTRIBUTES

visibility

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the segment's visibility on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|--------------|------------------|--|
| 0 | GKS\$K_INVISIBLE | The segment is not visible on the surface. |
| 1 | GKS\$K_VISIBLE | The segment is visible on the surface. |

highlighting

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument specifies whether GKS highlights the specified segment on the workstation surface. The defined values are as follows:

| Value | Constant | Description |
|--------------|--------------------|--|
| 0 | GKS\$K_NORMAL | The segment is not highlighted on the surface. |
| 1 | GKS\$K_HIGHLIGHTED | The segment is highlighted on the surface. |

priority

data type: **real**
access: **write-only**
mechanism: **by reference**

This argument specifies the priority of the specified segment. DEC GKS checks the priority of a segment when two segments overlap on the workstation surface, for all hardware devices that support this feature. Segment priorities range from 0.0 to 1.0, and each device supports a finite number of priorities (for more information, refer to the device-specific appendix in this manual).

Segment State List Inquiries INQUIRE SEGMENT ATTRIBUTES

detectability

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument determines whether the specified segment is detectable during pick input. The defined values are as follows:

| Value | Constant | Description |
|--------------|---------------------|-------------------------------|
| 0 | GKS\$K_UNDETECTABLE | You cannot pick this segment. |
| 1 | GKS\$K_DETECTABLE | You can pick this segment. |

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 120 | GKS\$_ERROR_120 | Specified segment name is invalid in routine **** |
| 122 | GKS\$_ERROR_122 | Specified segment does not exist in routine **** |

Segment State List Inquiries

INQUIRE SET OF ASSOCIATED WORKSTATIONS

INQUIRE SET OF ASSOCIATED WORKSTATIONS

Operating States: WSOP, WSAC, SGOP

Description

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

The list of associated workstations is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the segment exists and its name is valid. If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning segments, refer to Chapter 8, Segment Functions.

Syntax

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

GQASWK (*segment_name, member, error_status, num_ws, rmember*)

ginqassocws (*segment_name, max, start, actual, assocws, error_status*)

Segment State List Inquiries

INQUIRE SET OF ASSOCIATED WORKSTATIONS

Arguments

segment_name

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an existing segment.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

num_workstations

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstations associated with the specified segment.

list_workstations

data type: **array (integer)**
access: **write-only**
mechanism: **by descriptor**

This argument is the array containing the workstation identifiers corresponding to all the workstations associated with the specified segment.

return_size

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the number of workstation identifiers returned to the workstation list. You can use this argument to see if you specified an array

Segment State List Inquiries

INQUIRE SET OF ASSOCIATED WORKSTATIONS

that was large enough to hold all the returned values.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 120 | GKS\$ _ERROR_120 | Specified segment name is invalid in routine **** |
| 122 | GKS\$ _ERROR_122 | Specified segment does not exist in routine **** |

Pixel Inquiries

This section describes the pixel inquiries. Pixel inquiries return the color of an individual pixel or the color of a rectangular region of pixels on the device that supports this type of graphic output. These functions can be used to check a rectangular cell array region currently displayed on the workstation surface.

Pixel Inquiries

INQUIRE PIXEL

INQUIRE PIXEL

Operating States: WSOP, WSAC, SGOP

Description

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

The color of a pixel is available when DEC GKS is in any operating state except `GKS$K_GKCL` or `GKS$K_GKOP`, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category `GKS$K_WSCAT_OUTPUT` or `GKS$K_WSCAT_OUTIN`.
- The workstation has the ability to return information about pixels.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the capabilities of a workstation type, refer to the device-specific appendix in this manual.

Syntax

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

`GQPX` (*workstation_id*, *px*, *py*, *error_status*, *cindex*)

`ginqpixel` (*workstation_id*, *ppoint*, *pix*, *error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

world_x

world_y

data type: **real**
access: **read-only**
mechanism: **by reference**

These arguments are the X and Y world coordinates of the pixel about which you are inquiring.

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

color_index

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the color index corresponding to the color of the specified device coordinate. If the device coordinate does not translate to a valid pixel on the display surface, DEC GKS returns the value -1 to this argument to signal an invalid coordinate.

Pixel Inquiries

INQUIRE PIXEL

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 40 | GKS\$ _ERROR_40 | Specified workstation has no pixel store readback capability in routine **** |

INQUIRE PIXEL ARRAY

Operating States: WSOP, WSAC, SGOP

Description

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

DEC GKS determines the starting point within the color index array, determines the number of remaining elements, and then maps the remaining columns and rows, one for one, onto a rectangular portion of pixels on the display screen.

Next, DEC GKS translates a row of pixels to color indexes, fills the first dimension of the remaining array elements with the translated index values, and continues until all pixels are translated and the color index is full. (The first “dimension” of the array is either the row or the column, depending on whether your programming language supports row-major or column-major arrays.)

The list of color indexes corresponding to a pixel array is available when DEC GKS is in any operating state except GKS\$K_GKCL or GKS\$K_GKOP, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category GKS\$K_WSCAT_OUTPUT or GKS\$K_WSCAT_OUTIN.
- The workstation has the ability to return information about pixels.
- The dimensions specified for the color array are valid.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

Pixel Inquiries

INQUIRE PIXEL ARRAY

For more information concerning column-major arrays, row-major arrays, and color index arrays, refer to CELL ARRAY in Chapter 4, Output Functions. For more information concerning the capabilities of your device, refer to the device-specific appendix in this manual.

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*)

GQPXA (*workstation_id, corner_x, corner_y, dim_x, dim_y, scol, srow, pcols, prows, error_status, in_vals, carray*)

ginqpixelarray (*workstation_id, point, dimen, bufsize, covalid, pxarray, actual_size, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

column_number

row_number

data type: **integer**
access: **read-only**
mechanism: **by reference**

These arguments are the numbers of the column and row that designate the starting element in the color index array. DEC GKS begins placing color index values at this array element.

Pixel Inquiries INQUIRE PIXEL ARRAY

max_columns *max_rows*

data type: **integer**
access: **read-only**
mechanism: **by reference**

These arguments specify the numbers of columns and rows of pixels about which you inquire. The values must be less than or equal to the size of the buffer, from *column_number* and *row_number* to the last element of *color_index_array*.

world_x *world_y*

data type: **real**
access: **read-only**
mechanism: **by reference**

These arguments are the values specifying the upper left corner of the pixel array to be translated to color index values. You pass these arguments as a world coordinate value, and DEC GKS translates the point to device coordinates according to the current normalization and workstation transformations. (For more information concerning transformations, refer to Chapter 6, Transformation Functions.)

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

invalid_indexes_flag

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the flag that specifies whether there exist any invalid color index values. (GKS returns an invalid index value of -1 if a pixel is outside

Pixel Inquiries

INQUIRE PIXEL ARRAY

the display surface, possibly due to a transformation). The defined values are as follows:

| Value | Constant | Description |
|-------|------------------------|--|
| 0 | GKS\$K_INVALID_ABSENT | Color array contains no invalid indexes. |
| 1 | GKS\$K_INVALID_PRESENT | Color array contains invalid indexes. |

color_index_array

data type: **2-D array (integer)**

access: **write-only**

mechanism: **by descriptor**

This argument is the two-dimensional color index array. If DEC GKS cannot translate a pixel color to a color index value, DEC GKS fills the array element with the value -1.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|--------------|------------------------|--|
| -19 | DECGKS\$_ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$_ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |

Pixel Inquiries INQUIRE PIXEL ARRAY

| Error Number | Completion Status Code | Message |
|-------------------------|-------------------------------|---|
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 39 | GKS\$ _ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 40 | GKS\$ _ERROR_40 | Specified workstation has no pixel store readback capability in routine **** |
| 91 | GKS\$ _ERROR_91 | Dimensions of color array are invalid in routine **** |

Pixel Inquiries

INQUIRE PIXEL ARRAY DIMENSIONS

INQUIRE PIXEL ARRAY DIMENSIONS

Operating States: WSOP, WSAC, SGOP

Description

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

The dimensions of a pixel array are available when DEC GKS is in any operating state except **GKS\$K_GKCL** or **GKS\$K_GKOP**, and if the following conditions exist:

- The specified workstation identifier is valid and the associated workstation is open.
- The workstation is of category **GKS\$K_WSCAT_OUTPUT** or **GKS\$K_WSCAT_OUTIN**.

If these conditions are not met, the output arguments are undefined, and the function sets the error status argument to the number of one of the errors listed in the Error Messages section.

For more information concerning the dimensions of your workstation surface, refer to the device-specific appendix in this manual.

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*)

Pixel Inquiries INQUIRE PIXEL ARRAY DIMENSIONS

GQPXAD (*workstation_id, px, py, dx, dy, error_status, pa_cols, pa_rows*)

ginqpixelarraydim (*workstation_id, rect, dim, error_status*)

Arguments

workstation_id

data type: **integer**
access: **read-only**
mechanism: **by reference**

This argument is the integer value that identifies an open workstation.

starting_point_x ***starting_point_y***

data type: **real**
access: **read-only**
mechanism: **by reference**

These arguments are the X and Y values designating a corner of a rectangular area to be mapped onto the display surface. You pass these arguments as world coordinate values, and DEC GKS translates the point to device coordinates according to the current normalization and workstation transformations. (For more information concerning transformations, refer to Chapter 6, Transformation Functions.)

diagonal_point_x ***diagonal_point_y***

data type: **real**
access: **read-only**
mechanism: **by reference**

These arguments are the X and Y values of the point diagonal to the starting point that form the rectangle to be mapped onto the display surface. You pass these arguments as world coordinate values, and DEC GKS translates the point to device coordinates according to the current normalization and workstation transformations. (For more information concerning transformations, refer to Chapter 6, Transformation Functions.)

Pixel Inquiries

INQUIRE PIXEL ARRAY DIMENSIONS

error_status

data type: **integer**
access: **write-only**
mechanism: **by reference**

This argument is the error indicator. If the function writes the value 0 to this argument, all the remaining output arguments are valid. If the function writes any other number to this argument, the number corresponds to one of the error messages listed in the Error Messages section, and all the remaining output arguments are invalid.

dimension_device_x *dimension_device_y*

data type: **integer**
access: **write-only**
mechanism: **by reference**

These arguments are the dimensions of the pixel array.

Error Messages

If this inquiry function cannot return valid values, the number in the error status argument corresponds to one of the numbers in the following list:

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|--|
| -19 | DECGKS\$ _ERROR_NEG_19 | Invalid error status parameter specified in routine **** |
| -20 | DECGKS\$ _ERROR_NEG_20 | GKS not in proper state: GKS in the error state in routine **** |
| 7 | GKS\$ _ERROR_7 | GKS not in proper state; GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |

Pixel Inquiries

INQUIRE PIXEL ARRAY DIMENSIONS

| Error Number | Completion Status Code | Message |
|---------------------|-------------------------------|---|
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| 40 | GKS\$_ERROR_40 | Specified workstation has no pixel store readback capability in routine **** |
| 91 | GKS\$_ERROR_91 | Dimensions of color array are invalid in routine **** |

Program Example

Example 11-14 illustrates the use of the function INQUIRE PIXEL ARRAY DIMENSIONS.

Example 11-14: Determining the Dimensions of a Pixel Array

```
C      This program writes the return values of the functions
C      GKS$INQ_PIXEL_ARRAY_DIM to the workstation surface.
      IMPLICIT NONE
      INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
      INTEGER WS_ID, ERROR_STATUS, BEGIN_COL, COLORS( 2, 2 ),
* BEGIN_ROW, NUM_COLUMNS, NUM_ROWS, NUM_PIXEL_COLUMNS,
* NUM_PIXEL_ROWS
      REAL DEVICE_X, DEVICE_Y, WORLD_START_X, WORLD_START_Y,
* WORLD_DIAG_X, WORLD_DIAG_Y
      DATA WS_ID / 1 /,
* BEGIN_COL / 1 /, BEGIN_ROW / 1 /, NUM_COLUMNS / 2 /,
* NUM_ROWS / 2 /, WORLD_START_X / 0.1 /,
* WORLD_START_Y / 0.2 /, WORLD_DIAG_X / 0.2 /,
* WORLD_DIAG_Y / 0.1 /
      DATA COLORS / 2,3, 1,0 /
```

(continued on next page)

Pixel Inquiries

INQUIRE PIXEL ARRAY DIMENSIONS

Example 11-14 (Cont.): Determining the Dimensions of a Pixel Array

```
CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )
CALL GKS$ACTIVATE_WS( WS_ID )

C    Color a small section of the screen with cell array.
    CALL GKS$CELL_ARRAY( WORLD_START_X, WORLD_START_Y,
* WORLD_DIAG_X, WORLD_DIAG_Y, BEGIN_COL, BEGIN_ROW,
* NUM_COLUMNS, NUM_ROWS, %DESCR( COLORS ) )

C    You can obtain this information as long as the specified
C    workstation is open.
    CALL GKS$INQ_PIXEL_ARRAY_DIM( WS_ID, WORLD_START_X,
* WORLD_START_Y, WORLD_DIAG_X, WORLD_DIAG_Y, ERROR_STATUS,
* NUM_PIXEL_COLUMNS, NUM_PIXEL_ROWS)

C    Write the returned values to the screen.
    WRITE(6, *) 'The error status: ', ERROR_STATUS
    WRITE(6, *) 'The number of columns of pixels: ',
* NUM_PIXEL_COLUMNS
    WRITE(6, *) 'The number of rows of pixels: ',
* NUM_PIXEL_ROWS
    CALL GKS$DEACTIVATE_WS( WS_ID )
    CALL GKS$CLOSE_WS( WS_ID )
    CALL GKS$CLOSE_GKS()
END
```

When you compile, link, and execute this program on a VT241 terminal, the following values are written to the workstation surface:

```
$ FORTRAN  EXAMPLE_14 [RETURN]
$ LINK     EXAMPLE_14 [RETURN]
$ RUN      EXAMPLE_14 [RETURN]
The error status:                0
The number of columns of pixels: 47
The number of rows of pixels:   48
$
```

DEC GKS-Supported Workstations

This appendix lists the devices that DEC GKS supports and the defined workstation type of each device. You use the workstation type constants or values in calls to the function OPEN WORKSTATION (refer to Chapter 3, Control Functions). You can also compare the workstation type or value with the values written to INQUIRE WORKSTATION CONNECTION AND TYPE or INQUIRE LIST OF AVAILABLE WORKSTATION TYPES (refer to Chapter 11, Inquiry Functions).

If you are using a language binding and you wish to determine the corresponding workstation-type constants, refer to Appendix B, DEC GKS Constants.

For detailed information concerning each of the devices, refer to the *DEC GKS Device Specifics Reference Manual*.

A.1 Supported Workstation Types

Table A-1 lists the workstation types defined by DEC GKS.

Table A-1: DEC GKS-Supported Workstation Types

| Value | Constant | Description |
|-------|-----------------------|--------------------------|
| 0 | GKS\$K_WSTYPE_DEFAULT | Default workstation type |
| 2 | GKS\$K_GKSM_OUTPUT | GKSM output metafile |
| 3 | GKS\$K_GKSM_INPUT | GKSM input metafile |

(continued on next page)

Table A-1 (Cont.): DEC GKS-Supported Workstation Types

| Value | Constant | Description |
|--------------|--------------------|--|
| 5 | GKS\$K_WSTYPE_WISS | Workstation independent segment storage |
| 7 | GKS\$K_CGM_OUTPUT | CGM output metafile |
| 10 | GKS\$K_VT_OUTPUT | DIGITAL VT125 black and white output only |
| 11 | GKS\$K_VT125 | DIGITAL VT125 with color option |
| 12 | GKS\$K_VT125BW | DIGITAL VT125 (black and white) |
| 13 | GKS\$K_VT240 | DIGITAL VT240 with color option |
| 14 | GKS\$K_VT240BW | DIGITAL VT240 (black and white) |
| 15 | GKS\$K_LCP01 | DIGITAL LCG01 printer |
| 15 | GKS\$K_LCG01 | DIGITAL LCG01 printer |
| 16 | GKS\$K_VT330 | DIGITAL VT330 (black and white) |
| 17 | GKS\$K_VT340 | DIGITAL VT340 with color |
| 31 | GKS\$K_LA34 | DIGITAL LA34 with graphics option |
| 31 | GKS\$K_LA100 | DIGITAL LA100 |
| 32 | GKS\$K_LA50 | DIGITAL LA50 with 2:1 aspect ratio |
| 34 | GKS\$K_LA210 | DIGITAL LA210 |
| 35 | GKS\$K_LA75 | DIGITAL LA75 |
| 38 | GKS\$K_LN03_PLUS | DIGITAL LN03 PLUS |
| 41 | GKS\$K_VSII | DIGITAL VAXstation II (black and white) |
| 41 | GKS\$K_VSII_GPX | DIGITAL VAXstation II/GPX (color), and II/RC |
| 41 | GKS\$K_VS2000 | DIGITAL VAXstation 2000 |
| 51 | GKS\$K_LVP16A | DIGITAL LVP16 color graphics plotter (with 8 1/2 by 11 paper size) |
| 51 | GKS\$K_HP7475 | HP7475 pen plotter |
| 52 | GKS\$K_LVP16B | DIGITAL LVP16 color graphics plotter (with 11 by 17 paper size) |
| 53 | GKS\$K_HP7550 | HP7550 pen plotter |
| 54 | GKS\$K_HP7580 | HP7580 pen plotter |
| 55 | GKS\$K_LG_MPS-2000 | MPS-2000 film recorder |

(continued on next page)

Table A-1 (Cont.): DEC GKS-Supported Workstation Types

| Value | Constant | Description |
|--------------|----------------------------|--|
| 56 | GKS\$K_HP7585 | HP7585 pen plotter |
| 61 | GKS\$K_POSTSCRIPT | DIGITAL LPS40 and PostScript graphics handler |
| 70 | GKS\$K_TEK4014_OUTPUT | Tektronix—4014 output only |
| 72 | GKS\$K_TEK4014 | Tektronix—4014 |
| 80 | GKS\$K_TEK4107_OUTPUT | Tektronix—4107 output only |
| 82 | GKS\$K_TEK4107 | Tektronix—4107 |
| 210 | GKS\$K_DECWINDOWS_OUTPUT | DECwindows—output only |
| 211 | GKS\$K_DECWINDOWS | DECwindows—input/output device |
| 212 | GKS\$K_DECWINDOWS_DRAWABLE | DECwindows—an application window, output only |
| 213 | GKS\$K_DECWINDOWS_WIDGET | DECwindows—input/output within an application widget |

NOTE

In some languages, GKS\$K_CONID_DEFAULT may not be the number 0. For more information, refer to your language's definition file.

The DIGITAL LA34 and LA100 use the same DEC GKS graphics handler. Thus, the workstation type value is the same for both workstations. The same is true for the VSII, the VSII/GPX, the VSII/RC, the VS2000, the LVP16, and the HP7475.

Note that to specify a 2:1 aspect ratio on the LA50, SW1-5 must be left open. See the *LA50 Printer Programmer Reference Manual* for more information.

A.2 Default Workstation Types

The default workstation type for the DEC GKS products running on the VAX systems is the black and white VT240 workstation (14). The default workstation type for the DEC GKS products running on the VAXstations is the VSII workstation (41).

If you specify the value 0 or the constant `GKS$K_WSTYPE_DEFAULT` in a call to a function that accepts a workstation type as an argument, DEC GKS translates the logical name `GKS$WSTYPE` at run time and uses the translation as the type. In this manner, you can define `GKS$WSTYPE` to be a different workstation type value each time you execute your program, and each time the program accepts the newly defined workstation type. For more information, refer to Chapter 1, Introduction to DEC GKS.

A.3 Output-Only Devices

When you use the workstation types designated output only, you can specify the appropriate output-only workstation type and pass a file specification as the second argument to `OPEN WORKSTATION` (connection identifier), as follows:

```
INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
.
.
.
CALL GKS$OPEN_WS( 1, 'FILE_NAME.DAT', GKS$K_VT_OUTPUT )
CALL GKS$ACTIVATE_WS( 1 )
C   Generate output...
.
.
.
```

After the program executes, you can type or print the file at your workstation. The default file type for the connection identifier is `file_name.LIS`; otherwise, DEC GKS uses the file extension that you provide. For information concerning accessing allocated devices as workstations using `GKS$CONID` and `GKS$WSTYPE`, refer to Chapter 1, Introduction to DEC GKS.

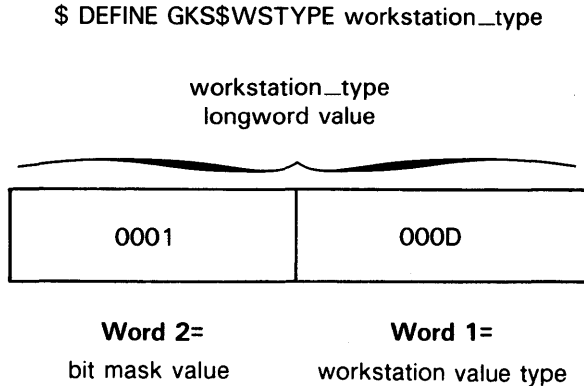
A.4 Using Bit Masks for Workstation Types

You can take advantage of device-dependent features of certain workstations by specifying a hexadecimal bit mask representation as the workstation type. For instance, by specifying different hexadecimal values as the workstation type, you can tell some graphics handlers to use different sizes of paper.

Figure A-1 illustrates the format of a hexadecimal representation of the workstation type. The bit mask in the first part of the workstation type value tells the graphics handler which feature to manipulate. The second part of the workstation type value specifies the hexadecimal representation of the workstation type. For instance, the value `d` (whose decimal equivalent

is the number 13) tells DEC GKS that the workstation type is a color VT240.

Figure A-1: Hexadecimal Bit Masks as Workstation Type Values



```
$ DEFINE GKS$WSTYPE %x0001000D
```

Note:

D = %d13, the workstation type constant for the VT241.

ZK-5137-86

For specific information concerning the supported bit masks for any given device, refer to the appropriate device-specific appendix in this manual.

A.4.1 An Alternative to Defining Bit Masks

In some instances, you may wish to take advantage of device-dependent features by using code within your programs instead of bit mask definitions at the DIGITAL Command Line.

For use within programs, DEC GKS defines a series of constants. By performing a bitwise OR operation on certain constants, you can control device-dependent features such as paper size. To use these constants, you must include the definition file for your programming language. (For more information about definition files, refer to Chapter 1, Introduction to DEC GKS.)

For example, if you wanted to use the LVP16 with landscape orientation and a paper size of A3, you can call OPEN WORKSTATION as follows:

```
INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'  
.  
.  
CALL GKS$OPEN_WS( 1, GKS$K_CONID_DEFAULT,  
* GKS$K_LVP16A .OR. GKS$M_LANDSCAPE .OR. GKS$M_PAPERSIZE_A3 )  
.  
.
```

The DEC GKS constants used as bit masks begin with the prefix GKS\$M.

For more information concerning bit mask constants and your particular device, refer to the appropriate device-specific appendix in this manual. For a complete list of the available bit mask constants, refer to Appendix B, DEC GKS Constants.

Appendix B

DEC GKS Constants

This appendix lists the defined DEC GKS constants for the GKS\$ interface. Using constants in your DEC GKS programs makes your code easier to read.

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

- GKSDEFS.ADA for VAX Ada
- GKSDEFS.BAS for VAX BASIC
- GKSDEFS.R32 for VAX BLISS
- GKSDEFS.H for VAX C
- GKSDEFS.LIB for VAX COBOL
- GKSDEFS.FOR for VAX FORTRAN using the GKS\$ functions
- GKSDEFS.PAS for VAX Pascal
- GKSDEFS.PLI for VAX PL/I routines declared as procedures (no value returns)
- GKSDEFS.PL2 for VAX PL/I routines declared as functions

Table B-1 lists the DEC GKS constant names, their values, and a brief description of each.

Table B-1: GKS\$ Constants

| Constant | Value | Description |
|-------------------------------|--------------|----------------------|
| Action Pending States: | | |
| GKS\$K_NOTPENDING | 0 | Not pending |
| GKS\$K_PENDING | 1 | Pending |
| Arc Types: | | |
| GKS\$K_ARC_TYPE_OPEN | 1 | Arc type open |
| GKS\$K_ARC_TYPE_PIE | 2 | Arc type pie |
| GKS\$K_ARC_TYPE_CHORD | 3 | Arc type chord |
| ASF Masks: | | |
| GKS\$M_LINETYPE | 1 | Line type |
| GKS\$M_LINEWIDTH | 2 | Line width |
| GKS\$M_PLINE_COLOR | 4 | Polyline color |
| GKS\$M_MARKERTYPE | 8 | Marker type |
| GKS\$M_MARKERSIZE | 16 | Marker size |
| GKS\$M_PMARK_COLOR | 32 | Polymarker color |
| GKS\$M_TEXT_FONT_PREC | 64 | Text precision |
| GKS\$M_CHAR_EXPAN_FAC | 128 | Expansion factor |
| GKS\$M_CHAR_SPACE | 256 | Character spacing |
| GKS\$M_TEXT_COLOR | 512 | Text color |
| GKS\$M_FILL_INTER_STYLE | 1024 | Interior style |
| GKS\$M_FILL_STYLE | 2048 | Fill style |
| GKS\$M_FILL_COLOR | 4096 | Fill color |
| GKS\$M_UNCHANGE_PLINE | 8192 | Unchanged polyline |
| GKS\$M_UNCHANGE_PMARK | 16384 | Unchanged polymarker |
| GKS\$M_UNCHANGE_TEXT | 32768 | Unchanged text |
| GKS\$M_UNCHANGE_FILL | 65536 | Unchanged fill area |
| GKS\$M_EDGEType | 131072 | Edge type |
| GKS\$M_EDGEWIDTH | 262144 | Edge width |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|--|--------------|------------------------------------|
| GKS\$M_EDGE_COLOR | 524288 | Edge color |
| GKS\$M_EDGE_CONTROL | 1048576 | Edge control |
| GKS\$M_SIMULATION | 1073741824 | Simulation |
| Attribute Control Function Types: | | |
| GKS\$K_ACF_CURRENT | 0 | Input data record current values |
| GKS\$K_ACF_SPECIFIED | 1 | Input data record specified values |
| Attribute Control Flags: | | |
| GKS\$K_ACF_POLYLINE | 0 | Data record polyline control flag |
| GKS\$K_ACF_FILL_AREA | 1 | Data record fill area control flag |
| Attribute Source States: | | |
| GKS\$K_ASF_BUNDLED | 0 | Bundled |
| GKS\$K_ASF_INDIVIDUAL | 1 | Individual |
| CGM Encoding Bit Masks: | | |
| GKS\$M_CHARACTER_ENCODING | 131072 | Character |
| GKS\$M_BINARY_ENCODING | 196608 | Binary |
| GKS\$M_CLEAR_TEXT_ENCODING | 262144 | Clear text |
| Choice Data Record Flags: | | |
| GKS\$K_CHOICE_PROMPT_OFF | 0 | Choice data record prompt off |
| GKS\$K_CHOICE_PROMPT_ON | 1 | Choice data record prompt on |
| Clear Screen States: | | |
| GKS\$K_CLEAR_CONDITIONALLY | 0 | Clear conditionally |
| GKS\$K_CLEAR_ALWAYS | 1 | Clear always |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|---------------------------------|--------------|------------------------------------|
| Clipping States: | | |
| GKS\$K_NOCLIP | 0 | Clipping off |
| GKS\$K_CLIP | 1 | Clipping on |
| Color Mapping Bit Masks: | | |
| GKS\$M_COLOR_MAP_VIRTUAL | 0 | Use the virtual color indexes |
| GKS\$M_COLOR_MAP_PHYSICAL | 16777216 | Use the physical color indexes |
| Color Table Size: | | |
| GKS\$M_COLOR_MAP_256 | 0 | 256 entries in the color table |
| GKS\$M_COLOR_MAP_2 | 67108864 | 2 entries in the color table |
| GKS\$M_COLOR_MAP_8 | 134217728 | 8 entries in the color table |
| GKS\$M_COLOR_MAP_16 | 201326592 | 16 entries in the color table |
| Connection Identifier: | | |
| GKS\$K_CONID_DEFAULT | 0 | Default connection identifier |
| Coordinate Switch: | | |
| GKS\$K_COORDINATES_WC | 0 | World coordinates |
| GKS\$K_COORDINATES_NDC | 1 | Normalized device coordinates |
| Deferral State Types: | | |
| GKS\$K_ASAP | 0 | As soon as possible |
| GKS\$K_BNIG | 1 | Before the next global interaction |
| GKS\$K_BNIL | 2 | Before the next local interaction |
| GKS\$K_ASTI | 3 | At some time |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|-------------------------------------|--------------|---------------------------------|
| Detectability Flags: | | |
| GKS\$K_UNDETECTABLE | 0 | Set to undetectable |
| GKS\$K_DETECTABLE | 1 | Set to detectable |
| Device Coordinate States: | | |
| GKS\$K_METERS | 0 | Meters |
| GKS\$K_OTHER_UNITS | 1 | Other units |
| Display Surface States: | | |
| GKS\$K_NOTEMPTY | 0 | Display surface not empty |
| GKS\$K_EMPTY | 1 | Display surface empty |
| Dots Per Inch (DPI): | | |
| GKS\$M_DPI_72 | 16777216 | 72 dots per inch |
| GKS\$M_DPI_90 | 0 | 90 dots per inch |
| GKS\$M_DPI_144 | 2097152 | 144 dots per inch |
| GKS\$M_DPI_180 | 50331648 | 180 dots per inch |
| Dynamic Modification States: | | |
| GKS\$K_IRG | 0 | Implicit regeneration necessary |
| GKS\$K_IMM | 1 | Immediate |
| Echo States: | | |
| GKS\$K_NOECHO | 0 | Echo disabled |
| GKS\$K_ECHO | 1 | Echo enabled |
| Edge Types: | | |
| GKS\$K_EDGE_SOLID | 1 | Edge type solid |
| GKS\$K_EDGE_DASHED | 2 | Edge type dashed |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------------|--------------|--|
| GKS\$K_EDGE_DOTTED | 3 | Edge type dotted |
| GKS\$K_EDGE_DASHED_DOTTED | 4 | Edge type dashed-dotted |
| GKS\$K_EDGE_DASH_2_DOT | -1 | Edge type dash-2-dots |
| GKS\$K_EDGE_DASH_3_DOT | -2 | Edge type dash-3-dots |
| GKS\$K_EDGE_LONG_DASH | -3 | Edge type long-dash |
| GKS\$K_EDGE_LONG_SHORT_DASH | -4 | Edge type long-short-dash |
| GKS\$K_EDGE_SPACED_DASH | -5 | Edge type spaced-dash |
| GKS\$K_EDGE_SPACED_DOT | -6 | Edge type spaced-dot |
| GKS\$K_EDGE_DOUBLE_DOT | -7 | Edge type double dots |
| GKS\$K_EDGE_TRIPLE_DOT | -8 | Edge type triple dots |
| Error Handling Modes: | | |
| GKS\$K_ERROR_OFF | 0 | No error handling |
| GKS\$K_ERROR_ON | 1 | Error handling |
| Escapes: | | |
| GKS\$K_ESC_SET_SPEED | -100 | Set speed |
| GKS\$K_ESC_PRINT | -101 | Print |
| GKS\$K_ESC_PRINT_VTP | -102 | Print viewport |
| GKS\$K_ESC_BEEP | -103 | Beep |
| GKS\$K_ESC_CLEAR_REGION | -104 | Clear region |
| GKS\$K_ESC_CLEAR_INPUT | -105 | Clear input |
| GKS\$K_ESC_POP_WORKSTATION | -106 | Pop workstation |
| GKS\$K_ESC_PUSH_WORKSTATION | -107 | Push workstation |
| GKS\$K_ESC_SET_ERR_HANDLING_MODE | -108 | Set Error Handling Mode |
| GKS\$K_ESC_SET_VIEWPORT_EVENT | -109 | Set viewport event |
| GKS\$K_ESC_ASSOC_WSTYPE_CONID | -110 | Associate a connection identifier with a workstation |
| GKS\$K_ESC_SET_SOFT_CLIP | -111 | Soft clipping |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|---------------------------------|--------------|--|
| GKS\$K_ESC_SET_WRITING_MODE | -150 | Set writing mode |
| GKS\$K_ESC_SET_LINE_CAP | -151 | Set line cap |
| GKS\$K_ESC_SET_LINE_JOIN | -152 | Set line join |
| GKS\$K_ESC_SET_EDGE_CTL | -153 | Set edge control flag in GKS state list |
| GKS\$K_ESC_SET_EDGE_TYPE | -154 | Set edge type in GKS state list |
| GKS\$K_ESC_SET_EDGE_WIDTH | -155 | Set edge width scale factor in GKS state list |
| GKS\$K_ESC_SET_EDGE_COLOR_INDEX | -156 | Set edge color index in GKS state list |
| GKS\$K_ESC_SET_EDGE_INDEX | -157 | Set edge index in GKS state list |
| GKS\$K_ESC_SET_EDGE_ASF | -158 | Set aspect source flag entries in GKS state list |
| GKS\$K_ESC_SET_CURSOR | -159 | Set cursor |
| GKS\$K_ESC_BEGIN_TRANS_BLOCK | -160 | Begin transformation block |
| GKS\$K_ESC_END_TRANS_BLOCK | -161 | End transformation block |
| GKS\$K_ESC_SET_SEG_HIGH_METHOD | -162 | Set segment highlighting method |
| GKS\$K_ESC_SET_HIGH_METHOD | -163 | Set highlighting method |
| GKS\$K_ESC_SET_EDGE_REP | -200 | Set edge representation |
| GKS\$K_ESC_SET_FONT_NAME | -201 | Set font name |
| GKS\$K_ESC_SET_WINDOW_TITLE | -202 | Set window title |
| GKS\$K_ESC_SET_RESET_STRING | -203 | Set reset string |
| GKS\$K_ESC_SET_CANCEL_STRING | -204 | Set cancel string |
| GKS\$K_ESC_SET_ENTER_STRING | -205 | Set enter string |
| GKS\$K_ESC_SET_ICON_BITMAPS | -206 | Set icon bitmaps |
| GKS\$K_ESC_PCPCMDS | -207 | PCM (buttons,dials) commands |
| GKS\$K_ESC_INQ_CURSOR | -250 | Inquire cursor |
| GKS\$K_ESC_INQ_WRITING_MODE | -251 | Inquire writing mode |
| GKS\$K_ESC_INQ_LINE_CAP | -252 | Inquire line cap |
| GKS\$K_ESC_INQ_LINE_JOIN | -253 | Inquire line join |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------------|--------------|--|
| GKS\$K_ESC_INQ_EDGE_ATTR | -254 | Inquire current edge attributes |
| GKS\$K_ESC_INQ_VIEWPORT_DATA | -255 | Inquire viewport data |
| GKS\$K_ESC_INQ_SPEED | -300 | Inquire speed |
| GKS\$K_ESC_INQ_LIST_FONT_NAMES | -301 | Inquire list of font names |
| GKS\$K_ESC_INQ_LIST_EDGE_INDEXES | -302 | Inquire list of edge indices |
| GKS\$K_ESC_INQ_SEGMENT_EXTENT | -303 | Inquire segment extent |
| GKS\$K_ESC_INQ_WINDOW_IDS | -304 | Inquire window identifiers |
| GKS\$K_ESC_INQ_SEG_HIGH_METHOD | -305 | Inquire segment highlighting |
| GKS\$K_ESC_INQ_HIGH_METHOD | -306 | Inquire highlighting method |
| GKS\$K_ESC_INQ_PASTEBOARD_ID | -307 | Inquire pasteboard identifier |
| GKS\$K_ESC_INQ_MENU_BAR_ID | -308 | Inquire menu bar identifier |
| GKS\$K_ESC_INQ_SHELL_ID | -309 | Inquire shell identifier |
| GKS\$K_ESC_INQ_LIST_ESC | -350 | Inquire list of escapes |
| GKS\$K_ESC_INQ_DEF_SPEED | -351 | Inquire default display speed |
| GKS\$K_ESC_INQ_LINE_CAP_JOIN_FAC | -352 | Inquire cap join facility |
| GKS\$K_ESC_INQ_FONT_NAME_FAC | -353 | Inquire font name facility |
| GKS\$K_ESC_INQ_EDGE_FAC | -354 | Inquire edge facilities |
| GKS\$K_ESC_INQ_PREDEF_EDGE_REP | -355 | Inquire predefined edge representation for workstation type and edge index |
| GKS\$K_ESC_INQ_MAX_EDGE_BUNDLE | -356 | Inquire maximum number of edge bundle entries |
| GKS\$K_ESC_INQ_CURSOR_SIZE | -357 | Inquire cursor size |
| GKS\$K_ESC_INQ_LIST_HIGH | -358 | Inquire list highlighting |
| GKS\$K_ESC_INQ_EDGE_REP | -359 | Inquire edge representation |
| GKS\$K_ESC_MAP_NDC_OF_WC | -400 | Map WC to NDC |
| GKS\$K_ESC_MAP_DC_OF_NDC | -401 | Map NDC to DC |
| GKS\$K_ESC_MAP_WC_OF_NDC | -402 | Map NDC to WC |
| GKS\$K_ESC_MAP_NDC_OF_DC | -403 | Map DC to NDC |
| GKS\$K_ESC_INQ_GDP_EXTENT | -404 | Inquire GDP extent |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|-------------------------------|--------------|-----------------------------------|
| GKS\$K_ESC_DOUBLE_BUFFER | -500 | Set double buffering mode |
| GKS\$K_ESC_SET_BCKGRND | -501 | Set background pixmap |
| GKS\$K_ESC_INQ_DBUFFER_PIXMAP | -502 | Inquire double buffer pixmap |
| GKS\$K_ESC_INQ_BCKGRND_PIXMAP | -503 | Inquire background pixmap |
| Fill Area Types: | | |
| GKS\$K_INTSTYLE_HOLLOW | 0 | Interior style hollow |
| GKS\$K_INTSTYLE_SOLID | 1 | Interior style solid |
| GKS\$K_INTSTYLE_PATTERN | 2 | Interior style pattern |
| GKS\$K_INTSTYLE_HATCH | 3 | Interior style hatched |
| GDP Bundle Types: | | |
| GKS\$K_POLYLN_ATTRI | 0 | GDP polyline bundle |
| GKS\$K_POLYMR_ATTRI | 1 | GDP polymarker bundle |
| GKS\$K_TEXT_ATTRI | 2 | GDP text bundle |
| GKS\$K_FILLAR_ATTRI | 3 | GDP fill area bundle |
| GDPs: | | |
| GKS\$K_GDP_DISJOINT_PLINE | -100 | Disjoint polyline |
| GKS\$K_GDP_CIRCLE_CTR_PT | -101 | Center and point on circle |
| GKS\$K_GDP_CIRCLE_3PT | -102 | 3 points on circle |
| GKS\$K_GDP_CIRCLE_CTR_RAD | -103 | Center and radius of circle |
| GKS\$K_GDP_CIRCLE_2PT_RAD | -104 | 2 points and radius of circle |
| GKS\$K_GDP_ARC_CTR_2PT | -106 | Center and 2 points of the arc |
| GKS\$K_GDP_ARC_3PT | -107 | 3 points of arc |
| GKS\$K_GDP_ARC_CTR_2VEC_RAD | -108 | Center and 2 vector radius of arc |
| GKS\$K_GDP_ARC_2PT_RAD | -109 | 2 points and radius of the arc |
| GKS\$K_GDP_ARC_CTR_PT_ANG | -110 | Center point and angle for arc |
| GKS\$K_GDP_ELLIPSE_CTR_AXES | -111 | Center and axes of ellipse |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------------|--------------|---|
| GKS\$K_GDP_ELLIPSE_CTR_3PT | -112 | Center and 3 points of ellipse |
| GKS\$K_GDP_ELLIPSE_FOCII_PT | -113 | Foci and point of ellipse |
| GKS\$K_GDP_ELIARC_CTR_AXES_2VEC | -114 | Center, 2 vectors of elliptic arc |
| GKS\$K_GDP_ELIARC_FOCII_2PT | -116 | Foci, 2 points on elliptic arc |
| GKS\$K_GDP_RECT_2PT | -125 | Rectangle by 2 points |
| GKS\$K_GDP_RESTRICTED_TEXT | -231 | Restricted text |
| GKS\$K_GDP_FILL_AREA_SET | -332 | Fill area set |
| GKS\$K_GDP_FCIRCLE_CTR_PT | -333 | Fill circle using center point |
| GKS\$K_GDP_FCIRCLE_3PT | -334 | Fill circle using 3 points |
| GKS\$K_GDP_FCIRCLE_CTR_RAD | -335 | Fill circle using center and radius |
| GKS\$K_GDP_FCIRCLE_2PT_RAD | -336 | Fill circle using 2 points and radius |
| GKS\$K_GDP_FARC_CTR_2PT | -338 | Fill arc using center and 2 points of the arc |
| GKS\$K_GDP_FARC_3PT | -339 | Fill arc using 3 points |
| GKS\$K_GDP_FARC_CTR_2VEC_RAD | -340 | Fill arc using 2 vectors and radius |
| GKS\$K_GDP_FARC_2PT_RAD | -341 | Fill arc using 2 points and radius of the arc |
| GKS\$K_GDP_FARC_CTR_PT_ANG | -342 | Fill arc using center, point, angle |
| GKS\$K_GDP_FELLIPSE_CTR_AXES | -343 | Fill ellipse using center, axes |
| GKS\$K_GDP_FELLIPSE_CTR_3PT | -344 | Fill ellipse with center, 3 points |
| GKS\$K_GDP_FELLIPSE_FOCII_PT | -345 | Fill ellipse using foci, point |
| GKS\$K_GDP_FELIARC_CTR_AXES_2VEC | -346 | Fill elliptic arc using center, axes, 2 vectors |
| GKS\$K_GDP_FELIARC_FOCII_2PT | -348 | Fill elliptic arc using foci, 2 points |
| GKS\$K_GDP_FRECT_2PT | -349 | Fill rectangle using 2 points |
| GKS\$K_GDP_IMAGE_ARRAY | -400 | Packed cell array GDP |
| GKS Level Types: | | |
| GKS\$K_LEVEL_MA | -3 | Minimal output, no input |
| GKS\$K_LEVEL_MB | -2 | Minimal output, request input |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------|--------------|--|
| GKS\$K_LEVEL_MC | -1 | Minimal output, full input |
| GKS\$K_LEVEL_0A | 0 | All primitives and attributes, no input |
| GKS\$K_LEVEL_0B | 1 | All primitives and attributes, request input |
| GKS\$K_LEVEL_0C | 2 | All primitives and attributes, full input |
| GKS\$K_LEVEL_1A | 3 | Basic segmentation with full output, no input |
| GKS\$K_LEVEL_1B | 4 | Basic segmentation with full output, request input |
| GKS\$K_LEVEL_1C | 5 | Basic segmentation with full output, full input |
| GKS\$K_LEVEL_2A | 6 | Workstation independent and segment storage, no input |
| GKS\$K_LEVEL_2B | 7 | Workstation independent and segment storage, request input |
| GKS\$K_LEVEL_2C | 8 | Workstation independent and segment storage, full input |
| GKS Status Types: | | |
| GKS\$K_GKCL | 0 | GKS closed |
| GKS\$K_GKOP | 1 | GKS open |
| GKS\$K_WSOP | 2 | At least one workstation open |
| GKS\$K_WSAC | 3 | At least one workstation active |
| GKS\$K_SGOP | 4 | At least one segment open |
| Highlighting Flags: | | |
| GKS\$K_NORMAL | 0 | Primitives are not highlighted |
| GKS\$K_HIGHLIGHTED | 1 | Primitives are highlighted |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|--------------------------------------|--------------|--|
| Highlighting Methods: | | |
| GKS\$K_HIGH_METHOD_DEFAULT | 0 | Default highlighting |
| GKS\$K_HIGH_METHOD_COMP | 1 | Highlight with complement mode |
| GKS\$K_HIGH_METHOD_COLOR | 2 | Highlight with color |
| GKS\$K_HIGH_METHOD_LINE | 3 | Highlight with extent line box |
| GKS\$K_HIGH_METHOD_FILL | 4 | Highlight with extent fill area |
| GKS\$K_HIGH_METHOD_DUAL | 5 | Highlight with extent line box and fill area |
| Implicit Regeneration States: | | |
| GKS\$K_IRG_SUPPRESSED | 0 | Implicit regeneration suppressed |
| GKS\$K_IRG_ALLOWED | 1 | Implicit regeneration allowed |
| Input Classes: | | |
| GKS\$K_INPUT_CLASS_NONE | 0 | No input class |
| GKS\$K_INPUT_CLASS_LOCATOR | 1 | Locator input class |
| GKS\$K_INPUT_CLASS_STROKE | 2 | Stroke input class |
| GKS\$K_INPUT_CLASS_VALUATOR | 3 | Valuator input class |
| GKS\$K_INPUT_CLASS_CHOICE | 4 | Choice input class |
| GKS\$K_INPUT_CLASS_PICK | 5 | Pick input class |
| GKS\$K_INPUT_CLASS_STRING | 6 | String input class |
| GKS\$K_INPUT_CLASS_VIEWPORT | 7 | Viewport input class |
| Input Device Type: | | |
| GKS\$K_INPUT_DEV_DEFAULT | 0 | Default input device |
| Input Mode Types: | | |
| GKS\$K_INPUT_MODE_REQUEST | 0 | Request mode |
| GKS\$K_INPUT_MODE_SAMPLE | 1 | Sample mode |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|---------------------------------|--------------|--|
| GKS\$K_INPUT_MODE_EVENT | 2 | Event mode |
| Input on Device Handler: | | |
| GKS\$K_CURSOR_MOVEMENT | 1 | Input key is a cursor movement |
| GKS\$K_CHARACTER | 2 | Input key is a character |
| GKS\$K_POINT_TRIGGER | 3 | Input key is a point trigger |
| GKS\$K_TERMINATION_TRIGGER | 4 | Input key is a termination trigger |
| GKS\$K_DELETE_KEY | 5 | Input key is delete |
| GKS\$K_BREAK | 6 | Input key is break |
| GKS\$K_CHOICE_NUMBER | 7 | Input key is a choice number |
| GKS\$K_CYCLE | 8 | Input key is cycle |
| GKS\$K_NOCYCLE | 9 | Input key is no cycle |
| GKS\$K_STROKE_MEASURE | 10 | Input key is a stroke measure |
| GKS\$K_TOGGLE_INSERT | 11 | Input key is insert toggle |
| GKS\$K_RESTORE_INITIAL | 12 | Input key is to restore initial string |
| GKS\$K_BUFFER_BEGINNING | 13 | Input key is move to beginning |
| GKS\$K_BUFFER_END | 14 | Input key is move to end |
| GKS\$K_VALUATOR_VALUE | 15 | Input key is a valuator measure |
| GKS\$K_SIGNAL | 16 | Input signal occurred |
| GKS\$K_LOG_ERROR | 17 | Input error occurred |
| Input on Device Handler: | | |
| GKS\$K_RELATIVE | 0 | Relative movement |
| GKS\$K_ABSOLUTE | 1 | Absolute movement |
| Input Priority States: | | |
| GKS\$K_INPUT_PRIORITY_HIGHER | 0 | Relative input priority higher |
| GKS\$K_INPUT_PRIORITY_LOWER | 1 | Relative input priority lower |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|--|--------------|---------------------------|
| Input Status Types: | | |
| GKS\$K_STATUS_NONE | 0 | No input obtained |
| GKS\$K_STATUS_OK | 1 | Input obtained |
| GKS\$K_STATUS_NOCHOICE | 2 | Input is NOCHOICE |
| GKS\$K_STATUS_NOPICK | 2 | Input is NOPICK |
| Line Cap Types: | | |
| GKS\$K_LINE_CAP_BUTT | 2 | Line cap type butted |
| GKS\$K_LINE_CAP_ROUND | 3 | Line cap type rounded |
| GKS\$K_LINE_CAP_SQUARE | 4 | Line cap type square |
| Line Join Types: | | |
| GKS\$K_LINE_JOIN_MITRE | 2 | Line join type mitre |
| GKS\$K_LINE_JOIN_ROUND | 3 | Line join type round |
| GKS\$K_LINE_JOIN_BEVEL | 4 | Line join type bevel |
| Line Types (standard): | | |
| GKS\$K_LINETYPE_SOLID | 1 | Line type solid |
| GKS\$K_LINETYPE_DASHED | 2 | Line type dashed |
| GKS\$K_LINETYPE_DOTTED | 3 | Line type dotted |
| GKS\$K_LINETYPE_DASHED_DOTTED | 4 | Line type dash-dotted |
| Line Types (DEC GKS Implementation Specific): | | |
| GKS\$K_LINETYPE_DASH_2_DOT | -1 | Line type dash-2-dots |
| GKS\$K_LINETYPE_DASH_3_DOT | -2 | Line type dash-3-dots |
| GKS\$K_LINETYPE_LONG_DASH | -3 | Line type long-dash |
| GKS\$K_LINETYPE_LONG_SHORT_DASH | -4 | Line type long-short-dash |
| GKS\$K_LINETYPE_SPACED_DASH | -5 | Line type spaced-dash |
| GKS\$K_LINETYPE_SPACED_DOT | -6 | Line type spaced-dot |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|--|--------------|----------------------------------|
| GKS\$K_LINETYPE_DOUBLE_DOT | -7 | Line type double dots |
| GKS\$K_LINETYPE_TRIPLE_DOT | -8 | Line type triple dots |
| Logical Types: | | |
| GKS\$K_FALSE | 0 | Logical FALSE |
| GKS\$K_TRUE | 1 | Logical TRUE |
| Marker Types (standard): | | |
| GKS\$K_MARKERTYPE_DOT | 1 | Marker type dot (.) |
| GKS\$K_MARKERTYPE_PLUS | 2 | Marker type plus (+) |
| GKS\$K_MARKERTYPE_ASTERISK | 3 | Marker type asterisk (*) |
| GKS\$K_MARKERTYPE_CIRCLE | 4 | Marker type circle (o) |
| GKS\$K_MARKERTYPE_DIAGONAL_CROSS | 5 | Marker type diagonal cross (X) |
| Marker Types (DEC GKS Implementation Specific): | | |
| GKS\$K_MARKERTYPE_SOLID_CIRCLE | -1 | Marker type solid circle |
| GKS\$K_MARKERTYPE_TRIANGLE_UP | -2 | Marker type hollow up triangle |
| GKS\$K_MARKERTYPE_SOLID_TRI_UP | -3 | Marker type solid up triangle |
| GKS\$K_MARKERTYPE_TRIANGLE_DOWN | -4 | Marker type hollow down triangle |
| GKS\$K_MARKERTYPE_SOLID_TRI_DOWN | -5 | Marker type solid down triangle |
| GKS\$K_MARKERTYPE_SQUARE | -6 | Marker type hollow square |
| GKS\$K_MARKERTYPE_SOLID_SQUARE | -7 | Marker type solid square |
| GKS\$K_MARKERTYPE_BOWTIE | -8 | Marker type hollow bow tie |
| GKS\$K_MARKERTYPE_SOLID_BOWTIE | -9 | Marker type solid bow tie |
| GKS\$K_MARKERTYPE_HOURLASS | -10 | Marker type hollow hourglass |
| GKS\$K_MARKERTYPE_SOLID_HGLASS | -11 | Marker type solid hourglass |
| GKS\$K_MARKERTYPE_DIAMOND | -12 | Marker type hollow diamond |
| GKS\$K_MARKERTYPE_SOLID_DIAMOND | -13 | Marker type solid diamond |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|-------------------------------------|--------------|---------------------------------|
| New Frame Action States: | | |
| GKS\$K_NEWFRAME_NOTNECESSARY | 0 | No new frame action on update |
| GKS\$K_NEWFRAME_NECESSARY | 1 | New frame action on update |
| Paper Size Bit Masks: | | |
| GKS\$M_PAPERSIZE_A | 0 | 8.5×11 inches |
| GKS\$M_PAPERSIZE_LEGAL | 65536 | 8.5×14 inches |
| GKS\$M_PAPERSIZE_B | 131072 | 11×17 inches |
| GKS\$M_PAPERSIZE_C | 196608 | 17×22 inches |
| GKS\$M_PAPERSIZE_D | 262144 | 22×34 inches |
| GKS\$M_PAPERSIZE_E | 327680 | 34×44 inches |
| GKS\$M_PAPERSIZE_A0 | 1048576 | 84.1×118.9 centimeters |
| GKS\$M_PAPERSIZE_A1 | 2097152 | 59.4×84.1 centimeters |
| GKS\$M_PAPERSIZE_A2 | 3145728 | 42×59.4 centimeters |
| GKS\$M_PAPERSIZE_A3 | 4194304 | 29.7×42.0 centimeters |
| GKS\$M_PAPERSIZE_A4 | 5242880 | 21.0×29.7 centimeters |
| GKS\$M_PAPERSIZE_A5 | 6291456 | 14.8×21.0 centimeters |
| GKS\$M_PAPERSIZE_B4 | 7340032 | 25.7×36.4 centimeters |
| GKS\$M_PAPERSIZE_B5 | 8388608 | 18.2×25.7 centimeters |
| Paper Orientation Bit Masks: | | |
| GKS\$M_LANDSCAPE | 0 | Landscape orientation |
| GKS\$M_PORTRAIT | 268435456 | Portrait orientation |
| Regeneration Flag States: | | |
| GKS\$K_POSTPONE_FLAG | 0 | Implicit regeneration postponed |
| GKS\$K_PERFORM_FLAG | 1 | Implicit regeneration performed |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|---|--------------|---|
| ReGIS Bit Masks: | | |
| GKS\$M_VT125_OUTPUT | 1048576 | VT125 color, output only |
| GKS\$M_VT125BW_OUTPUT | 2097152 | VT125 black/white, output only |
| GKS\$M_VT240_OUTPUT | 3145728 | VT240 color, output only |
| GKS\$M_VT240BW_OUTPUT | 4194304 | VT240 black/white, output only |
| GKS\$M_VT330BW_OUTPUT | 5242880 | VT330 black/white, output only |
| GKS\$M_VT340_OUTPUT | 6291456 | VT340 color, output only |
| Returned Type Values: | | |
| GKS\$K_VALUE_SET | 0 | Type of returned value is set |
| GKS\$K_VALUE_REALIZED | 1 | Type of returned value is realized |
| Simultaneous Events Flag: | | |
| GKS\$K_NOMORE_EVENTS | 0 | No more simultaneously generated events |
| GKS\$K_MORE_EVENTS | 1 | More simultaneously generated events |
| Text Horizontal Alignment Types: | | |
| GKS\$K_TEXT_HALIGN_NORMAL | 0 | Horizontal align normal |
| GKS\$K_TEXT_HALIGN_LEFT | 1 | Horizontal align left |
| GKS\$K_TEXT_HALIGN_CENTER | 2 | Horizontal align center |
| GKS\$K_TEXT_HALIGN_RIGHT | 3 | Horizontal align right |
| Text Path Types: | | |
| GKS\$K_TEXT_PATH_RIGHT | 0 | Path right |
| GKS\$K_TEXT_PATH_LEFT | 1 | Path left |
| GKS\$K_TEXT_PATH_UP | 2 | Path up |
| GKS\$K_TEXT_PATH_DOWN | 3 | Path down |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|---------------------------------------|--------------|---|
| Text Precision Types: | | |
| GKS\$K_TEXT_PRECISION_STRING | 0 | Text precision string |
| GKS\$K_TEXT_PRECISION_CHAR | 1 | Text precision character |
| GKS\$K_TEXT_PRECISION_STROKE | 2 | Text precision stroke |
| Text Vertical Alignment Types: | | |
| GKS\$K_TEXT_VALIGN_NORMAL | 0 | Vertical align normal |
| GKS\$K_TEXT_VALIGN_TOP | 1 | Vertical align top |
| GKS\$K_TEXT_VALIGN_CAP | 2 | Vertical align cap |
| GKS\$K_TEXT_VALIGN_HALF | 3 | Vertical align half |
| GKS\$K_TEXT_VALIGN_BASE | 4 | Vertical align base |
| GKS\$K_TEXT_VALIGN_BOTTOM | 5 | Vertical align bottom |
| Valid Values Flags: | | |
| GKS\$K_INVALID_ABSENT | 0 | Invalid values absent |
| GKS\$K_INVALID_PRESENT | 1 | Invalid values present |
| Visibility Flags: | | |
| GKS\$K_INVISIBLE | 0 | Set to invisible |
| GKS\$K_VISIBLE | 1 | Set to visible |
| VT3xx Masks: | | |
| GKS\$M_NOPOINTER | 65536 | VT330 or VT340 that does not have a mouse |
| GKS\$M_COLOR_MAP_RESET | 16777216 | VT330 or VT340 that saves the colormap |

Workstation Category Types:

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------------|--------------|---|
| GKS\$K_WSCAT_OUTPUT | 0 | Output |
| GKS\$K_WSCAT_INPUT | 1 | Input |
| GKS\$K_WSCAT_OUTIN | 2 | Out/In |
| GKS\$K_WSCAT_WISS | 3 | Workstation independent segment storage |
| GKS\$K_WSCAT_MO | 4 | Metafile output |
| GKS\$K_WSCAT_MI | 5 | Metafile input |
| Workstation Class Types: | | |
| GKS\$K_WSCLASS_VECTOR | 0 | Vector |
| GKS\$K_WSCLASS_RASTER | 1 | Raster |
| GKS\$K_WSCLASS_OTHERD | 2 | Other device |
| Workstation Color States: | | |
| GKS\$K_MONOCHROME | 0 | Monochrome |
| GKS\$K_COLOR | 1 | Color |
| Workstation States: | | |
| GKS\$K_WS_INACTIVE | 0 | Inactive |
| GKS\$K_WS_ACTIVE | 1 | Active |
| Workstation Types: | | |
| GKS\$K_WSTYPE_DEFAULT | 0 | Default workstation type |
| GKS\$K_GKSM_OUTPUT | 2 | GKS output metafile |
| GKS\$K_GKSM_INPUT | 3 | GKS input metafile |
| GKS\$K_WSTYPE_WISS | 5 | GKS workstation independent segment storage |
| GKS\$K_CGM_OUTPUT | 7 | CGM output metafile |
| GKS\$K_VT_OUTPUT | 10 | DIGITAL VT125 (output only) |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|-----------------------|--------------|--------------------------------------|
| GKS\$K_VT125BW_OUTPUT | 10 | Black and white DIGITAL VT125 |
| GKS\$K_VT125 | 11 | DIGITAL VT125 with color option |
| GKS\$K_VT125BW | 12 | Black and white DIGITAL VT125 |
| GKS\$K_VT240 | 13 | DIGITAL VT240 with color option |
| GKS\$K_VT240BW | 14 | Black and white DIGITAL VT240 |
| GKS\$K_LCP01 | 15 | DIGITAL LCP01 printer |
| GKS\$K_LCG01 | 15 | DIGITAL LCG01 printer |
| GKS\$K_VT330 | 16 | DIGITAL VT330 (black and white) |
| GKS\$K_VT340 | 17 | DIGITAL VT340 (color) |
| GKS\$K_LA34 | 31 | DIGITAL LA34 with graphics option |
| GKS\$K_LA100 | 31 | DIGITAL LA100 |
| GKS\$K_LA50 | 32 | DIGITAL LA50 with 2:1 aspect ratio |
| GKS\$K_LA210 | 34 | DIGITAL LA210 |
| GKS\$K_LA75 | 35 | DIGITAL LA75 |
| GKS\$K_LNO3_PLUS | 38 | DIGITAL LN03 PLUS |
| GKS\$K_VSII | 41 | DIGITAL VAXstation II |
| GKS\$K_VSII_GPX | 41 | DIGITAL VAXstation II/GPX |
| GKS\$K_VS2000 | 41 | DIGITAL VAXstation 2000 |
| GKS\$K_VS3200 | 41 | DIGITAL VAXstation 3200 |
| GKS\$K_VS3500 | 41 | DIGITAL VAXstation 3500 |
| GKS\$K_LVP16A | 51 | DIGITAL LVP16 color plotter (8.5×11) |
| GKS\$K_HP7475 | 51 | Hewlett Packard HP7475 |
| GKS\$K_LVP16B | 52 | DIGITAL LVP16 color plotter (11×17) |
| GKS\$K_HP7550 | 53 | HP7550 pen plotter |
| GKS\$K_HP7580 | 54 | HP7580 pen plotter |
| GKS\$K_LG_MPS2000 | 55 | Lasergraphics film recorder |

(continued on next page)

Table B-1 (Cont.): GKS\$ Constants

| Constant | Value | Description |
|----------------------------|--------------|-----------------------------------|
| GKS\$K_HP7585 | 56 | HP7585 pen plotter |
| GKS\$K_POSTSCRIPT | 61 | PostScript graphics handler |
| GKS\$K_COLOR_POSTSCRIPT | 62 | Color PostScript graphics handler |
| GKS\$K_TEK4014_OUTPUT | 70 | Tektronix—4014 (output only) |
| GKS\$K_TEK4014 | 72 | Tektronix—4014 |
| GKS\$K_TEK4107_OUTPUT | 80 | Tektronix—4107 (output only) |
| GKS\$K_TEK4107 | 82 | Tektronix—4107 |
| GKS\$K_TEK4207_OUTPUT | 83 | Tektronix 4207 (output only) |
| GKS\$K_TEK4207 | 84 | Tektronix 4207 |
| GKS\$K_TEK4128_OUTPUT | 85 | Tektronix 4128 (output only) |
| GKS\$K_TEK4128 | 86 | Tektronix 4128 |
| GKS\$K_TEK4129_OUTPUT | 87 | Tektronix 4129 (output only) |
| GKS\$K_VS500_OUTPUT | 87 | VS500 (output only) |
| GKS\$K_TEK4129 | 88 | Tektronix 4129 |
| GKS\$K_VS500 | 88 | VS500 interactive |
| GKS\$K_LJ250 | 91 | DIGITAL LJ250 90 DPI |
| GKS\$K_LJ250_180DPI | 92 | DIGITAL LJ250 180 DPI |
| GKS\$K_DECWINDOWS_OUTPUT | 210 | DECwindows output |
| GKS\$K_DECWINDOWS | 211 | DECwindows |
| GKS\$K_DECWINDOWS_DRAWABLE | 212 | DECwindows drawable |
| GKS\$K_DECWINDOWS_WIDGET | 213 | DECwindows widget |
| GKS\$K_DDIF | 250 | DDIF |
| Writing Modes: | | |
| GKS\$K_WRT_MODE_COMPLEMENT | 2 | Complement writing mode |
| GKS\$K_WRT_MODE_ERASE | 3 | Erase writing mode |
| GKS\$K_WRT_MODE_OVERLAY | 4 | Overlay writing mode |

DEC GKS Attribute Values

This appendix lists the initial values of all output attributes and normalization transformation settings according to the following categories:

- Polyline attributes
- Polymarker attributes
- Text attributes
- Fill area attributes
- Segment attributes
- Normalization transformation settings

This appendix also lists the following DEC GKS specific attribute types:

- Line types
- Marker types

C.1 Initial Polyline Attributes

This section lists the initial values for the polyline attributes.

| Attribute | Initial Value | Description |
|----------------|-----------------------|-----------------------------|
| Polyline index | 1 | Polyline bundle number 1 |
| Line type | GKS\$K_LINETYPE_SOLID | Solid line |
| Line width | 1.0 | Minimum width |
| Color index | 1 | Workstation-dependent value |
| Line type ASF | GKS\$K_ASF_INDIVIDUAL | Use current line type |

| Attribute | Initial Value | Description |
|------------------|-----------------------|------------------------------|
| Line width ASF | GKS\$K_ASF_INDIVIDUAL | Use current line width |
| Color index ASF | GKS\$K_ASF_INDIVIDUAL | Use current line color index |

C.2 Initial Polymarker Attributes

This section lists the initial values for the polymarker attributes.

| Attribute | Initial Value | Description |
|------------------------|----------------------------|--------------------------------|
| Polymarker index | 1 | Polymarker bundle number 1 |
| Marker type | GKS\$K_MARKERTYPE_ASTERISK | Asterisk for marker |
| Marker size | 1.0 | Nominal size |
| Color index | 1 | Workstation-dependent value |
| Marker type ASF | GKS\$K_ASF_INDIVIDUAL | Use current marker type |
| Marker size ASF | GKS\$K_ASF_INDIVIDUAL | Use current marker size |
| Marker color index ASF | GKS\$K_ASF_INDIVIDUAL | Use current marker color index |

C.3 Initial Text Attributes

This section lists the initial values for the text attributes.

| Attribute | Initial Value | Description |
|--------------------------------|--|--|
| Text index | 1 | Text bundle number 1 |
| Text font and precision | 1 GKS\$K_TEXT_PRECISION_STRING | Hardware font 1, string precision |
| Character expansion factor | 1.0 | Width-to-height ratio from font file |
| Character spacing | 0.0 | Adjacent character bodies |
| Color index | 1 | Workstation-dependent value |
| Text font and precision ASF | GKS\$K_ASF_INDIVIDUAL | Use current font and precision |
| Character expansion factor ASF | GKS\$K_ASF_INDIVIDUAL | Use current width and height ratio |
| Character spacing ASF | GKS\$K_ASF_INDIVIDUAL | Use current character space |
| Text color index ASF | GKS\$K_ASF_INDIVIDUAL | Use current text color index |
| Character height | 0.01 | Capital letters at 0.01 world coordinate units |
| Character up vector | 0,1 | Up vector parallel to y-axis in world coordinate units |
| Text path | GKS\$K_TEXT_PATH_RIGHT | Right angle clockwise from up vector |
| Text alignment | GKS\$K_TEXT_HALIGN_NORMAL GKS\$K_TEXT_VALIGN_NORMAL | Natural alignment with respect to text path |

C.4 Initial Fill Area Attributes

This section lists the initial values for the fill area attributes.

| Attribute | Initial Value | Description |
|-------------------------|------------------------|--|
| Fill area index | 1 | Fill area bundle number 1 |
| Interior style | GKS\$K_INTSTYLE_HOLLOW | Boundary of polygonal area |
| Style index | 1 | Workstation-dependent pattern or hatch style |
| Color index | 1 | Workstation-dependent value |
| Interior style ASF | GKS\$K_ASF_INDIVIDUAL | Use current interior style |
| Style index ASF | GKS\$K_ASF_INDIVIDUAL | Use current pattern or hatch style |
| Color index ASF | GKS\$K_ASF_INDIVIDUAL | Use current fill area color index |
| Pattern size | 1.0,1.0 | Unit square in world coordinates |
| Pattern reference point | 0.0,0.0 | Pattern starting point in world coordinates |

C.5 Initial Segment Attributes

This section lists the initial segment attributes.

| Attribute | Initial Value | Description |
|-----------------------|----------------------|--|
| Transformation number | 0 | The identity transformation presents the segment as stored in NDC space. |
| Visibility | GKS\$K_VISIBLE | The segment is visible. |
| Highlighting | GKS\$K_NORMAL | The segment is not highlighted. |
| Segment priority | 0.0 | The segment has the lowest priority. |
| Detectability | GKS\$K_UNDETECTABLE | The segment is undetectable. |

The default segment transformation is called the identity transformation. The identity transformation uses a 2 x 3 matrix whose first row is composed of the values 1.0, 0.0, 0.0, and whose second row is composed of the values 0.0, 1.0, 0.0.

C.6 Initial Normalization Transformation Settings

The initial normalization transformation number is the value 0.

The initial viewport input priority is in sequential order from the value 0 through the value 255, with transformation number 0 the highest and 255 the lowest.

The default normalization window and viewport limits are rectangular, begin with a lower left corner point of (0.0, 0.0), and extend to the value 1.0 on both the X and Y axes.

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

C.7 DEC GKS-Specific Line Types

The following list presents the DEC GKS-supported line types. To see which types your device supports, refer to the appropriate device-specific appendix.

| Value | Constant | Description |
|--------------|---------------------------------|---|
| -1 | GKS\$K_LINETYPE_DASH_2_DOT | Use a sequence of one dash followed by two dots. |
| -2 | GKS\$K_LINETYPE_DASH_3_DOT | Use a sequence of one dash followed by three dots. |
| -3 | GKS\$K_LINETYPE_LONG_DASH | Use a sequence of long dashes. |
| -4 | GKS\$K_LINETYPE_LONG_SHORT_DASH | Use a sequence of a long dash followed by a short dash. |
| -5 | GKS\$K_LINETYPE_SPACED_DASH | Use a sequence of dashes double spaced. |
| -6 | GKS\$K_LINETYPE_SPACED_DOT | Use a sequence of dots double spaced. |
| -7 | GKS\$K_LINETYPE_DOUBLE_DOT | Use a sequence of pairs of dots. |
| -8 | GKS\$K_LINETYPE_TRIPLE_DOT | Use a sequence of groups of three dots. |

C.8 DEC GKS-Specific Marker Types

The following list presents the DEC GKS-supported marker types. To see which types your device supports, refer to the appropriate device-specific appendix.

| Value | Constant | Description |
|--------------|----------------------------------|--|
| -1 | GKS\$K_MARKERTYPE_SOLID_CIRCLE | Use a filled circle. |
| -2 | GKS\$K_MARKERTYPE_TRIANGLE_UP | Use a hollow triangle pointing upward. |
| -3 | GKS\$K_MARKERTYPE_SOLID_TRI_UP | Use a filled triangle pointing upward. |
| -4 | GKS\$K_MARKERTYPE_TRIANGLE_DOWN | Use a hollow triangle pointing downward. |
| -5 | GKS\$K_MARKERTYPE_SOLID_TRI_DOWN | Use a filled triangle pointing downward. |
| -6 | GKS\$K_MARKERTYPE_SQUARE | Use a hollow square. |
| -7 | GKS\$K_MARKERTYPE_SOLID_SQUARE | Use a filled square. |
| -8 | GKS\$K_MARKERTYPE_BOWTIE | Use a hollow bow tie. |
| -9 | GKS\$K_MARKERTYPE_SOLID_BOWTIE | Use a filled bow tie. |
| -10 | GKS\$K_MARKERTYPE_HGLASS | Use a hollow hourglass. |
| -11 | GKS\$K_MARKERTYPE_SOLID_HGLASS | Use a filled hourglass. |
| -12 | GKS\$K_MARKERTYPE_DIAMOND | Use a hollow diamond. |
| -13 | GKS\$K_MARKERTYPE_SOLID_DIAMOND | Use a filled diamond. |

NOTE

For all solidly filled markers, DEC GKS uses the current marker color index.

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$LIBRARY:GKSMSG.S.FOR'
C     Check for success.
      IF ( GKS$_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 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 the *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.

D.1 Implementation-Specific Errors

All 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 INQUIRE DEFAULT CHOICE DEVICE DATA in Chapter 11, 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 INQUIRE DEFAULT CHOICE DEVICE DATA to obtain the maximum choices available for your workstations, and then break your menu into two smaller menus.
- 8 **Echo area is too short for data in routine ******
- 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 ERROR LOGGING.
User Action: Make sure that the error code passed to ERROR LOGGING is one of the codes described in this appendix.
- 20 GKS not in proper state: GKS in the ERROR state in routine ****
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 EVALUATE TRANSFORMATION MATRIX and to ACCUMULATE TRANSFORMATION MATRIX to make sure that you passed valid transformation components. Also, make sure that you specified a transformation matrix to SET SEGMENT TRANSFORMATION or to INSERT SEGMENT.
- 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 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 than 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).

- 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 OPEN WORKSTATION.
- 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 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.
- 44 Error trying to save or restore VT340 color map in routine ****
DECGKS\$ _ERROR_NEG_44:
User Action: Submit an SPR.
- 45 Unable to decompose fill area or fill area set in routine ****
DECGKS\$ _ERROR_NEG_45:
User Action: Simplify the fill area by breaking it up into smaller fill areas until the error does not occur.
- 46 No default connection identifier for specified workstation type in routine ****
DECGKS\$ _ERROR_NEG_46:
User Action: Define the logical GKS\$CONID to be a valid GKS connection identifier.

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: Internal error.
User Action: Submit an SPR.
- 94 Internal GKS error: Corrupted segment memory in routine ****
DECGKS\$ _ERROR_NEG_94:
Fatal: Internal error.
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: Internal error.
User Action: Submit an SPR.
- 97 Internal GKS error: Insufficient buffer size for translated logical name in routine ****
DECGKS\$ _ERROR_NEG_97:
Fatal: Internal error.
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: Internal error.
User Action: Submit an SPR.

- 100 Internal GKS error: Device handler received unexpected input access in routine ****
DECGKS\$_ERROR_NEG_100:
Fatal: Internal error.
User Action: Submit an SPR.
- 150 Edge index is less than zero in routine ****
DECGKS\$_ERROR_NEG_150:
User Action: Make sure the edge index passed in the escape function GKS\$K_ESC_SET_EDGE_INDEX is valid.
- 151 Edge width scale factor is less than zero ****
DECGKS\$_ERROR_NEG_151:
User Action: Make sure the edge width scale factor passed in the escape function GKS\$K_ESC_SET_EDGE_WIDTH is valid.
- 154 A representation for the specified edge index has not been predefined on this workstation in routine ****
DECGKS\$_ERROR_NEG_154:
User Action: Check the index of the edge being inquired about to make sure it is predefined.
- 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.
- 162 Edge index is invalid in routine ****
DECGKS\$ _ERROR_NEG_162:
User Action: Make sure you use a valid edge index.

- 163 Specified edge type is not supported on this workstation in routine ****
DECGKS\$_ERROR_NEG_163:
User Action: Make sure you use a valid edge type.
- 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 cannot 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:
- 305 Error parsing GKS\$[wstype]_INPUT_DEVICES logical name die to syntax error in routine ****
GKS\$_ERROR_NEG_305:
User Action: Check the definition of the logical name with the syntax description in the Tektronix 41xx chapter in the *DEC GKS Device Specifics Reference Manual*.

-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.

D.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 3, Control Functions, in the *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 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 OPEN GKS or CLOSE WORKSTATION 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 `ACTIVATE WORKSTATION` or `CLOSE SEGMENT` before the DEC GKS state changes to `GKS$K_WSAC`.)
- 4 GKS not in proper state: GKS shall be in the state SGOP in routine ****
GKS\$_ERROR_4:
Error: You called a function unsupported in the current state.
User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function `CREATE SEGMENT` before the DEC GKS state changes to `GKS$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 `ACTIVATE WORKSTATION` before the DEC GKS state changes to `GKS$K_WSAC`.)
- 6 GKS not in proper state: GKS shall be in the state WSOP or in the state WSAC in routine ****
GKS\$_ERROR_6:
Error: You called a function unsupported in the current state.
User Action: Call the appropriate DEC GKS control function to change the current state. (You must call the function `OPEN WORKSTATION` before the DEC GKS state changes to `GKS$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 OPEN WORKSTATION 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 OPEN WORKSTATION before the DEC GKS state changes to GKS\$K_WSOP.)

D.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 INQUIRE WORKSTATION CONNECTION AND 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 OPEN WORKSTATION, 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 OPEN WORKSTATION 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 OPEN WORKSTATION, 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 OPEN WORKSTATION, 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 ACTIVATE WORKSTATION, 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 `ACTIVATE WORKSTATION` 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 `OPEN WORKSTATION`.
- 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 `OPEN WORKSTATION`.
- 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 OPEN WORKSTATION.
- 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 OPEN WORKSTATION.
- 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 OPEN WORKSTATION.
- 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 OPEN WORKSTATION.

- 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 OPEN WORKSTATION.
- 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 GENERALIZED DRAWING PRIMITIVE, 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 OPEN WORKSTATION. You can use INQUIRE WORKSTATION MAXIMUM NUMBERS to determine the maximum number of open workstations that DEC GKS supports.

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

GKS\$ _ERROR_43:

User Action: You must remove the function call to **ACTIVATE WORKSTATION**. You can use **INQUIRE WORKSTATION MAXIMUM NUMBERS** to determine the maximum number of active workstations that DEC GKS supports.

D.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: Make sure either 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: Make sure either 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 ($[0,1] \times [0,1]$), 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: Make sure either 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 INQUIRE DISPLAY SPACE
SIZE to determine the maximum X and Y values of the device
coordinate plane.

D.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 SET POLYLINE REPRESENTATION 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 SET POLYLINE REPRESENTATION 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 INQUIRE POLYLINE FACILITIES 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 SET POLYMARKER REPRESENTATION 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 SET POLYMARKER REPRESENTATION to define a representation for a given index or use another, predefined index value.
- 69 Specified marker type is equal to zero in routine ****
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 INQUIRE POLYMARKER FACILITIES to obtain a list of supported line types for a given workstation.
- 71 Marker size scale factor is less than zero in routine ****
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 SET TEXT REPRESENTATION 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 SET TEXT REPRESENTATION to define a representation for the index value or use another, predefined index value.
- 75 Text font is equal to zero in routine ****
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 SET FILL AREA REPRESENTATION 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 SET FILL AREA REPRESENTATION to define a representation for the given index value or pass another, predefined index value.
- 83 Specified fill area interior style is not supported on this workstation in routine ****
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 INQUIRE FILL AREA FACILITIES to obtain a list of supported interior styles for a given workstation.

- 84 Style (pattern or hatch) index is equal to zero in routine ****
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 INQUIRE FILL AREA FACILITIES 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 SET PATTERN REPRESENTATION 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 SET PATTERN REPRESENTATION 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 SET FILL AREA INTERIOR STYLE.
- 91 Dimensions of color array are invalid in routine ****
GKS\$ _ERROR_91:
Error: One or more of the arguments passed to 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 SET COLOR REPRESENTATION 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 SET COLOR REPRESENTATION 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 SET PICK IDENTIFIER 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.

D.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 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.

D.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 CREATE SEGMENT 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 CREATE SEGMENT 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.

D.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 INQUIRE 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 INQUIRE 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 either you can process reports from the queue until empty or 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 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 AWAIT EVENT is a real value between 0.0 and 356, 400, specified in the format described in the AWAIT EVENT function description in Chapter 7, 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.

D.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:

This error code is required by the standard, but DEC GKS will never generate this error.

D.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.

D.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.

D.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: Submit an SPR.

- 305 Input/Output error has occurred while receiving data from a workstation ****
GKS\$ _ERROR_305:
User Action: Submit an SPR.
- 306 Input/Output error has occurred during program library management ****
GKS\$ _ERROR_306:
User Action: Submit an SPR.
- 307 Input/Output error has occurred while reading workstation description table ****
GKS\$ _ERROR_307:
User Action: Submit an SPR.
- 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.

DEC GKS Metafile Structures (GKSM, CGM)

This appendix provides a brief overview of the internal format of GKSM and CGM metafiles. DEC GKS defines the workstations GKS\$K_GKSM_OUTPUT and GKS\$K_CGM_OUTPUT to use when creating metafiles. DEC GKS defines the workstation GKS\$K_GKSM_INPUT to use when reading metafiles. Remember that DEC GKS can create, but cannot interpret, CGM metafiles.

If you need to understand the GKSM metafile format in detail, refer to the GKS ANSI standard document. If you need to understand the CGM metafile encoding formats in detail, refer to the CGM standard ANSI X3.122-1986. All references to the CGM standard in this appendix refer to this standard document.

The following sections briefly describe GKSM and CGM metafiles.

E.1 GKSM Metafiles

The GKS standard defined the GKS metafile (GKSM) for the purpose of storing and retrieving information about the generation of a picture. The metafile can contain information about GKS output function calls from level 0 to level 2.

The design of the GKSM metafile structure defines a sequence of logical data items. The data items include information in both a clear text encoding and an unspecified binary format. The following sections describe the format and coding of the GKSM logical data items.

E.1.1 Data Format Information

The proposed standard ISO 6093 will describe the representation of integer and real number representations. This standard is not likely to be completed for quite some time. There is a movement for ISO 6093 to support use of a comma rather than a period in floating-point numbers. DEC GKS does not support this use of commas.

Integers are formatted in decimal ASCII characters in the output metafile. Floating-point numbers are formatted in the standard F- or E-Floating formats, decimal ASCII characters, depending upon their value.

The GKSM metafile allows four possible ways to represent integers and floating-point numbers, as follows:

- Both integer and floating-point numbers are specified by their character representations.
- Integer numbers are specified by their character representations. Floating-point numbers are represented as scaled integers.
- Both integer and floating-point numbers are specified by their internal binary representations.
- Integer numbers are specified by their internal binary representations. Floating-point numbers are represented as scaled integers.

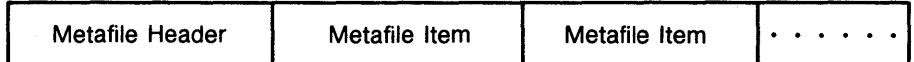
Remember that both integer and floating-point numbers are specified by their character representations.

GKSM metafiles also allow differing field length specifications for different fields of the metafile. The input workstation recognizes all the different field length specifications.

E.1.2 GKSM Structure

A GKSM metafile consists of a metafile header followed by metafile items. Each metafile item consists of an item header followed by item data. Figure E-1 illustrates this structure.

Figure E-1: GKSM Metafile Structure



ZK-5220-86

E.1.2.1 Metafile Header Structure

The metafile header contains 90 bytes. The bytes are divided into 13 fields as follows. Figure E-2 illustrates this structure.

Figure E-2: GKSM Metafile Header Structure



ZK-5221-86

Table E-1 describes the fields within the metafile header.

Table E-1: GKSM Metafile Header Fields

| Field | Size | Description |
|-------|----------|---|
| GKSM | 4 bytes | Containing string "GKSM". |
| N | 40 bytes | Containing name of author/installation. In DEC GKS, author is the process name at the time of metafile creation (16 bytes) and installation is "DEC GKS Version 2.0". |
| D | 8 bytes | Containing date (yy/mm/dd). |
| V | 2 bytes | Version number (01). |
| H | 2 bytes | Integer specifying how many bytes of the string "GKSM" occupy the beginning of each record (04). |
| T | 2 bytes | Length of item type indicator field (03). |

(continued on next page)

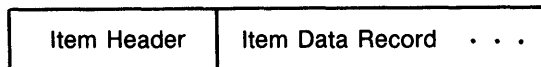
Table E-1 (Cont.): GKSM Metafile Header Fields

| Field | Size | Description |
|-------|----------|---|
| L | 2 bytes | Length of item data record length indicator field (08). |
| I | 2 bytes | Length of field for each integer in the item data record (08). |
| R | 2 bytes | Length of field for each real number in the item data record (14). |
| F | 2 bytes | Flag indicating if numbers are formatted as characters (1) or are stored in an internal binary format (2). DEC GKS value is 01. |
| RI | 2 bytes | Flag indicating if real numbers are stored as real numbers (01) or as scaled integers (02). DEC GKS value is 01. |
| ZERO | 11 bytes | Scaling information. Not used. |
| ONE | 11 bytes | Scaling information. Not used. |

E.1.2.2 Metafile Item Structure

There are several different types of metafile items. Each item consists of an item header and an item data record. The item header format is the same for all types of metafile items, but the item data record varies in length and format for each type of metafile item. Figure E-3 illustrates the structure of a metafile item.

Figure E-3: GKSM Metafile Item Structure

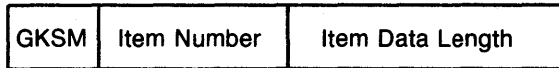


ZK-5222-86

E.1.2.3 Item Header Structure

Each item header contains 15 bytes, divided in three fields. Figure E-4 illustrates the item header structure.

Figure E-4: GKSM Metafile Item Header Structure



ZK-5223-86

Table E-2 presents the item header fields.

Table E-2: GKSM Metafile Item Header Fields

| Field | Size | Description |
|------------------|---------|---|
| GKSM | 4 bytes | Contains the string "GKSM". |
| Item Number | 3 bytes | Contains an integer identifying the item. |
| Item Data Length | 8 bytes | Contains an integer specifying the length, in bytes, of the item data record. |

E.1.2.4 Layout of Item Data Records

Each item data type, identified by a unique item number (an integer), has a specific format associated with it. Table E-3 lists the possible item numbers and their associated formats.

Table E-3: GKSM Metafile Data Record Fields

| Number | Format |
|--------|---|
| 0 | END ITEM—Last item of the metafile. No data record. |
| 1 | CLEAR WORKSTATION—For all active workstations. a) Integer, 0 = CONDITIONALLY or 1 = ALWAYS. |
| 2 | REDRAW ALL SEGMENTS ON WORKSTATION—No data record. |
| 3 | UPDATE WORKSTATION—For all active workstations. a) Integer, 0 = POSTPONE, 1 = PERFORM. |
| 4 | SET DEFERRAL STATE—a) Integer = deferral mode, 0 = ASAP, 1 = BNIG, 2 = BNIL, 3 = ASTI; b) Integer = regeneration mode, 0 = ALLOWED, 1 = SUPPRESSED. |

(continued on next page)

Table E-3 (Cont.): GKSM Metafile Data Record Fields

| Number | Format |
|---------------|--|
| 5 | MESSAGE—a) Integer = number of characters in string; b) string with specified number of characters. |
| 6 | ESCAPE—For all active workstations. a) Integer = function id, b) Integer = bytes of integer array d, c) Integer = bytes of real number array e, d) integer argument array, e) real argument array. |
| 11 | POLYLINE—a) Integer = N, Number of polymarkers, b) N pairs of real numbers. Each pair specifies the X and Y coordinates of a point as real numbers. |
| 12 | POLYMARKER—a) Integer = N, Number of points of the fill area, b) N pairs of real numbers. Each pair specifies the X and Y coordinates of a point as real numbers. |
| 13 | TEXT—a) Two real numbers specifying the starting position of string, b) Number N of characters in the string and c) N characters of the string. |
| 14 | FILL AREA—a) Integer = N, Number of points of the polyline, b) N pairs of real numbers. Each pair specifies the X and Y coordinates of a point as real numbers. |
| 15 | CELL ARRAY—a) Three pairs of X-Y coordinates points. First two points are specified in the function call, the third one is another corner, b) Integer = number of rows in array, c) Integer = number of columns in array, d) Integer array of color indexes stored row by row. |
| 16 | GDP—a) Integer = GDP identifier, b) Integer N = Number of points, c) Number of bytes of the integer array f, d) Number of bytes of the real array g, e) Array containing coordinate points, f) Array containing integer data, g) Array containing real data. |
| 21 | POLYLINE INDEX—a) Integer = polyline index. |
| 22 | LINETYPE—a) Integer = line type. |
| 23 | LINEWIDTH SCALE FACTOR—a) Real number = line width scale factor. |
| 24 | POLYLINE COLOR INDEX—a) Integer = polyline color index. |
| 25 | POLYMARKER INDEX—a) Integer = polymarker index. |
| 26 | MARKER TYPE—a) Integer = marker type. |
| 27 | MARKER SIZE SCALE FACTOR—a) Real number = marker size scale factor. |
| 28 | POLYMARKER COLOR INDEX—a) Integer = polymarker color index. |

(continued on next page)

Table E-3 (Cont.): GKSM Metafile Data Record Fields

| Number | Format |
|---------------|---|
| 29 | TEXT INDEX—a) Integer = text index. |
| 30 | TEXT FONT AND PRECISION—a) Integer = text font, b) Integer, 0 = STRING, 1 = CHAR, 2 = STROKE. |
| 31 | CHARACTER EXPANSION FACTOR—a) Real number = character expansion factor. |
| 32 | CHARACTER SPACING—a) Real number = character spacing. |
| 33 | TEXT COLOR INDEX—a) Integer = text color index. |
| 34 | CHARACTER VECTORS—a) Two real numbers specifying character height vector, b) Two real numbers specifying character width vector. |
| 35 | TEXT PATH—a) Integer 0 = RIGHT, 1 = LEFT, 2 = UP, 3 = DOWN. |
| 36 | TEXT ALIGNMENT—a) Integer = Horizontal component, 0 = NORMAL, 1 = LEFT, 2 = CENTER, 3 = RIGHT, b) Integer = Vertical component, 0 = NORMAL, 1 = TOP, 2 = CAP, 3 = HALF, 4 = BASE, 5 = BOTTOM. |
| 37 | FILL AREA INDEX—a) Integer = fill area index. |
| 38 | FILL AREA INTERIOR STYLE—a) Integer, 0 = HOLLOW, 1 = SOLID, 2 = PATTERN, 3 = HATCH. |
| 39 | FILL AREA STYLE INDEX—a) Integer = fill area style index. |
| 40 | FILL AREA COLOR INDEX—a) Integer = fill area color index. |
| 41 | PATTERN SIZE—a) Two real numbers specifying pattern width as X and Y components, b) Two real numbers specifying pattern height as X and Y components. |
| 42 | PATTERN REFERENCE POINT—a) Two real numbers specifying pattern reference point. |
| 43 | ASPECT SOURCE FLAGS—a) 13 integers specifying aspect source flags. 0 = BUNDLED, 1 = INDIVIDUAL. |
| 44 | PICK IDENTIFIER—a) Integer = pick identifier. |
| 51 | POLYLINE REPRESENTATION—a) Integer = polyline index; b) Integer = line type; c) Real = line width scale factor; d) polyline color index. |
| 52 | POLYMARKER REPRESENTATION—a) Integer = polymarker index; b) Integer = marker type; c) Real = marker size scale factor; d) polymarker color index. |

(continued on next page)

Table E-3 (Cont.): GKSM Metafile Data Record Fields

| Number | Format |
|---------------|--|
| 53 | TEXT REPRESENTATION —a) Integer = text index; b) Integer = text font; c) Integer, text precision, 0 = STRING, 1 = CHAR, 2 = STROKE; d) Real = character expansion factor; e) Real = character spacing; f) text color index. |
| 54 | FILL AREA REPRESENTATION —a) Integer = fill area index; b) Integer = interior style, 0 = HOLLOW, 1 = SOLID, 2 = PATTERN, 3 = HATCH; c) Integer = style index; d) Integer = fill area color index. |
| 55 | PATTERN REPRESENTATION —a) Integer = pattern index; b) Integer = number of columns in color array; c) Integer = number of rows; d) Integer array of the number of columns and rows specified containing color index values. |
| 56 | COLOR REPRESENTATION —a) Integer = color index, b) Three real numbers specifying red, green, and blue intensities, respectively. |
| 61 | CLIPPING RECTANGLE —a) Four real numbers specifying XMIN, XMAX, YMIN, and YMAX, respectively. |
| 71 | WORKSTATION WINDOW —a) Four real numbers specifying XMIN, XMAX, YMIN, and YMAX, respectively. |
| 72 | WORKSTATION VIEWPORT —a) Four real numbers specifying XMIN, XMAX, YMIN, and YMAX, respectively. |
| 81 | CREATE SEGMENT —a) Integer = segment name. |
| 82 | CLOSE SEGMENT —No data record. |
| 83 | RENAME SEGMENT —a) Integer = old name; b) Integer = new name. |
| 84 | DELETE SEGMENT —a) Integer = segment name. |
| 91 | SET SEGMENT TRANSFORMATION —a) Integer = segment name; b) Six real numbers specifying the transformation matrix values. |

(continued on next page)

Table E-3 (Cont.): GKSM Metafile Data Record Fields

| Number | Format |
|---------------|---|
| 92 | SET VISIBILITY—a) Integer = segment name; b) Integer = visibility, 0 = VISIBLE, 1 = INVISIBLE. |
| 93 | SET HIGHLIGHTING—a) Integer = segment name; b) Integer, highlighting, 0 = NORMAL, 1 = HIGHLIGHTED. |
| 94 | SET SEGMENT PRIORITY—a) Integer = segment name; b) Real = priority. |
| 95 | SET SEGMENT DETECTABILITY—a) Integer = segment name; b) Integer, detectability, 0 = UNDETECTABLE, 1 = DETECTABLE. |
| > 100 | User Item—User-defined number of bytes. |

E.1.3 GKSM Physical File Organization

The GKSM metafile has varying length record format, with a limit on the maximum record size of 4096 bytes. This file is of sequential organization.

Each metafile item occupies two or more RMS records: one for the item header and one or more for the item data record. The metafile header occupies one RMS record. The record item data record occupies at least one RMS record. If the item data record has a length greater than 4096 bytes, then the data record is split into two or more RMS records.

E.2 Computer Graphics Metafiles (CGM)

DEC GKS supports the Computer Graphics Metafile (CGM) format for use in creating metafiles. To create a CGM metafile, open and activate a workstation of type GKS\$K_CGM_OUTPUT. (Remember that DEC GKS cannot interpret CGM metafiles.)

The CGM standard defines a metafile as being the capture of static picture definitions for many types of graphical applications, including DEC GKS programs. Since the CGM standard provides functionality for many types of graphics applications (not just GKS), certain GKS functionalities may not be supported by the CGM format and certain CGM capabilities cannot be used by a GKS program. When you create a CGM metafile using DEC GKS, CGM records only those features supported by the CGM format. See Section E.2.2 for detailed information concerning the relationship between DEC GKS and CGM picture storage.

The CGM standard defines three *encodings*. Encodings are formats used to store data within the metafile. The data types and values used to store information within the CGM metafile vary depending on the encoding you use to create the metafile. The following list presents the three CGM encodings:

- Character encoding—A format whose physical file takes a minimal amount of storage
- Binary encoding—A format easily stored and read by many types of machine architectures and applications
- Clear text encoding—A format that can easily be read or edited by application programmers who wish to use the metafile

DEC GKS supports two of the three formats: the character and clear text encodings. The following bit mask is valid for use with the GKS\$K_CGM_OUTPUT workstation:

%x000n0007

The value in the first part (000n) specifies the desired encoding. The value in the second part is the hexadecimal value of the GKS\$K_CGM_OUTPUT workstation type (%d7).

The possible values for n include the following:

| n | Encoding |
|---|----------------------|
| 2 | Character encoding. |
| 4 | Clear text encoding. |

If you choose, you can use bitmask constant values within your program to specify an encoding, as follows:

```

      .
      .
      .
      CALL GKSS$OPEN_WS( WS_ID, 'CGM_METAFILE.TXT',
* GKSS$K_CGM_OUTPUT .OR. GKSS$M_CHARACTER_ENCODING )
C    or,
      .
      .
      .
      CALL GKSS$OPEN_WS( WS_ID, 'CGM_METAFILE.TXT',
* GKSS$K_CGM_OUTPUT .OR. GKSS$M_CLEAR_TEXT_ENCODING )
      .
      .
  
```

For more information concerning constants, refer to Appendix B, DEC GKS Constants.

The following subsections describe the following topics:

- CGM structure
- Supported encodings
- Element descriptions
- Differences between CGM and DEC GKS graphical facilities
- CGM physical file organization

E.2.1 CGM Structure

The CGM standard defines three components within a metafile, as shown in Figure E-5.

Figure E-5: CGM Components



ZK-5847-HC

The metafile descriptor component contains data relevant to the functional capabilities required to interpret that metafile. For instance, this component can contain data such as a metafile descriptive string or title, the version number of the CGM standard used by the implemented CGM interpreter, the date of the metafile creation, and so forth. (Remember that the format of this data depends on the encoding you choose.)

The metafile defaults component contains data relevant to all the picture definitions contained in the metafile. For instance, this component can contain data such as the virtual display coordinate (VDC) boundary (this corresponds to the DEC GKS normalized device coordinate plane), attribute settings, and so forth.

Each metafile picture component contains data relevant to pictures created by a DEC GKS program. Since the DEC GKS standard does not define its graphical output in terms of pictures, the CGM interpreter must use the *display surface empty* and *new frame necessary at update* entries in the DEC GKS state list to determine when a picture ends and when a new picture begins. (See Section E.2.2 for more information concerning the differences in terminology between DEC GKS and CGM.)

CGM files contain components called *elements*. Each element serves a distinct purpose, and depending on its functionality, includes applicable data. CGM specifies an element by providing the encoding-dependent *opcode* and argument data. The opcode is a character or series of characters that specify the beginning of a distinct element.

The following list describes the types of elements in a CGM metafile:

| Category | Description |
|------------------------------|---|
| Delimiter Elements | Separate components within the metafile. |
| Metafile Descriptor Elements | Describe the functional content and unique characteristics of the CGM metafile. |
| Picture Descriptor Elements | Define the limits of the virtual device coordinates (VDCs) and the parameter modes for the attribute elements. |
| Control Elements | Specify size and precision of VDC coordinates, and format descriptions of the CGM elements. |
| Graphical Primitive Elements | Describe the geometric objects in the picture. |
| Attribute Elements | Describe the various appearances of the graphical elements. |
| Escape Elements | Describe device- and system-specific functionality. |
| External Elements | Pass information not needed for the creation of a picture (for instance, a message sent to the user of the metafile). |

Although CGM defines many data types that correspond to graphical data (for instance, an index data type for bundle index specifications), there are a few data types from which all others are derived. The following list presents all the basic data types of information contained in a CGM metafile:

| Data Type | Description |
|------------------|---|
| Integer | Integer values such as bundle indexes, integer data, and so forth. |
| Real | Real values such as VDC distance values; red, green, and blue color intensities; coordinate points; and so forth. |
| String | Character strings such as metafile description titles and string data. |
| Point List | Lists of points such as polyline points, polymarker points, and so forth. |

The characters used to specify an opcode and its data are encoding specific. The following subsections provide a brief overview of the two supported encodings.

E.2.2 Differences Between GKS and CGM

Since CGM is designed to format files for many types of graphical applications, there is no unique relationship between CGM and GKS. If CGM does not support a graphical facility of DEC GKS, the CGM metafile does not attempt to simulate such a facility. If the CGM metafile structure supports a graphical facility unsupported by DEC GKS, then a DEC GKS program will not generate those unsupported CGM elements.

As mentioned, DEC GKS does not define its graphical output in terms of pictures, as does CGM. Consequently, the CGM interpreter must determine what constitutes a new CGM picture definition.

The following list presents the DEC GKS graphical facilities unsupported by CGM:

- CGM does not support the changing of workstation transformations. Workstation transformations cause the CGM interpreter to start a new picture definition.
- A call to CLEAR WORKSTATION causes the CGM interpreter to start a new picture definition.
- CGM has no elements that correspond to the DEC GKS SET_primitive_REPRESENTATION functions.
- CGM does not support the DEC GKS segment functions.

The following list presents the CGM facilities that are unsupported by DEC GKS:

- DEC GKS does not support the disjoint polyline or the polygon set as primitives.

- DEC GKS does not support the CGM higher-level primitives (circle, rectangle, ellipse) as primitives, but can store them as generalized drawing primitives instead.
- DEC GKS does not support the extended text processing facilities of CGM (such as named fonts, changing character sets, appended text, restricted text).
- DEC GKS does not support the fill area edge, auxiliary color, and direct color specification CGM facilities.

E.2.3 Character Encoding

The CGM character encoding provides a character code for each of the element opcodes, and provides storage-saving methods for storing argument data. This is the most storage-efficient encoding.

The CGM character encoding specifies either one or two 7-bit ASCII characters that correspond to each element opcode. For instance, for the BEGIN METAFILE opcode, CGM places the two ASCII characters 3/0 and 2/0 into the metafile. (Table E-4 lists the ASCII notations that correspond to each of the element opcodes.)

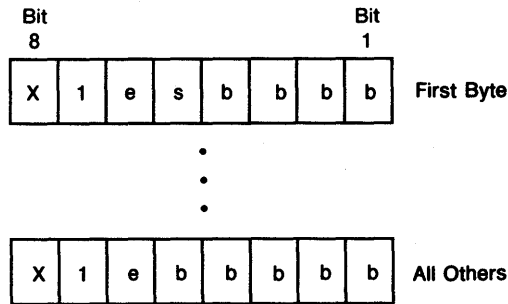
To translate the opcode notation into an ASCII value that corresponds to a character, multiply the first number by the value 16, and add the product to the number after the slash character (/). So, the notation 3/0 corresponds to ASCII value 48. For many 7-bit ASCII charts, the first number specifies the chart column and the number following the slash indicates the chart row. So, to find the ASCII character that corresponds to 3/0, look in column 3, row 0.

To encode data, the CGM character encoding uses a basic format. The basic format applies to the following CGM data types:

- Enumerated types
- Color indexes
- Indexes other than color indexes
- Integers
- Real numbers

Figure E-6 presents the CGM character encoding basic format.

Figure E-6: CGM Basic Data Encoding Format

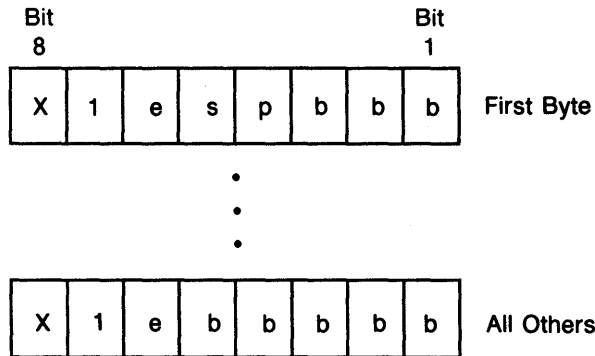


7K-5848-HC

CGM encodes each type of data in one or more bytes. Each byte contains bits that specify data values. In Figure E-6, bit X is the value 0. Bit e (the sixth bit) is the extension flag. This flag contains the value 1 in all bytes except the last byte in the data specification. In the last byte, the flag contains the value 0. Bit s (the fifth bit of the first byte) is the sign bit (the value 0 for nonnegative numbers; the value 1 for negative numbers). Bits labeled b specify the numeric value in binary. The most significant bits are in the first byte and the least significant bits are in the last byte.

CGM encodes each real number as an integer mantissa followed by an exponent. The exponent is the power of 2 by which the integer mantissa is to be multiplied. Figure E-7 illustrates how CGM uses the basic format to encode real numbers.

Figure E-7: CGM Basic Encoding Format for Real Numbers



ZK-5849-HC

Bit *e* is the extension bit and bit *s* is the sign bit. Bit *p* is the “exponent follows” bit, which is always the value 1. The last three bits in the first byte contain the exponent; the remaining bits are the mantissa.

The DEC GKS CGM character encoding scheme uses the *displacement mode* to encode point list data. Displacement mode specifies pairs of VDC values that are the X and Y delta values relative to the last specified point.

CGM codes character strings as sequences of bytes starting with the designated OPEN CHARACTER STRING character and ending with the STRING TERMINATOR character.

The CGM encoding scheme defines many ways to encode data. For complete information concerning character encoding, refer to the CGM standard documentation.

E.2.4 Clear Text Encoding

The CGM clear text encoding provides a character string for each of the element opcodes, and numbers and delimiters to specify argument data. Using this type of encoding, you can easily type or edit the metafile.

For example, this encoding represents the BEGIN METAFILE opcode as the character string BEGMF. DEC GKS uses the semicolon (;) to separate element opcodes. (Table E-4 lists the character strings that correspond to each of the element opcodes.)

DEC GKS specifies integers as numbers and separates the decimal portion of real numbers using a period (.). If you edit a clear text encoded metafile, you can insert comments delimited by percent signs (%). DEC GKS uses the single quote character to delimit strings ('). The DEC GKS CGM clear text encoding mechanism for point lists is as follows:

- DEC GKS encloses each pair of points in parentheses (()).
- DEC GKS separates each point specification, within a pair, using a comma (,).
- DEC GKS separates the parenthetical point groupings using spaces.

For more information concerning clear text encoding, refer to the CGM standard documentation.

E.2.5 CGM Element Descriptions

Table E-4 lists the opcodes required for each of the CGM elements. In the column labeled Opcode, the first opcode listed is the 7-bit ASCII notation of the character(s) used by the character encoding, and the second opcode listed is the character string used by the clear text encoding.

Table E-4: CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|---------------------|-----------------------|---|
| BEGIN METAFILE | 3/0 2/0 BEGMF | A string value specifying the metafile identifier. |
| END METAFILE | 3/0 2/1 ENDMF | No data required. |
| BEGIN PICTURE | 3/0 2/2 BEGPIC | A string value that is the picture identifier. |
| BEGIN PICTURE BODY | 3/0 2/3 BEGPICBODY | No data required. |
| END PICTURE | 3/0 2/4 ENDPIC | No data required. |
| METAFILE VERSION | 3/1 2/0 MFVERSION | An integer value corresponding to the version of the CGM standard being used. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|-----------------------------------|-----------------------------------|--|
| METAFILE DESCRIPTION | 3/1 2/1 MFDESC | A string value that is a description of the metafile contents. |
| VDC TYPE | 3/1 2/2 VDCTYPE | An enumerated type specifying the virtual display coordinate type, which corresponds to the DEC GKS NDC plane (INTEGER, REAL). |
| INTEGER PRECISION | 3/1 2/3 INTEGERPREC | A value (of an encoding-dependent data type) that specifies the integer precision. |
| REAL PRECISION | 3/1 2/4 REALPREC | A value or values (of an encoding-dependent data type) that specify the subfields of the real number precision. |
| INDEX PRECISION | 3/1 2/5 INDEXPREC | A value (of an encoding-dependent data type) that specifies the precision of an index into a bundle table. |
| COLOR PRECISION | 3/1 2/6 COLRPREC | A value (of an encoding-dependent data type) that specifies the subfields of the precision of red, green, and blue color intensity values. |
| COLOR INDEX PRECISION | 3/1 2/7 COLRINDEXPREC | A value (of an encoding-dependent data type) that specifies the precision of an index into a color table. |
| MAX COLOR INDEX | 3/1 2/8 MAXCOLRINDEX | A positive nonzero integer that is the maximum color index value. |
| COLOR VALUE EXTENT | 3/1 2/9 COLRVALUEEXT | Two sets of red, green, and blue intensity real values that are the minimum and maximum color values. |
| METAFILE ELEMENT LIST | 3/1 2/10 MFELEMLIST | A value (of an encoding-dependent data type) containing a list of all application-specific elements contained in this metafile. |
| BEGIN DEFAULTS REPLACEMENT | 3/1 2/11 BEGMFDEFAULTS | Control, picture descriptor, and attribute element list of the same format as described for the corresponding elements. |
| END DEFAULTS REPLACEMENT | 3/1 2/12 ENDMFDEFAULTS | No data required. |
| FONT LIST | 3/1 2/13 FONTLIST | A list of strings that assigns a font index value, beginning with the value 1, to each font in the list. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|----------------------------|---------------------------|--|
| CHARACTER SET LIST | 3/1 2/14 CHARSETLIST | A list of information that specifies up to five of the supported character sets (from ISO 2022). Each pair consists of an enumerated value (such as <94-character>) followed by a short string describing the "tail end" of designating escape sequences for that set (such as 4/1). |
| CHARACTER CODING ANNOUNCER | 3/1 2/15 CHARCODING | An enumerated type specifying the code extension technique assumed by the metafile creator (BASIC 7-BIT, BASIC 8-BIT, EXTENDED 7-BIT, EXTENDED 8-BIT). |
| SCALING MODE | 3/2 2/0 SCALEMODE | An enumerated type value and a real value. The enumerated value specifies either ABSTRACT or METRIC. If ABSTRACT, then the VDC space is correctly displayed at any size. If METRIC, then the real value is the workstation surface distance in millimeters that corresponds to a single VDC point. |
| COLOR SELECT MODE | 3/2 2/1 COLRMODE | An enumerated type that specifies color selection support (INDEXED, DIRECT); DIRECT specifies that color selections are by red, green, and blue intensity value. |
| LINE WIDTH SPEC MODE | 3/2 2/2 LINEWIDTHMODE | An enumerated type specifying line width. ABSOLUTE specifies a measurement in VDC points; SCALED specifies a scale factor to be applied to a workstation-dependent nominal width. |
| MARKER SIZE SPEC MODE | 3/2 2/3 MARKERSIZEMODE | An enumerated type specifying marker size. ABSOLUTE specifies a measurement in VDC points; SCALED specifies a scale factor to be applied to a workstation-dependent nominal size. |
| EDGE WIDTH SPEC MODE | 3/2 2/4 EDGEWIDTHMODE | An enumerated type specifying edge width. ABSOLUTE specifies a measurement in VDC points; SCALED specifies a scale factor to be applied to a workstation-dependent nominal width. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|---------------------|---------------------------|---|
| VDC EXTENT | 3/2 2/5 VDCEXT | Two sets of points that define opposite corners of a rectangular area of the VDC. This establishes the positive and negative directions for the VDC plane. |
| BACKGROUND COLOR | 3/2 2/6 BACKCOLR | A set of red, green, and blue intensity values for the background color. |
| VDC INTEGER PREC | 3/3 2/0 VDCINTEGERPREC | A value (of an encoding-dependent data type) containing the precision for integers used to designate VDC points. |
| VDC REAL PREC | 3/3 2/1 VDCREALPREC | A value (of an encoding-dependent data type) containing the precision for real numbers used to designate VDC points. |
| AUXILIARY COLOR | 3/3 2/2 AUXCOLR | An integer auxiliary color index used to color a primitive in transparency mode. |
| TRANSPARENCY | 3/3 2/3 TRANSPARENCY | An enumerated type that specifies whether the transparency color is used to draw subsequent primitives (OFF, ON). |
| CLIP RECTANGLE | 3/3 2/4 CLIPRECT | Two VDC point values specifying the clipping rectangle range. |
| CLIP INDICATOR | 3/3 2/5 CLIP | An enumerated type specifying the clipping status (OFF, ON). |
| POLYLINE | 2/0 INCRLINE | A set of points, each consecutive point connected to the last by a line. |
| DISJOINT POLYLINE | 2/1 INCRDISJTLINE | A set of points, the first connected to the second, the third connected to the fourth, and so forth, leaving spaces in the line. |
| POLYMARKER | 2/2 INCRMARKER | A set of points, a special character drawn at each point. |
| TEXT | 2/3 TEXT | A VDC starting point, an enumerated flag, and a string. If the flag is NOT FINAL, then you can specify elements to change the text attributes between this element and the APPEND TEXT element. If the flag is FINAL, then the string is the entire string to be displayed. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|-----------------------------|------------------------|--|
| RESTRICTED TEXT | 2/4 RESTRTEXT | Two VDC values that are the height and width vectors, a VDC starting point, an enumerated flag (as described in TEXT), and a string. The text must be contained within the parallelogram created using the starting point and height and width vectors. |
| APPEND TEXT | 2/5 APNDTEXT | An enumerated flag value (as described in TEXT) and a string. The flag value determines whether you can specify other elements between this element and a subsequent APPEND element. |
| POLYGON | 2/6 INCRPOLYGON | A series of VDC points specifying a polygon. |
| POLYGON SET | 2/7 INCRPOLYGONSET | A flagged point list, each list item containing a point and an enumerated flag. Each point is connected to the subsequent point or to the current closure point, but not to both. The flag can be one of the edge values INVISIBLE, VISIBLE, CLOSE INVISIBLE, CLOSE VISIBLE. |
| CELL ARRAY | 2/8 CELLARRAY | Two diagonal VDC corner points, a third corner point clockwise between the starting point and diagonal points, a two-dimensional list of either color indexes or intensity values, and local color precision (format determined by the encoding). |
| GDP | 2/9 GDP | An integer GDP identifier, a point list, and a data record (used in an interpreter-dependent manner). |
| RECTANGLE | 2/10 RECT | Two VDC points specifying the starting point and the diagonal point of the rectangle. |
| CIRCLE | 3/4 2/0 CIRCLE | A VDC center point and a VDC distance vector used as the radius. |
| CIRCLE ARC 3 POINT | 3/4 2/1 ARC3PT | A starting point, an intermediate point, and an end point. |
| CIRCLE ARC 3 POINT CLOSE | 3/4 2/2 ARC3PTCLOSE | A starting point, an intermediate point, an end point, and an enumerated close flag (PIE, CHORD). |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|---------------------------|--------------------------|---|
| CIRCULAR ARC CENTER | 3/4 2/3 ARCCTR | A center point, a distance X and Y vector for the starting point, a distance X and Y vector for the end point, and a VDC radius distance vector. |
| CIRCULAR ARC CENTER CLOSE | 3/4 2/4 ARCCTRCLOSE | A center point, a distance X and Y vector for the starting point, a distance X and Y vector for the end point, a VDC radius distance vector, and an enumerated close flag (PIE, CHORD). |
| ELLIPSE | 3/4 2/5 ELLIPSE | A center point and an endpoint for each conjugate diameter. |
| ELLIPTICAL ARC | 3/4 2/6 ELLIPARC | A center point, two endpoints on each conjugate diameter, a distance X and Y vector for the starting point, and a distance X and Y vector for the end point. |
| ELLIPTICAL ARC CLOSE | 3/4 2/7 ELLIPARCCLOSE | A center point, two endpoints on each conjugate diameter, a distance X and Y vector for the starting point, a distance X and Y vector for the end point, and an enumerated close flag (PIE, CHORD). |
| LINE BUNDLE INDEX | 3/5 2/0 LINEINDEX | Integer index value into the line bundle table. |
| LINE TYPE | 3/5 2/1 LINETYPE | Integer line type value. |
| LINE WIDTH | 3/5 2/2 LINEWIDTH | Either a VDC absolute value or a real scale specification. |
| LINE COLOR | 3/5 2/3 LINECOLR | Either an integer index value or a set of red, green, and blue real values. |
| MARKER BUNDLE INDEX | 3/5 2/4 MARKERINDEX | An integer index value into the polymarker bundle table. |
| MARKER TYPE | 3/5 2/5 MARKERTYPE | An integer value specifying a marker type. |
| MARKER SIZE | 3/5 2/6 MARKERSIZE | Either a VDC absolute value or a real scale specification. |
| MARKER COLOR | 3/5 2/7 MARKERCOLR | Either an integer index value or a set of red, green, and blue real values. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|-------------------------------|-----------------------------|---|
| TEXT BUNDLE INDEX | 3/5 3/0 TEXTINDEX | An integer value that is a pointer into the text bundle table. |
| TEXT FONT INDEX | 3/5 3/1 TEXTFONTINDEX | An integer index value associated with a previously specified font. |
| TEXT PRECISION | 3/5 3/2 TEXTPREC | An enumerated type (STRING, CHARACTER, STROKE). |
| CHARACTER EXPANSION FACTOR | 3/5 3/3 CHAREXPAN | A nonnegative real number specifying the height-to-width ratio. |
| CHARACTER SPACING | 3/5 3/4 CHARSPACE | A real value specifying character spacing. |
| TEXT COLOR | 3/5 3/5 TEXTCOLR | Either a color index integer or a set of red, green, and blue intensity values. |
| CHARACTER HEIGHT | 3/5 3/6 CHARHEIGHT | A VDC value specifying character height. |
| CHARACTER ORIENTATION | 3/5 3/7 CHARORI | A pair of X and Y directional vector values (VDC) that define which way is up, and a pair of X and Y directional vector values (VDC) that define the text base. |
| TEXT PATH | 3/5 3/8 TEXTPATH | An enumerated type value that determines the text path (RIGHT, LEFT, UP, DOWN). |
| TEXT ALIGNMENT | 3/5 3/9 TEXTALIGN | An enumerated type specifying horizontal alignment (NORMAL HORIZONTAL, LEFT, CENTRE, RIGHT, CONTINUOUS HORIZONTAL), an enumerated type specifying vertical alignment (NORMAL VERTICAL, TOP, CAP, HALF, BASE, BOTTOM, CONTINUOUS VERTICAL), and two real values specifying continuous horizontal and vertical alignments, that align the string with a coordinate outside its text extent. |
| CHARACTER SET INDEX | 3/5 3/10 CHARSETINDEX | An integer index value that chooses a previously specified character set. |
| ALTERNATE CHARACTER SET INDEX | 3/5 3/11 ALTCHARSETINDEX | An integer index value that chooses a previously specified character set. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|-----------------------------|-------------------------------|--|
| FILL BUNDLE INDEX | 3/6 2/0 FILLINDEX | An integer value that points into the fill area bundle table. |
| INTERIOR STYLE | 3/6 2/1 INTSTYLE | An enumerated type that specifies interior fill area style (HOLLOW, SOLID, PATTERN, HATCH, EMPTY). |
| FILL COLOR | 3/6 2/2 FILLCOLR | Either an integer color index value or a set of red, green, and blue intensity values. |
| HATCH INDEX | 3/6 2/3 HATCHINDEX | An integer value that specifies a hatch style. |
| PATTERN INDEX | 3/6 2/4 PATINDEX | An integer value that specifies a pattern type. |
| EDGE BUNDLE INDEX | 3/6 2/5 EDGEINDEX | An integer value that points into the edge bundle table. |
| EDGE TYPE | 3/6 2/6 EDGETYPE | An integer value that specifies the edge type. |
| EDGE WIDTH | 3/6 2/7 EDGEWIDTH | Either an absolute edge width specified in a VDC value, or an edge width scale factor. |
| EDGE COLOR | 3/6 2/8 EDGECOLR | Either an integer color index value or a set of red, green, and blue intensity values. |
| EDGE VISIBILITY | 3/6 2/9 EDGEVIS | An enumerated value specifying edge visibility (OFF, ON). |
| FILL REFERENCE POINT | 3/6 2/10 FILLREFPT | A real value specifying the fill area reference point. |
| PATTERN TABLE | 3/6 2/11 PATTABLE | An integer value specifying the placement of this pattern in the pattern table, a two-dimensional list of either color indexes or intensity values, and local color precision (format determined by the encoding). |
| PATTERN SIZE | 3/6 2/12 PATSIZE | Two VDC values that specify the X and Y components of the height distance vector, and two VDC values that specify the X and Y components of the width distance vector. |

(continued on next page)

Table E-4 (Cont.): CGM Element Descriptions

| Element Name | Opcode | Argument Data Description |
|--------------------------|----------------------|---|
| COLOR TABLE | 3/6 3/0 COLRTABLE | An integer that specifies a pointer into the bundle table where the first color value is placed, and a list of sets of red, green, and blue intensity values used to fill the table. |
| ASPECT SOURCE FLAGS | 3/6 3/1 ASF | A list of pairs of enumerated ASF type values and ASF values (INDIVIDUAL, BUNDLED). |
| ESCAPE | 3/7 2/0 ESCAPE | An integer function identifier, and a data record (implementation-dependent use). |
| MESSAGE | 3/9 2/1 MESSAGE | An enumerated type specifying the action flag that determines whether the application requires some action by the user before resuming application execution (NO ACTION, ACTION), and the text string containing the message. |
| APPLICATION DATA | 3/7 2/1 APPLDATA | An integer identifier, and a data record, both to be used in an application-dependent manner that does not affect the picture being generated. |
| OPEN CHARACTER STRING | 1/11 5/8 ' | A character that signifies the beginning of a character string. NOTE: This character is not an opcode. It usually follows an opcode that requires string data. |
| STRING TERMINATOR | 1/11 5/12 ' | A character that signifies the end of a character string. NOTE: This character is not an opcode. It usually follows an opcode that requires string data. |

E.2.5.1 CGM Encoding Examples

Example E-1 presents a simple DEC GKS program.

Example E-1: CGM Metafile Creation

```
IMPLICIT NONE
INTEGER WS_ID, GKS$K_VT240, GKS$K_CONID_DEFAULT
REAL X_ARRAY( 2 ), Y_ARRAY( 2 )
DATA X_ARRAY /0.0, 1.0/
DATA Y_ARRAY /0.5, 0.5/
DATA WS_ID / 1 /, GKS$K_VT240 / 13 /,
* GKS$K_CONID_DEFAULT / 0 /

CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )
CALL GKS$ACTIVATE_WS( WS_ID )

CALL GKS$POLYLINE( 2, X_ARRAY, Y_ARRAY )

CALL GKS$DEACTIVATE_WS( WS_ID )
CALL GKS$CLOSE_WS( WS_ID )
CALL GKS$CLOSE_GKS()
END
```

The following listing presents the clear text encoded CGM file produced by Example E-1 (you need to define the logical GKS\$WSTYPE to be %x00040007 to specify the clear text encoding).

```
BEGMF 'CGM_OUTPUT_FILE.CGM';
MFVERSION 1;
MFDESC 'DEC GKS output 11/19/86';
MFELEMLIST 'DRAWINGPLUS';
VDCTYPE REAL;
BEGMFDEFAULTS;
ALTCHARSETINDEX 2;
ENDMFDEFAULTS;
FONTLIST 'DEC GKS Stroke -1' 'DEC GKS Stroke -2' 'DEC GKS Stroke -3'
'DEC GKS Stroke -4' 'DEC GKS Stroke -5' 'DEC GKS Stroke -6'
'DEC GKS Stroke -7' 'DEC GKS Stroke -8' 'DEC GKS Stroke -9'
'DEC GKS Stroke -11' 'DEC GKS Stroke -12' 'DEC GKS Stroke -13'
'DEC GKS Stroke -14' 'DEC GKS Stroke -15' 'DEC GKS Stroke -16'
'DEC GKS Stroke -17' 'DEC GKS Stroke -18' 'DEC GKS Stroke -19'
'DEC GKS Stroke -20' 'DEC GKS Stroke -21' 'DEC GKS Stroke -22'
'DEC GKS Stroke -23';
CHARSETLIST STD94 'B';
CHARCODING BASIC8BIT;
INTEGERPREC -221646135 1870427260;
REALPREC -99999.992188 99999.992188 7;
INDEXPREC -221646135 1870427260;
COLRPREC 31;
COLRINDEXPREC 31;
BEGPIC '10:54:34.93';
SCALEMODE ABSTRACT 0.000000;
COLRMODE INDEXED;
LINEMODE SCALED;
MARKERSIZEMODE SCALED;
```

```

EDGEWIDTHMODE SCALED;
VDCEXT (0.000000,0.000000) (1.000000,1.000000);
BACKC OLR 0 0 0;
BEGPICBODY;
CLIPRECT (0.000000,0.000000) (1.000000,1.000000);
CHARHEIGHT 0.000000;
CHARORI 0.000000 0.000000 0.000000 0.000000;
PATSIZE 0.000000 0.000000 0.000000 0.000000;
FILLREFPT (0.000000,0.000000);
CLIPRECT (0.000000,0.000000) (1.000000,1.000000);
ASF LINETYPE INDIV LINEWIDTH INDIV LINECOLR INDIV MARKERTYPE
INDIV MARKERSIZE INDIV MARKERCOLR INDIV TEXTPREC
INDIV TEXTFONTINDEX INDIV CHAREXP INDIV CHARSPACE INDIV TEXTCOLR INDIV
INTSTYLE INDIV PATINDEX INDIV HATCHINDEX INDIV FILLCOLR IN DIV;
LINEINDEX 1;
LINETYPE 1;
LINEWIDTH 1.000000;
LINECOLR 1;
INCRLINE (0.000000,0.500000) (1.000000,0.000000);
TEXTINDEX 1;
CHARSETINDEX 1;
TEXTFONTINDEX 1;
TEXTPREC STRING;
CHAREXPAN 1.000000;
TEXTCOLR 1;
CHARHEIGHT 0.010000;
CHARORI 0.000000 0.010000 0.010000 0.000000;
MARKERINDEX 1;
MARKERTYPE 3;
MARKERSIZE 1.000000;
MARKERCOLR 1;
FILLINDEX 1;
FILLCOLR 1;
PATSIZE 0.000000 1.000000 1.000000 0.000000;
ENDPIC;
ENDMF;

```

The following listing presents the character-encoded CGM file produced by Example E-1 (you need to define the logical GKS\$WSTYPE to be %x0020007 to specify the character encoding). The question marks (?) in the data represent the ASCII escape character.

```

0 ?\?XZ.CGM_CHAR?\1 A1!\?XDEC GKS output 11/19/86?\1*\?XA?\1"A1+5;B1,-
?XDEC GKS Stroke -1?\?XDEC GKS Stroke -2?\?XDEC GKS Stroke -3?\
?XDEC GKS Stroke -4?\?XDEC GKS Stroke -5?\?XDEC GKS Stroke -6?\
?XDEC GKS Stroke -7?\?XDEC GKS Stroke -8?\?XDEC GKS Stroke -9?\
?XDEC GKS Stroke -11?\?XDEC GKS Stroke -12?\?XDEC GKS Stroke -13?\
?XDEC GKS Stroke -14?\?XDEC GKS Stroke -15?\?XDEC GKS Stroke -16?\
?XDEC GKS Stroke -17?\?XDEC GKS Stroke -18?\?XDEC GKS Stroke -19?\
?XDEC GKS Stroke -20?\?XDEC GKS Stroke -21?\?XDEC GKS Stroke -22?\
?XDEC GKS Stroke -23?\1.0?XB?\@?X<?\1/A1#a01$\'XXX01%' 1&'_1'' 0"
?X10:57:33.35?\2 @!tA2!@2"A2#A2$A2%ltAltA1@1@2&0#3$ltAltA1@1
@56ltA57ltAltAltAltA6,ltAltAltAltA6*ltAltA3$ltAltA1@1@61@'?GA'?GB'
?GC'?GD'?GE'?GG'?GF'?GH'?GI'?GJ'?GK'?GN'?GM'?GL'?G5 A5!A5"1@5#A ltAhQ1
@ltA50A5:A51A52@531@55A56mcz|EW57ltAmcz|EWMcz|EWltA5$A5$C5&1@5'A6 A6
"A6,ltA1@1@ltA0$0!

```

E.2.6 CGM Physical File Organization

The DEC GKS CGM metafile outputs 512 byte records. Using the clear text encoding, the DEC GKS CGM metafile separates element opcodes with a semicolon (;), a line-feed, and a carriage return character.

Language-Specific Programming Information

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

NOTE

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

F.1 Passing Arguments by Descriptor

DEC GKS requires array descriptors of class A or NCA, which include a bounds block for two-dimensional arrays. Array descriptors of class NCA must be contiguous.

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

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

- CELL ARRAY
- INQUIRE SET OF ACTIVE WORKSTATIONS
- INQ_AVAILABLE GENERALIZED DRAWING PRIMITIVES

- **INQ_COLOR_INDEXES**
- **INQ_DEF_CHOICE_DATA**
- **INQ_DEF_LOCATOR_DATA**
- **INQ_DEF_PICK_DATA**
- **INQ_DEF_STRING_DATA**
- **INQ_DEF_STROKE_DATA**
- **INQ_DEF_VALUATOR_DATA**
- **INQ_FILL_FAC**
- **INQ_FILL_INDEXES**
- **INQ_GDP**
- **INQ_OPEN_WS**
- **INQ_PAT_INDEXES**
- **INQ_PAT_REP**
- **INQ_PIXEL_ARRAY**
- **INQ_PLINE_FAC**
- **INQ_PLINE_INDEXES**
- **INQ_PMARK_FAC**
- **INQ_PMARK_INDEXES**
- **INQ_PREDEF_PAT_REP**
- **INQ_SEG_NAMES_ON_WS**
- **INQ_SET_ASSOC_WS**
- **INQ_TEXT_FAC**
- **INQ_TEXT_INDEXES**
- **INQ_WSTYPE_LIST**
- **INQ_XFORM_LIST**
- **REQUEST_STROKE**
- **SET PATTERN REPRESENTATION**

F.2 Programming in BASIC

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

F.3 Programming in VAX C

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

F.4 Programming in VAX COBOL

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

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

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

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

Example F-1: Macro Subroutine Used to Build Array Descriptors

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

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

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

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

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

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

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

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

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

Example F-2: A Sample COBOL Program Using the Subroutine BUILDESC

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                C09.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.          VAX-11.
OBJECT-COMPUTER.         VAX-11.
DATA DIVISION.

WORKING-STORAGE SECTION.
01    valthree PIC S9(9)  COMP VALUE 3.
01    valfour  PIC S9(9)
COMP VALUE 4.
01    valone   PIC S9(9)
COMP VALUE 1.
01    valzero  PIC S9(9)
COMP VALUE 0.
01    valpoint1 USAGE IS COMP-1 VALUE 0.1.
01    valpoint5 USAGE IS COMP-1 VALUE 0.5.
01    colidx.
      05 dim1 OCCURS 3 TIMES.
      /n      10 colia OCCURS 4 TIMES
PIC S9(9) COMP.
01    colidx_d.
      05 desc OCCURS 11 TIMES PIC S9(9) COMP.

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

      CALL "GKS$OPEN_GKS" USING
          BY DESCRIPTOR 'GKS.ERR'.
      CALL "GKS$OPEN_WS" USING
          BY REFERENCE valone,valzero,valzero.

      CALL "GKS$ACTIVATE_WS" USING
          BY REFERENCE valone.
```

(continued on next page)

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

```
CALL "BUILDESC" USING
    BY REFERENCE colidx_d,
    BY DESCRIPTOR colia(1,1),
    BY VALUE valthree, valfour.
CALL "GKS$CELL_ARRAY" USING
    BY REFERENCE valpoint1, valpoint1, valpoint5, valpoint5,
    BY REFERENCE valone, valone,
    BY REFERENCE valthree, valfour,
    BY REFERENCE colidx_d.

CALL "GKS$DEACTIVATE_WS" USING
    BY REFERENCE valone.
CALL "GKS$CLOSE_WS" USING
    BY REFERENCE valone.
CALL GKS$CLOSE_GKS".
EXIT PROGRAM.
END PROGRAM C09.
```

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

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

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

F.5 Programming in VAX Pascal

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

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

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

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

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

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

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

The following type definitions have changed in the GKSDEFS.PAS include file.

Table F-1: Type Definitions

| Definition | Data Type |
|------------------------|---------------------------|
| Asf_Flag_Array | Array [1...13] of Integer |
| Coord_limit_Array | Array [1...4] of Real |
| Up_Vector_Array | Array [1...2] of Real |
| Two_real | Array [1...2] of Real |
| Indices_Array | Array [1...4] of Integer |
| Twointeger | Array [1...2] of Integer |
| GKS\$Asf_Flag_Array | Array [1...13] of Integer |
| GKS\$Coord_limit_Array | Array [1...4] of Real |

(continued on next page)

Table F-1 (Cont.): Type Definitions

| Definition | Data Type |
|----------------------|--------------------------|
| GKS\$Up_Vector_Array | Array [1...2] of Real |
| GKS\$Two_real | Array [1...2] of Real |
| GKS\$Indices_Array | Array [1...4] of Integer |
| GKS\$Twointeger | Array [1...2] of Integer |

DEC GKS Device-Independent Fonts

This appendix provides additional information about the fonts which can be accessed from the DEC GKS software in stroke-precision text.

One font is used as the standard DEC GKS font for stroke precision text. Figure G-2 illustrates the DEC GKS multinational font. It is a monospaced font; all characters are the same size. DEC GKS uses this as the default font.

Other fonts, known as the Hershey fonts, are also available. These character fonts were digitized by Dr. A. V. Hershey of the Naval Surface Weapons Laboratory, and have been supplied by the National Bureau of Standards. The character information for these fonts has been organized into 22 fonts, as shown in Figures G-3 through Figure G-24. The Hershey fonts are not monospaced; each character box is a different size. The character box for each character is not necessarily the same size as the character. In most cases, the character box is larger than the character, although for some characters (for example, those with descenders) the character may go outside of its box.

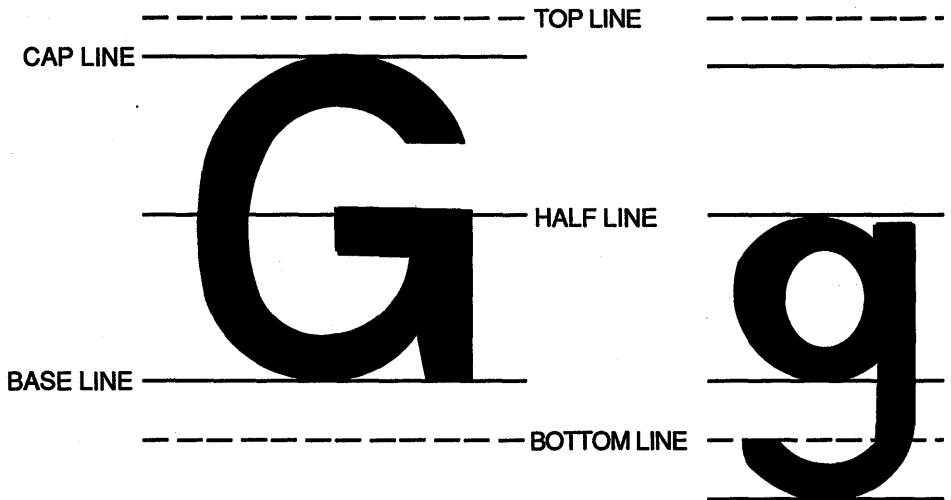
G.1 Font File Formats

The center line for all fonts lies exactly halfway between the left and right lines of each character.

Similarly, the half line lies exactly halfway between the base line and the cap line.

Figure G-1 illustrate the font lines:

Figure G-1: DEC GKS Font Lines

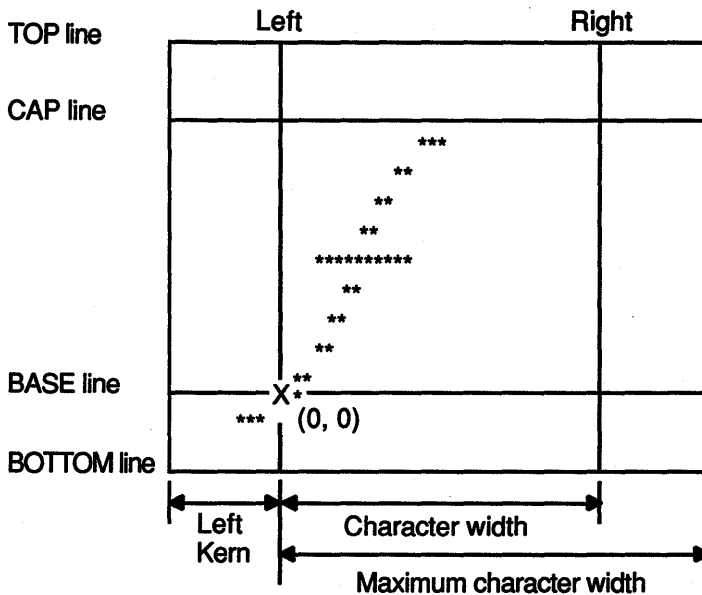


ZK-1449-GE

This restriction applies to the font file formats because the center line and the halfline are calculated by DEC GKS and are not data items in the font file. DIGITAL reserves the right to change font file formats in future releases.

G.2 Font Design

The stroke font is designed as follows.



X is the origin of this coordinate.

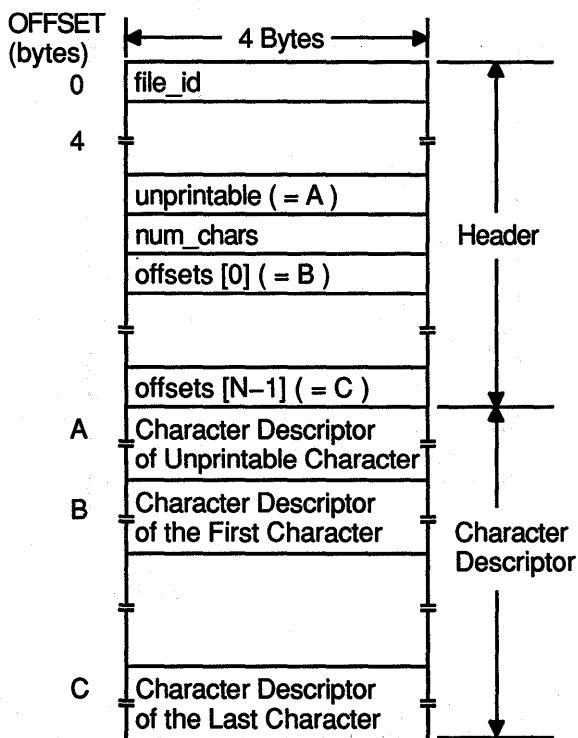
ZK-1582A-GE

The character shapes are drawn using a coordinate system in which the base line is $Y = 0$, and the left line is $X = 0$.

Each glyph in a font file is stored as one or more polylines. The origin for these polylines is the left base point of the character. The points of the polylines are specified in integer numbers.

In DEC GKS, the X and Y coordinates of a glyph are normalized by the distance between the CAP and BASE lines. The value of the distance between CAP and BASE defines the precision of the glyph design. That is, if this value is large, you can make a more detailed design for a polyline.

G.3 Stroke Font File



ZK-1583A-GE

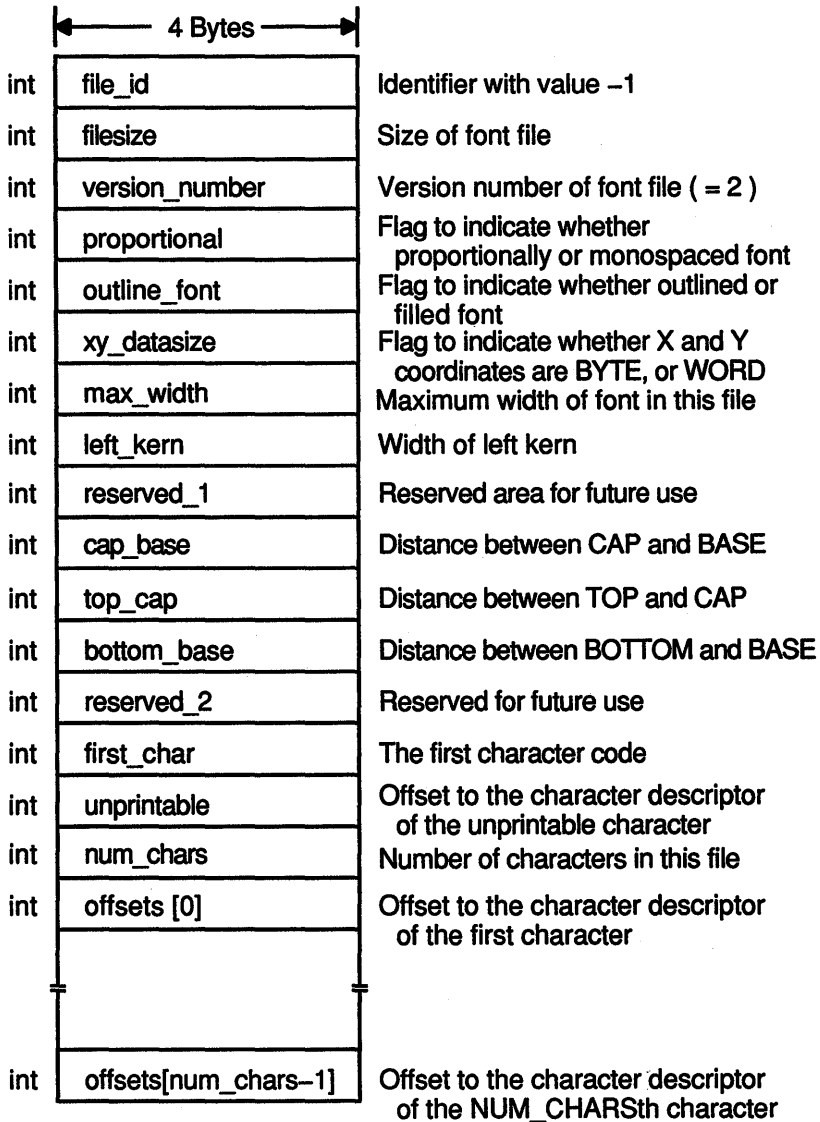
The stroke font file has the structure shown in the preceding figure. This structure is divided into two parts; the header and a set of character descriptors.

The header structure has common information for all glyphs in a stroke font file and control data. Each character descriptor contains the data for each glyph. To point to each character descriptor, there are offset arrays in the header structure. The index of an offset array is calculated by subtracting the first character from the output character code. Each offset array has the number of bytes from the top of the stroke font file.

A character descriptor has all the information to make a glyph. This includes the number of polylines in a glyph, the character width, and the data set of each polyline, which contains the number of points in the polyline and the set of X and Y point coordinates.

G.3.1 Stroke Font File Header

The header of the stroke font file has the following structure.



ZK-1584A-GE

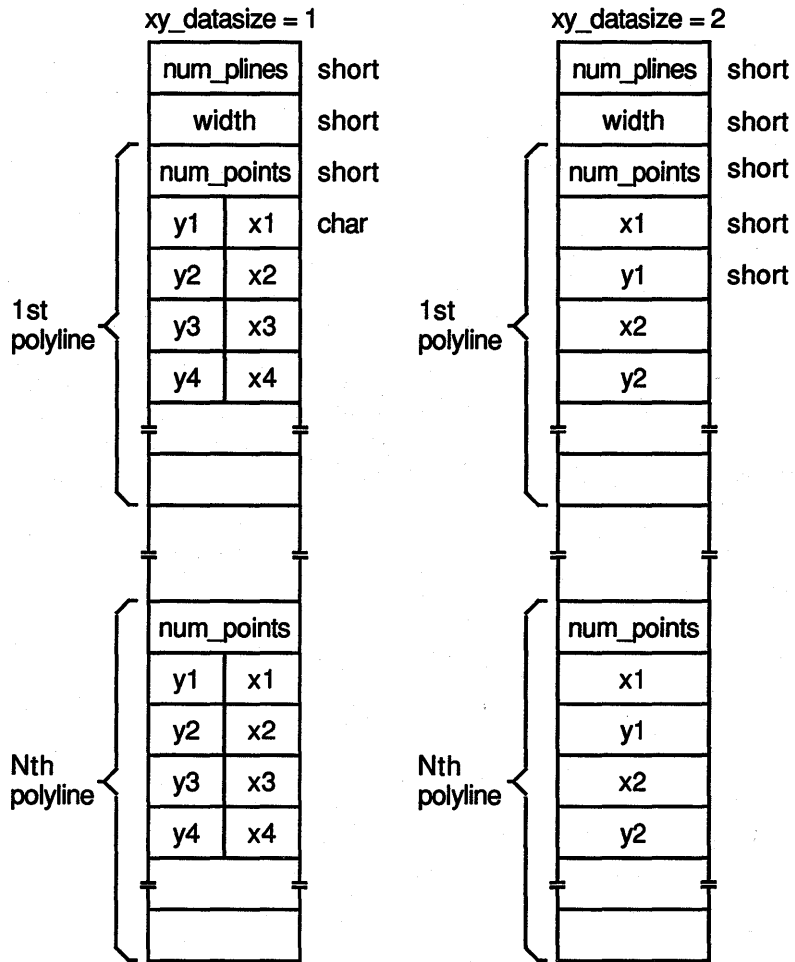
The header contains the following elements:

- **file_id**
This element helps identify the file as a DEC GKS font file. In DEC GKS, this identifier is a 32-bit integer with the value -1.
- **filesize**
This element specifies the size of the font file, in bytes. This is used to map the font file into virtual memory. It is stored as an integer value.
- **version_number**
This element specifies the version number of the stroke font file. This identifier is a 32-bit integer with the value 2.
- **proportional**
This element indicates whether or not the font is proportionally spaced. This integer is 1 if the font is proportionally spaced, and 0 if it is monospaced. **Proportionally spaced** means that different characters have different widths. **Monospaced** means that every character has the same width.
- **outline_font**
This element is an integer flag that indicates if the font is stored as outline. If the value of this element is not equal to zero, then the font is assumed to be stored as outline. This means that the polylines specified are filled instead of being drawn.
- **xy_datasize**
This element indicates whether X and Y coordinates are stored as a byte signed integer (byte) or as a 2-byte signed integer (word). If this element is 1, then the X and Y coordinates are stored as a byte. If this element is 2, the coordinates are stored as a word.
- **max_width**
This element specifies the width of the font characters. This is an integer. If the font is proportionally spaced, this number specifies the maximum width of the font.
- **left_kern**
This element specifies the maximum length of the negative direction for the X coordinate of the glyph in the stroke font file. This is an integer.
- **reserved_1**
Reserved for future use.

- **cap_base**
This element specifies the distance of the cap line above the base line. This is an integer.
- **top_cap**
This element specifies the distance of the top line above the cap line. This is an integer.
- **bottom_base**
This element specifies the distance of the bottom line below the base line. This is an integer.
- **reserved_2**
Reserved for future use.
- **first_char**
This element specifies the numeric value of the first character in the font. ("First" here refers to an ascending order. For example, for the ASCII character set, the first non-control character is a space which has the numeric value of 32. The corresponding entry in the font file would contain the number 32.) This element is an integer.
- **unprintable**
This element specifies the offset to the character descriptor of the unprintable character. This is an integer.
- **num_chars**
This element specifies the number of characters supported in the font file. There should be no gaps in the number of characters supported. If there are characters which should not be displayed, they should be indicated by the *number of polylines = 0*, which generates a space. This is an integer.
- **offsets**
This element is an array of *num_chars* entry points. This array contains the offsets to the character descriptor of each glyph to be displayed.

G.3.2 Character Descriptor

The character descriptor has the following structure:



ZK-1585A-GE

The character descriptor contains the following elements:

- **num_plines**
This element specifies the number of polylines making up a glyph. This is a word.
- **width**
This element specifies the width of a glyph. This is a word. This element is ignored for monospaced fonts.
- **num_points**
This element specifies the number of points in a polyline. This is a word.
- **x and y**
These elements specify the X and Y coordinate values of a point. If *xy_datasize* in the header is 1, these values are 1-byte signed integers. If *xy_datasize* is 2, these values are 2-byte signed integers.

This section presents the DEC GKS device-independent fonts. These figures represent the ASCII characters 33 through 126, beginning in the upper left corner and incrementing horizontally to the lower right corner. Not all characters are present in all the fonts. Fonts 1 and -1 specify the same font.

Example G-1 presents a program that you can execute if you want to see the ASCII value next to the corresponding font character on the workstation surface.

Example G-1: Printing the ASCII Values of Font Characters

```
IMPLICIT NONE
INCLUDE 'SYS$LIBRARY:GKSDEFS.FOR'
INTEGER      WS_ID, ASCVAL, FONT, COL, ROW, MAXROW, MAXCOL,
*            DUMMY_INTEGER, WS_TYPE, ERROR_STATUS
REAL         MAXX, MAXY, RATIO, HEIGHT, X1, Y1
CHARACTER*1  TXT
CHARACTER*4  FONTTYPE
CHARACTER*3  ASCSTR
CHARACTER*40 HEADER, DUMMY_STRING
DATA        WS_ID /1/, HEIGHT /0.66/, MAXROW /19/,
*           MAXCOL /5/

C           Set up the DEC GKS and the workstation environment.
CALL GKS$OPEN_GKS( 'SYS$ERROR:' )
CALL GKS$OPEN_WS( WS_ID, GKS$K_CONID_DEFAULT, GKS$K_VT240 )
CALL GKS$ACTIVATE_WS( WS_ID )
```

(continued on next page)

Example G-1 (Cont.): Printing the ASCII Values of Font Characters

```
C    Inquire about workstation and set up transformations.
    CALL GKS$INQ_WS_TYPE( WS_ID, ERROR_STATUS, DUMMY_STRING,
*   WS_TYPE, DUMMY_INTEGER )

    CALL GKS$INQ_MAX_DS_SIZE( WS_TYPE, ERROR_STATUS, DUMMY_INTEGER,
*   MAXX, MAXY, DUMMY_INTEGER, DUMMY_INTEGER )

    RATIO = MAXY / MAXX
    CALL GKS$SET_WINDOW( 1, 0.0, 15.0, 0.0, 23.0 )
    CALL GKS$SET_VIEWPORT( 1, 0.0, 1.0, 0.0, RATIO )
    CALL GKS$SELECT_XFORM( 1 )
    CALL GKS$SET_WS_WINDOW( 1, 0.0, 1.0, 0.0, RATIO )
    CALL GKS$SET_WS_VIEWPORT( 1, 0.0, MAXX, 0.0, MAXY )

C    Get the font number.
    WRITE( 5, * ) 'Enter Font Number (-1 to -23): '
    READ( 5, * ) FONT
    CALL GKS$CLEAR_WS( WS_ID, GKS$K_CLEAR_ALWAYS )

C    Draw headings.
    CALL GKS$SET_TEXT_HEIGHT( 0.75 * HEIGHT )
    CALL GKS$SET_TEXT_SPACING( -0.1 )
    CALL GKS$SET_TEXT_ALIGN( GKS$K_TEXT_HALIGN_CENTER,
*   GKS$K_TEXT_VALIGN_HALF )
    CALL GKS$SET_TEXT_FONTPREC( 1, GKS$K_TEXT_PRECISION_STROKE )
    WRITE( FONTTYPE, 10 ) FONT
10   FORMAT( I4 )
    HEADER = 'ASCII VALUES FOR CHARACTERS OF FONT ' // FONTTYPE
    CALL GKS$TEXT ( 7.5, 22.0, HEADER )

C    Draw ascii numbers.
    CALL GKS$SET_TEXT_FONTPREC( 1, GKS$K_TEXT_PRECISION_STROKE )
    CALL GKS$SET_TEXT_HEIGHT( HEIGHT )
    CALL GKS$SET_TEXT_SPACING( -0.4 )
    DO COL = 1, 5
        X1 = COL * 3.0 - 2.5
        DO ROW = 1, MAXROW
            Y1 = 20.0 - ROW
            ASCVAL = (COL - 1) * MAXROW + ROW + 31
            WRITE( ASCSTR, 20 ) ASCVAL
20   FORMAT( I3 )
            CALL GKS$TEXT ( X1, Y1, ASCSTR )
        END DO
    END DO
```

(continued on next page)

Example G-1 (Cont.): Printing the ASCII Values of Font Characters

```
C      Draw font characters.
      CALL GKS$SET_TEXT_FONTPREC( FONT, GKS$K_TEXT_PRECISION_STROKE )
      DO COL = 1, 5
        X1 = COL * 3.0 - 1.25
        DO ROW = 1, MAXROW
          Y1 = 20.0 - ROW
          ASCVAL = (COL - 1) * MAXROW + ROW + 31
          TXT = CHAR( ASCVAL )
          CALL GKS$TEXT ( X1, Y1, TXT )
        END DO
      END DO

      CALL GKS$DEACTIVATE_WS( WS_ID )
      CALL GKS$CLOSE_WS( WS_ID )
      CALL GKS$CLOSE_GKS()
      END
```

DEC GKS Device-Independent Fonts

Each figure includes a name that describes the font. **Simplex** means that the characters are made up of one line. **Duplex** means that the characters are made up of two lines. **Complex** means that the characters are made up of more than two lines.

**Figure G-2: DEC GKS Default Font Number 1
ISO Standard Character Set**

| | | | | | | | | | | | |
|----|---|---|---|---|---|----|---|---|---|---|---|
| ! | 0 | @ | P | ` | p | ; | ° | À | Ë | à | ÿ |
| " | 1 | A | Q | a | q | ! | ± | Á | Ä | á | ÿ |
| # | 2 | B | R | b | r | " | ² | Â | Å | â | ÿ |
| \$ | 3 | C | S | c | s | # | ³ | Ã | Ä | ã | ÿ |
| % | 4 | D | T | d | t | \$ | µ | Ä | Å | ä | ÿ |
| & | 5 | E | U | e | u | % | ¶ | Å | Æ | å | ÿ |
| ' | 6 | F | V | f | v | & | · | Æ | Ç | æ | ÿ |
| (| 7 | G | W | g | w | ' | ¸ | Ç | È | ç | ÿ |
|) | 8 | H | X | h | x | (| ¹ | È | É | è | ÿ |
| * | 9 | I | Y | i | y |) | º | É | Ê | é | ÿ |
| + | : | J | Z | j | z | * | » | Ê | Ë | ê | ÿ |
| , | ; | K | [| k | { | + | ¼ | Ë | Ü | ë | ÿ |
| - | < | L | \ | l | | , | ½ | Ü | Ý | ü | ÿ |
| . | = | M |] | m | } | - | ¾ | Ý | ÿ | ÿ | ÿ |
| / | > | N | ^ | n | ~ | . | ¿ | ÿ | ÿ | ÿ | ÿ |
| | ? | O | _ | o | ~ | / | | | | | |

ZK-1574-84

DEC GKS Device-Independent Fonts

**Figure G-3: DEC GKS Font Number -2
Small Uniplex Simplex Roman and Greek**

!"#\$%&'()*+,-./0123
456789:;=>?&ABCDEFGH
IJKLMNOPQRSTUVWXYZ(
/)|-'ABΓΔΕΖΗΘΙΚΛΜΝΞΟ
ΠΡΣΤΥΦΧΨΩ'×| ▣

ZK-1575-84

DEC GKS Device-Independent Fonts

**Figure G-4: DEC GKS Font Number -3
Large Simplex Uniplex Roman**

!"#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFGH
IJKLMNOPQRSTUVWXYZ▣
↓→↑←‘abcdefghijklmno
pqrstuvwxyzx|·→

ZK-1576-84

DEC GKS Device-Independent Fonts

Figure G-5: DEC GKS Font Number -4
Large Uniplex Simplex Greek

!"#\$%&'()*+,-./0123
456789:;<=>?@ABΓΔΕΖΗ
ΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩεθκ
↓→↑←'αβγδεζηθικλμνξο
πρστυφχψω∇φ×|·→

ZK-1577-84

DEC GKS Device-Independent Fonts

**Figure G-6: DEC GKS Font Number -5
Large Simplex Uniplex Script**

!"#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFGHIJ
KLMNOPQRSTUVWXYZ□
↓→↑←'abcdefghijklmno
pqrstuvwxyzx|·→

ZK-1578-84

DEC GKS Device-Independent Fonts

Figure G-7: DEC GKS Font Number -6
Medium Complex Duplex Roman

!"#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFG
HIJKLMN OPQRSTUVWXYZ[
'] ^ ` ' a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~

ZK-1579-84

DEC GKS Device-Independent Fonts

Figure G-8: DEC GKS Font Number -7
Medium Complex Duplex Greek

!"#\$%&'()*+,-./0123
456789:;<=>?@ABΓΔΕΖΗ
ΘΙΚΛΜΝΞΟΠΡΣΤΤΦΧΨΩεθ[
'] ^ ← ' α β γ δ ε ζ η θ ι κ λ μ ν ξ ο
π ρ σ τ υ φ χ ψ ω φ ξ | } ~

ZK-1580-84

DEC GKS Device-Independent Fonts

Figure G-9: DEC GKS Font Number -8
Medium Complex Duplex Italic

!'#\$%&'()+,-./0123*

456789:;<=>?@ABCDEFG

HIJKLMNPOQRSTUVWXYZ[

]^_`'abcdefghijklmno

pqrstuvwxyz{|}~

ZK-1581-84

DEC GKS Device-Independent Fonts

**Figure G-10: DEC GKS Font Number -9
Large Complex Duplex Roman**

!'#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFG
HIJKLMN OPQRSTUVWXYZ[
]~^←'abcdefghijklmno
pqrstuvwxyz{|}~

ZK-1582-84

DEC GKS Device-Independent Fonts

Figure G-11: DEC GKS Font Number -10
Large Complex Duplex Greek

!"#\$%&'()*+,-./0123
456789:;<=>?@ABΓΔΕΖΗ
ΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩεθ[
]~←'αβγδεζηθικλμνξο
πρστυφχψωφ{|}~

ZK-1583-84

DEC GKS Device-Independent Fonts

Figure G-12: DEC GKS Font Number -11
Large Complex Duplex Italic

!'#\$%&'()+,-./0123*
456789:;<=>?@ABCDEFG
HIJKLMN OPQRSTUVWXYZ'
°^←'abcdefghijklmno
pqrstuvwxyz}}~

ZK-1584-84

DEC GKS Device-Independent Fonts

Figure G-13: DEC GKS Font Number -12
Large Simplex Duplex Roman

!'#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFGH
HIJKLMNPOQRSTUVWXYZ'
°^←'abcdefghijklmno
pqrstuvwxyz(|)~

ZK-1585-84

DEC GKS Device-Independent Fonts

Figure G-14: DEC GKS Font Number -13
Large Complex Duplex Script

!"#\$%&'()*+,-./0123
456789:;<=>?@ABCDEFGHI
JKLMNOPQRSTUVWXYZ
°^←'abcdefghijklmno
pqrstuvwxyz}}~

ZK-1586-84

DEC GKS Device-Independent Fonts

**Figure G-15: DEC GKS Font Number -14
Large Complex Duplex Cyrillic**

!"#\$%&'()*+,-./0123
456789:;ю=я?@АБВГДЕЖ
ЗИЙКЛМНОПРСТУФХЦШЩЪ
ЫЬЭЮЯабвгдежзийклмно
прстуфхцшщъыьэ

ZK-1587-84

DEC GKS Device-Independent Fonts

Figure G-16: DEC GKS Font Number -15
Large Complex Triplex Roman

!'#\$%&'()*+,-./0123
456789:;='°?@ABCDEFGH
HIJKLMN OPQRSTUVWXYZ[
] ^ ← 'abcdefghijklmnop
pqrstuvwxyz{|}~

ZK-1588-84

DEC GKS Device-Independent Fonts

Figure G-17: DEC GKS Font Number -16
Large Complex Triplex Italic

!'#\$%&'()+,-./0123*
456789:;<=>?@ABCDEFG
HIJKLMN OPQRSTUVWXYZ'
°^←'abcdefghijklmno
pqrstuvwxyz|•~

ZK-1589-84

DEC GKS Device-Independent Fonts

Figure G-18: DEC GKS Font Number -17
Large Gothic Triplex German

!'#\$%&'()*+,-./0123
456789:;='°?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ
[] ^ _ ` ' a b c d e f g h i j k l m n o
p q r s t u v w x y z { }

ZK-1590-84

DEC GKS Device-Independent Fonts

Figure G-19: DEC GKS Font Number -18
Large Gothic Triplex English

!"#\$%&'()*+,-./0123
456789:;='°?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ[
]~←'abcdefghijklmno
pqrstuvwxyz{|}~

ZK-1591-84

DEC GKS Device-Independent Fonts

Figure G-20: DEC GKS Font Number -19
Large Gothic Triplex Italian

!"#\$%&'()*+,-./0123
456789:;='°?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ
[] ^ _ ` ' a b c d e f g h i j k l m n o
p q r s t u v w x y z { } ~

ZK-1592-84

Appendix H

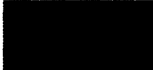






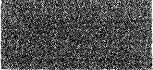










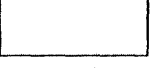

DEC GKS Color Chart

This appendix presents a chart of 64 colors and their corresponding red, green, and blue intensity values. If you are working with a color VT125, a VT241, or a VAXstation II/GPX, you can use this color chart as a guide when calling the function `SET COLOR REPRESENTATION`. The colors presented are the 64 colors supported by the VT125 and the VT241. For information concerning the availability and use of colors on these workstations, refer to the appropriate device-specific appendix in this manual.

You should use this color chart as a guide. You should not expect your monitor to display the colors exactly as shown. Colors can vary from monitor to monitor depending on the following factors:


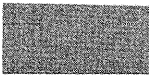















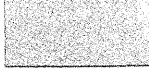

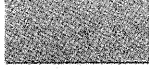
- The current background color (affects lighter shades)
- The current brightness and contrast control settings
- The available room light
- The proximity of the primitive to other colors on the display

Table H-1: DEC GKS Color Chart

| Red | Green | Blue | | Red | Green | Blue | |
|--------|--------|--------|---|--------|--------|--------|---|
| 0.0000 | 0.0000 | 0.0000 |  | 0.6133 | 0.4200 | 1.0000 |  |
| 0.0000 | 0.0000 | 0.5600 |  | 0.5700 | 0.1400 | 1.0000 |  |
| 0.3300 | 0.3300 | 0.3300 |  | 0.5600 | 0.0000 | 0.8400 |  |
| 0.2142 | 0.2142 | 0.6258 |  | 0.6646 | 0.2862 | 0.8538 |  |
| 0.0000 | 0.0000 | 0.8400 |  | 0.5600 | 0.0000 | 0.5600 |  |
| 0.2862 | 0.2862 | 0.8538 |  | 0.7119 | 0.4281 | 0.7119 |  |
| 0.1400 | 0.1400 | 1.0000 |  | 1.0000 | 0.1400 | 1.0000 |  |
| 0.6700 | 0.6700 | 0.6700 |  | 1.0000 | 0.4200 | 1.0000 |  |
| 0.5679 | 0.5679 | 0.8521 |  | 0.9235 | 0.7765 | 0.9235 |  |
| 1.0000 | 1.0000 | 1.0000 |  | 1.0000 | 0.7000 | 1.0000 |  |



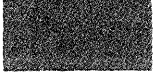
















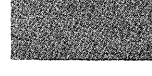
(continued on next page)

Table H-1 (Cont.): DEC GKS Color Chart

| Red | Green | Blue | | Red | Green | Blue | |
|--------|--------|--------|---|--------|--------|--------|---|
| 0.8400 | 0.0000 | 0.5600 |  | 1.0000 | 0.6133 | 0.4200 |  |
| 0.8538 | 0.2862 | 0.6646 |  | 1.0000 | 0.5700 | 0.1400 |  |
| 1.0000 | 0.1400 | 0.5700 |  | 0.8400 | 0.5600 | 0.0000 |  |
| 1.0000 | 0.4200 | 0.6133 |  | 0.8538 | 0.6646 | 0.2862 |  |
| 0.5600 | 0.0000 | 0.0000 |  | 0.5600 | 0.5600 | 0.0000 |  |
| 0.6258 | 0.2142 | 0.2142 |  | 0.7119 | 0.7119 | 0.4281 |  |
| 0.8400 | 0.0000 | 0.0000 |  | 1.0000 | 1.0000 | 0.1400 |  |
| 0.8538 | 0.2862 | 0.2862 |  | 1.0000 | 1.0000 | 0.4200 |  |
| 1.0000 | 0.1400 | 0.1400 |  | 0.9235 | 0.9235 | 0.7765 |  |
| 0.8521 | 0.5679 | 0.5679 |  | 1.0000 | 1.0000 | 0.7000 |  |




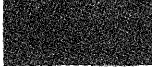
(continued on next page)

Table H-1 (Cont.): DEC GKS Color Chart

| Red | Green | Blue | | Red | Green | Blue | |
|--------|--------|--------|---|--------|--------|--------|---|
| 0.5600 | 0.8400 | 0.0000 |  | 0.4200 | 1.0000 | 0.6133 |  |
| 0.6646 | 0.8538 | 0.2862 |  | 0.1400 | 1.0000 | 0.5700 |  |
| 0.5700 | 1.0000 | 0.1400 |  | 0.0000 | 0.8400 | 0.5600 |  |
| 0.6133 | 1.0000 | 0.4200 |  | 0.2862 | 0.8538 | 0.6646 |  |
| 0.0000 | 0.5600 | 0.0000 |  | 0.0000 | 0.5600 | 0.5600 |  |
| 0.2142 | 0.6258 | 0.2142 |  | 0.4281 | 0.7119 | 0.7119 |  |
| 0.0000 | 0.8400 | 0.0000 |  | 0.1400 | 1.0000 | 1.0000 |  |
| 0.2862 | 0.8538 | 0.2862 |  | 0.4200 | 1.0000 | 1.0000 |  |
| 0.1400 | 1.0000 | 0.1400 |  | 0.7765 | 0.9235 | 0.9235 |  |
| 0.5679 | 0.8521 | 0.5679 |  | 0.7000 | 1.0000 | 1.0000 |  |

(continued on next page)

Table H-1 (Cont.): DEC GKS Color Chart

| Red | Green | Blue | | Red | Green | Blue | |
|------------|--------------|-------------|---|------------|--------------|-------------|---|
| 0.0000 | 0.5600 | 0.8400 |  | 0.1400 | 0.5700 | 1.0000 |  |
| 0.2862 | 0.6646 | 0.8538 |  | 0.4200 | 0.6133 | 1.0000 |  |

DEC GKS GDPs and Escapes

This appendix describes all the DEC GKS-supported generalized drawing primitives (GDPs) and escapes. Most of the GDPs and escapes are supported by all the DEC GKS workstations. If all DEC GKS-supported workstations do not support a particular GDP or escape, this appendix flags the corresponding description.

All GDPs and escapes have negative values as identification numbers. (You pass the identification numbers to either **GENERALIZED DRAWING PRIMITIVE** or **ESCAPE**.) DEC GKS defines GDP and escape constants in the definition file for your particular programming language. For more information concerning the definition files, refer to Chapter 1, Introduction to DEC GKS.

For further information concerning the use of GDPs, refer to **GENERALIZED DRAWING PRIMITIVE** in Chapter 4, Output Functions. For further information concerning the use of escapes, refer to **ESCAPE** in Chapter 3, Control Functions. The function descriptions for **GENERALIZED DRAWING PRIMITIVE** and **ESCAPE** list the error messages that may be generated by using any GDP or escape.

Some of the GDPs and escapes require additional information contained in a data record. All required data records must be passed to **GENERALIZED DRAWING PRIMITIVE** and **ESCAPE** in the DEC GKS GDP/escape standard data record format. For all GDPs and escapes, you must pass the exact data record size as specified in the descriptions in this appendix. If you do not, the call to either **GENERALIZED DRAWING PRIMITIVE** or **ESCAPE** generates an error message. For a complete description of the standard GDP/escape data record format, refer to Chapter 1, Introduction to DEC GKS.

Data Record Format Used in This Appendix

Data Record Format Used in This Appendix

Since this appendix uses a short notation to describe the required contents of a GDP/escape data record, you may wish to read the description of the GDP/escape data record format in Chapter 1, Introduction to DEC GKS, before reading further.

In this appendix, the descriptions of the first three components of the data record are the values actually contained in the data record. The descriptions of the last four components do not describe the contents of the last four components; they describe the contents of the arrays whose addresses occupy the last four components of the data record.

Consider the following list of arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Three points on the circumference. |
| gdp_id | -10 |
| data_record | (4 components) 2 0 0 (address of) int_value_1, int_value_2 |
| data_record_size | 16 bytes |

The data record portion of this GENERALIZED DRAWING PRIMITIVE description (*data_record*) specifies that the data record has four components. The first component is an integer value (2), specifying the number of valid elements in the integer array.

The next two components of the data record contain zeros (0), specifying the number of valid elements in the real and string arrays whose addresses occupy the last three components of the data record. Since the arrays contain no valid elements, you do not have to include room for these array addresses in your data record.

Data Record Format Used in This Appendix

The fourth component specifies the address of an array; the array itself contains identifiers *int_value_1* and *int_value_2*. The GDP description in this appendix describes the purpose of these integers. GENERALIZED DRAWING PRIMITIVE uses the address provided in the fourth component to locate the integer array.

NOTE

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 7, Input Functions.

Generalized Drawing Primitives (GDPs)

Generalized Drawing Primitives (GDPs)

The following sections describe the DEC GKS-supported Generalized Drawing Primitives (GDPs). The sections identify each GDP by the following:

- The numeric identifier that you pass to `GENERALIZED DRAWING PRIMITIVE`.
- The title of the primitive (for instance, "Circle").
- The constant equivalent of the numeric identifier.
- The list of supporting workstations.
- The description of the primitive.
- The list of the arguments passed to `GENERALIZED DRAWING PRIMITIVE` and the contents of the data record, if applicable. The names of the arguments are identical to the argument descriptions of `GENERALIZED DRAWING PRIMITIVE` in Chapter 4, Output Functions.
- The list of GDP-specific error messages, if applicable.

If you specify points to `GENERALIZED DRAWING PRIMITIVE` that cannot be used to uniquely define a primitive, you generate error number `DECGKS$_ERROR_NEG_158`. For more information concerning error `DECGKS$_ERROR_NEG_158`, refer to the individual escape or GDP description in this appendix.

Most of the DEC GKS GDPs are capable of generating error number `GKS$_ERROR_100` (Number of points is invalid in routine ****). If it is not clear how a GDP can generate this error message, the description of the individual GDP provides additional information.

The following information applies to all DEC GKS GDPs:

- DEC GKS applies normalization transformations to the world coordinates of a specified GDP, but draws the GDP on the NDC plane. This will sometimes cause unexpected results. For instance, if you include a rectangular GDP in a segment and then rotate the segment, DEC GKS alters the coordinate points but still draws the sides of the rectangle parallel to the X and Y axes. Also, when specifying coordinate values for

Generalized Drawing Primitives (GDPs)

circles, the current normalization transformation affects only the size of the circle, and does not alter the shape.

- All radius specifications constitute vector values. The only significance of the radius vector is its length in world coordinates.
- You specify angles in radians. (To calculate radians, use the formula $360 \text{ degrees} = 2 * \pi \text{ radians}$.) Positive rotation is counterclockwise; negative rotation is clockwise.
- Some GDPs require vector values in the X and Y coordinate arrays passed to GENERALIZED DRAWING PRIMITIVE. When you specify a vector value, you pass two sets of world coordinate points. DEC GKS calculates the distance, the angle, or both values, using the two specified points.

Using a GDP, you calculate all vectors from a single *vector origin point*. The vector origin point is the first point in a vector specification; you specify the second point of the vector specification in the X and Y coordinate array that you pass to GENERALIZED DRAWING PRIMITIVE.

For instance, the GDP GKS\$K_GDP_ARC_CTR_2VEC_RAD requires, in the X and Y coordinate array, the following values:

- The center point of the circular arc
- The vector origin point
- The second point in a vector whose angle determines an endpoint of the arc
- The second point in another vector whose angle determines another endpoint of the arc
- The second point in a third vector that specifies the distance used for the circular arc's radius

DEC GKS calculates the vector values from the vector origin point to specified second points, and then applies those values to the center point of the circular arc.

Two useful vector origin points would be the center point of the arc or the origin of the world coordinate plane (0.0, 0.0). Using the center point of the arc would allow you to specify vector values in direct relation to the coordinates used to form the arc; using the origin of the world coordinate plane can make it easier for you to calculate vector values without tying them to the actual coordinate values of the arc (for instance, the center of the arc may move due to altered normalization transformations,

Generalized Drawing Primitives (GDPs)

forcing you to keep altering your vector origin point according to the new position of the arc's center). Figure I-1 illustrates the use of two different vector origin points.

The following information applies to specific types of GDPs:

- **Arcs**—When forming arcs, the DEC GDPs begin at the first specified arc point and move towards the second point in a counterclockwise direction.
- **Ellipses**—You can form ellipses in two ways. First, you can provide GENERALIZED DRAWING PRIMITIVE with the center point, and two axis vectors. DEC GKS calculates which vector specifies the greater distance, and uses both the distance and angle values to form the major axis. Then, DEC GKS calculates the distance specified by the second vector and uses the distance for the minor axis.

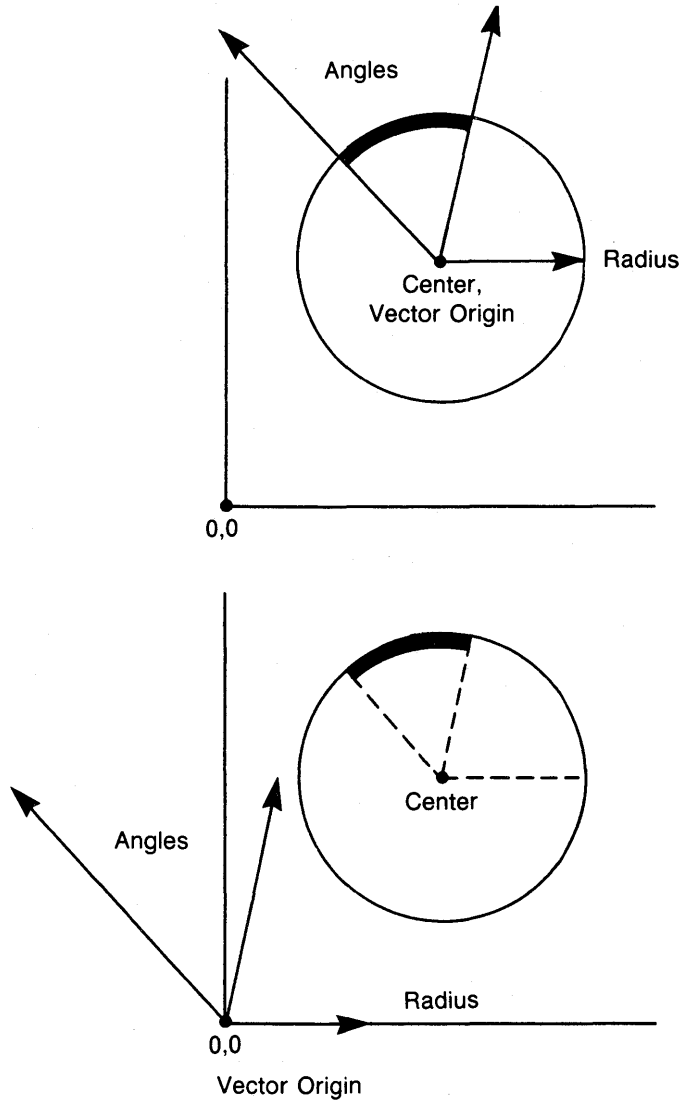
To form ellipses a second way, you can provide GENERALIZED DRAWING PRIMITIVE with the two focal points, and one point on the circumference of the ellipse. If you provide the focal points to GENERALIZED DRAWING PRIMITIVE, DEC GKS uses the following formula to form the ellipse:

$$| \text{focal_1 point} | + | \text{focal_2 point} | = 2a$$

The letter a equals the distance from the center point to the circumference along the major axis. Figure I-2 illustrates the formation of an ellipse.

Generalized Drawing Primitives (GDPs)

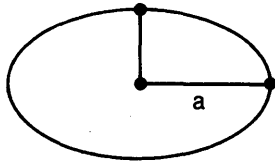
Figure I-1: Using Vector Origin Points



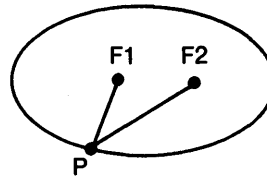
ZK-5929-HC

Generalized Drawing Primitives (GDPs)

Figure I-2: Forming an Ellipse



Center point, and
major and minor axes.



Sum of distances from focal points
to any point equals $2a$.

ZK-5788-HC

The following sections describe the DEC GKS-specific GDPs, by category.

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

Unfilled GDPs

This section describes all unfilled GDPs. Unfilled GDPs use the current polyline attributes. You should make sure that the attributes are set to the requirements of your application before you generate these GDPs.

-100 Disjoint Polyline

Constant: GKS\$K_GDP_DISJOINT_PLINE

Supporting workstations: All DEC GKS-supported workstations.

This GDP creates a series of line segments connecting the first and second specified points, the third and fourth specified points, and so forth.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|---|
| number_of_points | n points (Two for each requested line segment.) |
| x_coordinates | n x and y coordinate values. |
| y_coordinates | |
| gdp_id | -100 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 100 | DECGKS\$_ERROR_100 | Number of points is invalid in routine ****. (Either n is not an even number or n < 2.) |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

-101 Circle: Center and Point on Circumference

Constant: GKS\$K_GDP_CIRCLE_CTR_PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified center point and a single point on the circle's circumference.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---------------------------------|
| number_of_points | 2 |
| x_coordinates y_coordinates | Center and circumference point. |
| gdp_id | -101 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|----------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the center point and the point on the circumference are the same point, DEC GKS cannot form a circle.) |

-102 Circle: 3 Points on Circumference

Constant: GKS\$K_GDP_CIRCLE_3PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP draws the circle whose circumference includes the three specified points.

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|-----------------------------|
| number_of_points | 3 |
| x_coordinates y_coordinates | Three circumference points. |
| gdp_id | -102 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|----------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data. (For instance, if the three points form a straight line, DEC GKS cannot generate a corresponding circle.) |

-103 Circle: Center and Radius

Constant: GKS\$K_GDP_CIRCLE_CTR_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified center point and radius vector value.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 3 |
| x_coordinates y_coordinates | Center point, vector origin point, and radius vector point. |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

| Argument | Required Value |
|-------------------------------|----------------|
| <code>gdp_id</code> | -103 |
| <code>data_record</code> | null |
| <code>data_record_size</code> | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data. (For instance, if the radius vector specifies a distance of 0, then DEC GKS cannot generate a corresponding circle.) |

-104 Circle: 2 Points on Circumference, and Radius

Constant: `GKS$K_GDP_CIRCLE_2PT_RAD`

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified circumference points and the radius vector point. The circle is drawn so that the circumference, clockwise from the first point to the second, is no greater than pi radians (half of the circle).

GKS\$GDP Arguments:

| Argument | Required Value |
|--|---|
| <code>number_of_points</code> | 4 |
| <code>x_coordinates</code> <code>y_coordinates</code> | Two points, vector origin point, and radius vector point. |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

| Argument | Required Value |
|------------------|----------------|
| gdp_id | -104 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|-------------------------|---|
| -158 | DECGKS\$ _ERROR_NEG_158 | GDP primitive is not defined by the supplied data **** (For instance, if the distance between points is more than twice the specified radius, then DEC GKS cannot form the circle.) |

-106 Arc: Center and 2 Points on Arc

Constant: GKS\$K_GDP_ARC_CTR_2PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circular arc using the center point, the second point as a starting point of the arc, and the third point as one of the following components:

- The second point, located on the arc
- The second point of a ray (the first point is the center point), whose intersection with the circular path of the arc determines the second point of the arc

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 3 |
| x_coordinates y_coordinates | Center point and the beginning and end points of the arc. |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

| Argument | Required Value |
|-------------------------------|---|
| <code>gdp_id</code> | -106 |
| <code>data_record</code> | (4 components) 1 0 0 (address of) <code>arc_type</code> |
| <code>data_record_size</code> | 16 bytes |

The integer array contains the single element `arc_type`, which can be any of the following values:

| Value | Constant | Description |
|-------|------------------------------------|--|
| 1 | <code>GKS\$K_ARC_TYPE_OPEN</code> | Form an arcing line. |
| 2 | <code>GKS\$K_ARC_TYPE_PIE</code> | Connect both ends of the arc to its center. |
| 3 | <code>GKS\$K_ARC_TYPE_CHORD</code> | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------------------|--|
| -158 | <code>DECGKS\$ _ERROR_NEG_158</code> | GDP primitive is not defined by the supplied data in routine **** (For instance, the center point and one of the points on the circumference may be the same point.) |
| -159 | <code>DECGKS\$ _ERROR_NEG_159</code> | <code>Arc_type</code> is invalid in routine **** (For instance, if you specify a value other than 1, 2, or 3.) |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

-107 Arc: 3 Points on Circumference

Constant: GKS\$K_GDP_ARC_3PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the circular arc using a line beginning at the first point, running through the second point, and connecting to the third point.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Three points on the circumference. |
| gdp_id | -107 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arcing line. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, the three points may form a straight line.) |
| -159 | DECGKS\$_ERROR_NEG_159 | Arc_type is invalid in routine **** (For instance, if you specify a value other than 1, 2, or 3.) |

-108 Arc: Center, 2 Vectors, and a Radius

Constant: GKS\$K_GDP_ARC_CTR_2VEC_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circular arc by using the two vectors to calculate directions from the center point. DEC GKS uses the vector direction to form rays whose angles, along with the radius value, determine the starting and ending points of the arc.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 5 |
| x_coordinates y_coordinates | Center, vector origin point, two vectors points, and the radius vector point |
| gdp_id | -108 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

Generalized Drawing Primitives (GDPs) Unfilled GDPs

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arc. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|-------------------------------------|
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc type is invalid in routine **** |

-109 Arc: 2 Points on Arc and Radius

Constant: GKS\$K_GDP_ARC_2PT_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms an arc from the specified beginning and end points, and forms the radius vector point. The arc is drawn so that the circumference, clockwise from the first point to the second, is no greater than π radians (half a circle).

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------------------|--|
| number_of_points | 4 |
| x_coordinates y_ coordinates | Two points, vector origin point, and radius vector point |
| gdp_id | -109 |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

| Argument | Required Value |
|------------------|--|
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arc. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|---|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** |
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc type is invalid in routine **** |

-110 Arc: Center, Starting Point, and Angle

Constant: GKS\$K_GDP_ARC_CTR_PT_ANG

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms an arc by using the distance between the center point and the arc starting point as the radius, and using the angle value to determine the endpoint of the arc.

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 2 |
| x_coordinates y_coordinates | Center and starting point |
| gdp_id | -110 |
| data_record | (5 components) 1 1 0 (address of) arc_type (address of) angle in radians |
| data_record_size | 20 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arc. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|-------------------------|-------------------------------------|
| -159 | DECGKS\$ _ERROR_NEG_159 | Arc type is invalid in routine **** |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

-111 Ellipse: Center, and 2 Axis Vectors

Constant: GKS\$K_GDP_ELLIPSE_CTR_AXES

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the ellipse using a center point, one vector to establish the distance and direction of the first axis, and a second vector to establish the distance of the second axis.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 4 |
| x_coordinates y_coordinates | Center point, vector origin point, minor and major axis vectors. |
| gdp_id | -111 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, one of the vectors may have a length of 0.) |

-113 Ellipse: Focal Points and Point on Circumference

Constant: GKS\$K_GDP_ELLIPSE_FOCII_PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the ellipse using the two focal points and a single point on the circumference.

Generalized Drawing Primitives (GDPs) Unfilled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Two focal points and the point on the circumference. |
| gdp_id | -113 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|----------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, the point may be on the line segment between the focal points.) |

-114 Elliptic Arc: Center, 2 Axis Vectors, and 2 Vectors

Constant: GKS\$K_GDP_ELIARC_CTR_AXES_2VEC

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the elliptic arc using a center point, one axis vector (the largest of the two) to establish the distance and direction of the major axis, a second axis vector to establish the distance of the minor axis, and two vectors whose directions are used to determine the arc end points. The largest axis vector determines both the distance and the direction of the major axis of the elliptic arc.

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|---|
| number_of_points | 6 |
| x_coordinates | Center point, vector origin point, two directional axis vectors, and two end point vectors. |
| y_coordinates | |
| gdp_id | -114 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arcing line. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, due to the vector values, DEC GKS may attempt to form a straight line.) |
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc_type is invalid in routine **** (For instance, if you specify a value other than 1, 2, or 3.) |

-116 Elliptic Arc: Focal Points and 2 Points on Circumference

Constant: GKS\$K_GDP_ELIARC_FOCII_2PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms an elliptic arc using two focal points, the beginning point of the elliptic arc, and the end point as one of the following components:

- The end point, located on the arc
- The second point of a ray (the first point is the first specified focus point of the ellipse), whose intersection with the elliptic path of the arc determines the end point of the arc

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 4 |
| x_coordinates y_coordinates | Two focal points and two points on the circumference. |
| gdp_id | -116 |

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

| Argument | Required Value |
|------------------|--|
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arcing line. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, due to the specified values, DEC GKS may attempt to form a straight line.) |
| -159 | DECGKS\$_ERROR_NEG_159 | Arc_type is invalid in routine **** (For instance, if you specify a value other than 1, 2, or 3.) |

-125 Rectangle: 2 Corners

Constant: GKS\$K_GDP_RECT_2PT

Supporting workstations: All DEC GKS-supported workstations.

Generalized Drawing Primitives (GDPs)

Unfilled GDPs

This GDP forms the rectangle from the specified diagonal corner points. The sides of the rectangle are parallel to the X and Y axes.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|-------------------------|
| number_of_points | 2 |
| x_coordinates | Diagonal corner points. |
| y_coordinates | |
| gdp_id | -125 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the specified points have the same X or Y value, DEC GKS cannot form a rectangle.) |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Filled GDPs

This section describes all filled GDPs. Filled GDPs use the current fill area attributes. You should make sure that the attributes are set to the requirements of your application before you generate these GDPs.

-332 Fill Area Set

Constant: GKS\$K_GDP_FILL_AREA_SET

Supporting workstations: All DEC GKS-supported workstations.

This GDP contains at least three points that together define at least one fill area.

A fill area set consists of one or more fill areas, each consisting of three or more points that may intersect. A fill area set has both interior and edge attributes. Interior attributes are similar to regular fill areas, and edge attributes are similar to polylines. These attributes are set with various GKS escape functions.

The filled regions of a fill area set are determined by the even-odd rule, which considers the entire fill area set as a single primitive. It is therefore possible to create donut-like objects, where the area surrounding the hole is filled.

For more information about fill area and polyline attributes, see Appendix C, DEC GKS Attribute Values.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|-----------------|
| number_of_points | >=3 |
| x_coordinates | x and y points. |
| y_coordinates | |

Generalized Drawing Primitives (GDPs) Filled GDPs

| Argument | Required Value |
|------------------|--|
| gdp_id | -332 |
| data_record | (4 components) number of fill areas (>=1) 0 0 (array of integers (number of points in each fill area)) |
| data_record_size | 16 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the specified points have the same X or Y value, DEC GKS cannot form a rectangle.) |

-333 Filled Circle: Center and Point on Circumference

Constant: GKS\$K_GDP_FCIRCLE_CTR_PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified center point and a single point on the circle's circumference.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|--|
| number_of_points | 2 |
| x_coordinates | Center point and a point on the circumference. |
| y_coordinates | |

Generalized Drawing Primitives (GDPs) Filled GDPs

| Argument | Required Value |
|------------------|----------------|
| gdp_id | -333 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|-------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the center point and the point on the circumference are the same point, DEC GKS cannot form a circle.) |

-334 Filled Circle: 3 Points on Circumference

Constant: GKS\$K_GDP_FCIRCLE_3PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP draws the circle whose circumference includes the three specified points.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|-----------------------------|
| number_of_points | 3 |
| x_coordinates y_coordinates | Three circumference points. |
| gdp_id | -334 |
| data_record | null |
| data_record_size | 0 bytes |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data. (For instance, if the three points form a straight line, DEC GKS cannot generate a corresponding circle.) |

-335 Filled Circle: Center and Radius

Constant: GKS\$K_GDP_FCIRCLE_CTR_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified center point and radius vector value.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 3 |
| x_coordinates y_coordinates | Center point, vector origin point, and radius vector point. |
| gdp_id | -335 |
| data_record | null |
| data_record_size | 0 bytes |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|---|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data. (For instance, if the radius vector specifies a distance of 0, then DEC GKS cannot generate a corresponding circle.) |

-336 Filled Circle: 2 Points on Circumference, and Radius

Constant: GKS\$K_GDP_FCIRCLE_2PT_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a circle from the specified circumference points and the radius vector point. The circle is drawn so that the circumference, clockwise from the first point to the second, is no greater than π radians (half of the circle).

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 4 |
| x_coordinates y_coordinates | Two points, vector origin point, and the radius vector point. |
| gdp_id | -336 |
| data_record | null |
| data_record_size | 0 bytes |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data **** (For instance, if the distance between points is more than twice the specified radius, then DEC GKS cannot form the circle.) |

-338 Filled Arc: Center and 2 Points on Arc

Constant: GKS\$K_GDP_FARC_CTR_2PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms a filled circular arc using the center point, the second point as a starting point of the arc, and the third point as one of the following components:

- The second point, located on the arc
- The second point of a ray (the center point), whose intersection with the circular path of the arc determines the second point of the arc

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Center point and beginning and end points of the arc. |
| gdp_id | -338 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

Generalized Drawing Primitives (GDPs)

Filled GDPs

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|--|
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, the center point and one of the points on the circumference may be the same point.) |
| -159 | DECGKS\$_ERROR_NEG_159 | Arc_type is invalid in routine **** (For instance, if you specify a value other than 2 or 3.) |

-339 Filled Arc: 3 Points on Circumference

Constant: GKS\$K_GDP_FARC_3PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the arc beginning at the first point, running through the second point, and connecting to the third point.

Generalized Drawing Primitives (GDPs) Filled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Three points on the circumference. |
| gdp_id | -339 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|---|
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Fill the area formed by connecting the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, the three points may form a straight line.) |
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc_type is invalid in routine **** (For instance, if you specify any value other than 2 or 3.) |

Generalized Drawing Primitives (GDPs)

Filled GDPs

-340 Filled Arc: Center, 2 Vectors, and a Radius

Constant: GKS\$K_GDP_FARC_CTR_2VEC_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the arc by using the two vectors to calculate directions from the center point. DEC GKS uses the vector directions to form rays that determine the starting and ending points of the arc.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 5 |
| x_coordinates y_coordinates | Center, vector origin point, two vectors, and the radius vector point. |
| gdp_id | -340 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|---|
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Fill the area formed by connecting the beginning and end points of the arc. |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|---|
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc_type is invalid in routine **** (For instance, if you specify any value other than 2 or 3.) |

-341 Filled Arc: 2 Points on Arc, and Radius

Constant: GKS\$K_GDP_FARC_2PT_RAD

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms an arc from the specified beginning and end points, and from the radius vector point. The arc is drawn so that the circumference, clockwise from the first point to the second, is no greater than pi radians (half of a circle).

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 4 |
| x_coordinates y_coordinates | Two points, vector origin point, and radius vector point. |
| gdp_id | -341 |
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

Generalized Drawing Primitives (GDPs)

Filled GDPs

| Value | Constant | Description |
|-------|-----------------------|--|
| 1 | GKS\$K_ARC_TYPE_OPEN | Form an arcing line. |
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Connect the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data **** (For instance, if the distance between the points is more than twice the specified radius, then DEC GKS cannot form the arc.) |
| -159 | DECGKS\$ _ERROR_NEG_ 159 | Arc_type is invalid in routine **** (For instance, if you specify a value other than 1, 2, or 3.) |

-342 Filled Arc: Center, Starting Point, and Angle

Constant: GKS\$K_GDP_FARC_CTR_PT_ANG

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the filled, circular arc by using the distance between the center point and the arc starting point as a radius, and by using the angle value to determine the endpoint of the arc.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|----------------------------|
| number_of_points | 2 |
| x_coordinates | Center and starting point. |
| y_coordinates | |

Generalized Drawing Primitives (GDPs)

Filled GDPs

| Argument | Required Value |
|-------------------------------|--|
| <code>gdp_id</code> | -342 |
| <code>data_record</code> | (5 components) 1 1 0 (address of) <code>arc_type</code> (address of) <code>angle</code> —in radians |
| <code>data_record_size</code> | 20 bytes |

The integer array contains the single element `arc_type`, which can be any of the following values:

| Value | Constant | Description |
|-------|------------------------------------|---|
| 2 | <code>GKS\$K_ARC_TYPE_PIE</code> | Connect both ends of the arc to its center. |
| 3 | <code>GKS\$K_ARC_TYPE_CHORD</code> | Fill the area formed by connecting the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------------------|--|
| -159 | <code>DECGKS\$ _ERROR_NEG_159</code> | <code>Arc_type</code> is invalid in routine **** (For instance, if you specify any value other than 2 or 3.) |

-343 Filled Ellipse: Center, and 2 Axis Vectors

Constant: `GKS$K_GDP_FELLIPSE_CTR_AXES`

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the ellipse using a center point, one axis vector (the largest of the two) to establish the distance and direction of the major axis, and a second axis vector to establish the distance of the minor axis.

Generalized Drawing Primitives (GDPs)

Filled GDPs

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 4 |
| x_coordinates y_coordinates | Center point, vector origin point, and minor and major axis vectors. |
| gdp_id | -343 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, one of the vectors may have a length of 0.) |

-345 Filled Ellipse: Focal Points and Point on Circumference

Constant: GKS\$K_GDP_FELLIPSE_FOCII_PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the ellipse using the two focal points and a single point on the circumference.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Two focal points and the point on the circumference. |

Generalized Drawing Primitives (GDPs) Filled GDPs

| Argument | Required Value |
|------------------|----------------|
| gdp_id | -345 |
| data_record | null |
| data_record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, the point may be on the line segment between the focal points.) |

-346 Filled Elliptic Arc: Center, 2 Axis Vectors, and 2 Vectors

Constant: GKS\$K_GDP_FELIARC_CTR_AXES_2VEC

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the elliptic arc using a center point, one axis vector (the largest of the two) to establish the distance and direction of the major axis, a second axis vector to establish the distance of the minor axis, and two vectors whose directions are used to determine the arc end points.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|---|
| number_of_points | 6 |
| x_coordinates y_coordinates | The center point, vector origin point, two directional axis vectors, and two end point vectors. |
| gdp_id | -346 |

Generalized Drawing Primitives (GDPs)

Filled GDPs

| Argument | Required Value |
|------------------|--|
| data_record | (4 components) 1 0 0 (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|---|
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Fill the area formed by connecting the beginning and end points of the arc. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, due to the vector values, DEC GKS may attempt to form a straight line.) |
| -159 | DECGKS\$_ERROR_NEG_159 | Arc_type is invalid in routine **** (For instance, if you specify any value other than 2 or 3.) |

-348 Filled Elliptic Arc: Focal Points and 2 Points on Circumference

Constant: GKS\$K_GDP_FELIARC_FOCII_2PT

Supporting workstations: All DEC GKS-supported workstations.

Generalized Drawing Primitives (GDPs)

Filled GDPs

This GDP forms the elliptic arc using two focal points, the beginning point of the elliptic arc, and the end point as one of the following components:

- The end point, located on the arc
- The second point of a ray (the first point is the first specified focus point of the ellipse), whose intersection with the elliptic path of the arc determines the end point of the arc

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|---|
| number_of_points | 4 |
| x_coordinates | Two focal points and two points on the circumference. |
| y_coordinates | |
| gdp_id | -348 |
| data_record | (4 components) |
| | 1 |
| | 0 |
| | 0 |
| | (address of) arc_type |
| data_record_size | 16 bytes |

The integer array contains the single element *arc_type*, which can be any of the following values:

| Value | Constant | Description |
|-------|-----------------------|---|
| 2 | GKS\$K_ARC_TYPE_PIE | Connect both ends of the arc to its center. |
| 3 | GKS\$K_ARC_TYPE_CHORD | Fill the area formed by connecting the beginning and end points of the arc. |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, due to the specified values, DEC GKS may attempt to form a straight line.) |
| -159 | DECGKS\$_ERROR_NEG_159 | Arc_type is invalid in routine **** (For instance, if you specify any value other than 2 or 3.) |

-349 Filled Rectangle: Two Corners

Constant: GKS\$_K_GDP_FRECT_2PT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the rectangle from the specified diagonal corner points. The sides of the rectangle are parallel to the X and Y axes.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|-------------------------|
| number_of_points | 2 |
| x_coordinates y_coordinates | Diagonal corner points. |
| gdp_id | -349 |
| data_record | null |
| data_record_size | 0 bytes |

Generalized Drawing Primitives (GDPs)

Filled GDPs

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|--------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_ 158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the specified points have the same X or Y value, DEC GKS cannot form a rectangle.) |

Generalized Drawing Primitives (GDPs)

Cell Array GDPs

Cell Array GDPs

This section describes all cell array GDPs. You need to pass the following points to the cell array GDPs:

- Starting point.
- Diagonal point.
- Point R, which is the third point in the parallelogram moving the starting point to the diagonal point along the X axis. To form a rectangular cell array, make sure that point R has the X value of the diagonal point and the Y value of the starting point.

For more information concerning cell arrays, refer to Chapter 4, Output Functions.

-400 Packed Cell Array

Constant: GKS\$K_GDP_IMAGE_ARRAY

Supporting workstations: All DEC GKS-supported workstations *except* for the PostScript workstations.

This GDP forms a cell array from the starting point, diagonal point, point R, and the contents of a data record. The data record includes an array that contains color indexes specified in 1, 8, or 16 bits. When you specify the color indexes in increments less than a longword, the array uses less memory and DEC GKS can read the data quicker.

GKS\$GDP Arguments:

| Argument | Required Value |
|--------------------------------|--|
| number_of_points | 3 |
| x_coordinates y_coordinates | Starting point, diagonal point, and point R. |

Generalized Drawing Primitives (GDPs) Cell Array GDPs

| Argument | Required Value |
|-------------------------------|--|
| <code>gdp_id</code> | -400 |
| <code>data_record</code> | (4 components) 3 + <code>n_longwords</code> 0 0 (address of) rows, columns, <code>bits_per_index</code> , and <code>color_indexes</code> |
| <code>data_record_size</code> | 16 bytes |

The following list describes the contents of the integer array:

| Component | Description |
|-----------------------------|---|
| Rows | This element is the number of rows in the cell array. |
| Columns | This element is the number of columns in the cell array. |
| <code>Bits_per_index</code> | This element is the number of bits used, within <code>color_indexes</code> , to store a single color index value. (DEC GKS uses the color index value to color the corresponding cell in the cell array.) This value may be 1, 8, or 16. |
| <code>Color_indexes</code> | These components are the contiguous bit increments that specify color indexes. These elements are <code>n_longwords</code> in size and contain the color indexes in row-major order. Color indexes should be specified in row-major order. |

You can calculate the value `n_longwords` using the following formula:

$$\text{INT}((\text{Rows} * \text{Columns} * \text{Bits_per_index} + 31) / 32)$$

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| -158 | DECGKS\$_ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the starting and diagonal points have the same X or Y value, DEC GKS cannot form a cell array rectangle.) |

Generalized Drawing Primitives (GDPs)

Text GDPs

Text GDPs

The following sections describe the text GDPs. Text GDPs use the text attributes. For complete information concerning text, refer to Chapter 4, Output Functions.

-231 Text: Restricted Text Extent Rectangle

Constant: GKS\$K_GDP_RESTRICTED_TEXT

Supporting workstations: All DEC GKS-supported workstations.

This GDP forms the text string within the extent rectangle formed by the specified width and height vectors, and by the text starting point. (This GDP only uses the vectors to determine distance.) This GDP uses the current text height, character spacing, and character expansion factor only if the resulting text string fits within the specified extent rectangle. Otherwise, this GDP chooses the text attributes that form a string that fits within the text extent rectangle. Note that this GDP does not change any of the current text attributes.

GKS\$GDP Arguments:

| Argument | Required Value |
|------------------|---|
| number_of_points | 4 |
| x_coordinates | Starting point, vector origin point, width vector, and height vector. |
| y_coordinates | |
| gdp_id | -231 |
| data_record | (7 components) |
| | 0 |
| | 0 |
| | 1 |
| | null address |
| | null address |
| | (address of) string_length |
| | (address of) string_address |
| data_record_size | 28 bytes |

Generalized Drawing Primitives (GDPs)

Text GDPs

The string length array contains the single element *string_length*, which is the length of your text string. The string address array contains the single element *string_address*, which is the address of your text string.

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| -158 | DECGKS\$ _ERROR_NEG_158 | GDP primitive is not defined by the supplied data in routine **** (For instance, if the distances of the height and width vectors do not form a valid extent rectangle.) |

Escape Functions

Escape Functions

The following sections describe the DEC GKS-supported escape functions. The sections identify each escape by the following:

- The numeric identifier that you pass to ESCAPE.
- The title of the escape (for instance, “Set Display Speed”).
- The valid DEC GKS operating states during which you can use the escape.
- The constant equivalent of the numeric identifier.
- The list of supporting workstations.
- The description of the escape.
- The list of the arguments passed to ESCAPE and the contents of the input and output data records, if applicable. The names of the arguments are identical to the argument descriptions of ESCAPE in Chapter 3, Control Functions.
- The list of escape-specific error messages, if applicable.

Many of the escape data records require that you pass a workstation identifier. In all the data record descriptions that follow, the identifier *ws_id* specifies the workstation identifier component of the record.

Some of the escapes require that you pass a coordinate range as part of the input data record. In all the data record descriptions that follow, the identifier *coord_range* is a set of four real numbers in the following order: ([XMIN,XMAX] x [YMIN,YMAX]). For more information concerning this coordinate range notation, refer to Chapter 1, Introduction to DEC GKS.

The following sections describe the DEC GKS-specific escape functions, by category.

Control Escape Functions

This section describes all the escape functions that affect the workstation as do the DEC GKS control functions. For more information concerning the DEC GKS data structures and control functions, refer to Chapter 3, Control Functions.

-100 Set Display Speed

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_SPEED

Supporting workstations: The LVP16 and all HPGL protocol plotter workstations.

This escape controls the speed of output generation. DEC GKS measures the speed in device coordinate vector/second measurements.

The DEC GKS-supported plotters have pen speeds that are within the range 0.38 cm/second to 38.1 cm/second. The graphics handlers round your increment values to the nearest multiple of 0.38. You can specify the value 0.0 to obtain the default speed of 0.38 cm/second. If you are using one of the plotters to produce acetate slides, the recommended speed is 10 cm/second.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-----------------|---|
| function_id | -100 |
| in_data | (5 components) 1 1 0 (address of) ws_id (address of) display_speed |

Escape functions

Control Escape Functions

| Argument | Required Value |
|----------------------|----------------|
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 |

The real array contains the single element *display_speed*, which is expressed in device coordinate vectors/second. This value must be greater than, or equal to, 0.

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -155 | DECGKS\$_ERROR_NEG_155 | Display speed is less than zero in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-101 Generate Hardcopy of Workstation Surface

Operating states: WSOP, WSAC, SGOP
Constant: GKS\$K_ESC_PRINT

Escape functions

Control Escape Functions

Supporting workstations: The ReGIS devices and Tektronix—4014 workstations. All DEC GKS GKS\$K_WSCAT_OUTIN workstations (terminals).

This escape generates a hardcopy of the currently displayed picture on a printer attached to the workstation.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -101 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |

Escape functions

Control Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-103 Beep

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_BEEP

Supporting workstations: The VAXstation, ReGIS, VT output-only, Tektronix—4014, and Tektronix—4107 workstations.

This escape signals the application user by ringing a bell or by using some other sound generator.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -103 |
| in_data | (5 components) 1 2 0 (address of) ws_id (address of) rel_loudness, sound_duration |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The real array contains the element *rel_loudness*, which is the relative loudness of the sound on a scale from 0.0 (silent) to 1.0 (loudest possible for

Escape functions

Control Escape Functions

the device) and contains the element *sound_duration*, which is the number of seconds to maintain the sound; this value must be greater than or equal to 0.

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| -156 | DECGKS\$_ERROR_NEG_156 | Loudness is outside the range [0,1] in routine **** |
| -157 | DECGKS\$_ERROR_NEG_157 | Duration is less than zero in routine **** |
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-106 Pop Workstation

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_POP_WORKSTATION

Supporting workstations: The VAXstation workstations.

This escape places the display window containing the specified workstation in front of all other display windows. Remember that if you pop a workstation window, you pop all input windows associated with that workstation.

Escape functions

Control Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -106 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-107 Push Workstation

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_PUSH_WORKSTATION

Supporting workstations: The VAXstation workstations.

Escape functions Control Escape Functions

This escape places the display window containing the specified workstation behind all other display windows. Remember that if you push a workstation window, you push all input windows associated with that workstation.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -107 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Escape functions

Control Escape Functions

-108 Set Error Handling Mode

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_ERR_HANDLING_MODE

Supporting workstations: All workstations.

This escape allows you to suppress as much error checking as possible if set to GKS\$K_ERROR_OFF. Otherwise, GKS executes normally and logs errors as necessary, returning those errors specified by standard and internal errors.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -108 |
| in_data | (4 components) 1 0 0 error_mode (GKS\$K_ERROR_OFF) or (GKS\$K_ERROR_ON) |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |

Escape functions Control Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-109 Set Viewport Event

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_VIEWPORT_EVENT

Supporting workstations: The VAXstation workstations.

This escape allows an application to receive events that the workstation viewport has changed in some way. These events are reported through the input event queue with the input class constant GKS\$K_INPUT_CLASS_VIEWPORT. There is no corresponding GET INPUT function or escape. The event simply indicates that something in the workstation viewport has changed.

The application can use the appropriate workstation inquiry functions to determine what values have actually changed. This type of event is normally reported where the GKS workstation is implemented in a windowing environment. The user may change the workstation viewport through the window system. The DEC GKS VAXstation (UIS) workstation type and the DECwindows series of workstation types are windowing environments where this event can be reported.

Escape functions

Control Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -109 |
| in_data | (4 components) 2 0 0 (address of) ws_id, on_off |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the elements *ws_id*, the workstation identifier for which the value should be set, and *on_off*, used to turn on or off the reporting of the change in the workstation viewport. GKS\$K_TRUE turns it on; GKS\$K_FALSE turns it off.

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-110 Associated Workstation Type Connection ID

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_ASSOC_WSTYPE_CONID

Supporting workstations: All workstations.

This escape establishes a connection identifier for a specified workstation type. When an inquiry function references the workstation after this connection identifier is set, the workstation returns the workstation type and the connection identifier, treating them as a pair, where this pairing is possible and relevant. In addition, this escape may cancel an association rather than set one.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -110 |
| in_data | (7 components) 2 0 1 (address of) ws_type, set 0 (address of) length conid (address of) conid |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Escape functions

Control Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-111 Software Clipping

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_SOFT_CLIP

Supporting workstations: All workstations.

Because some hardware may not correctly clip very large primitives, you can force GKS to use software clipping, in some cases. GKS uses hardware clipping by default if the graphics device has this capability.

The second integer parameter (*flag*) controls this behavior. If it is set to GKS\$K_TRUE, software clipping is always used. If it is set to GKS\$K_FALSE, software clipping is used only if hardware clipping is unavailable.

Escape functions

Control Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -111 |
| in_data | (4 components) 2 0 0 (address of) ws_id, flag |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Escape functions

Output, Attribute, and Transformation Escape Functions

Output, Attribute, and Transformation Escape Functions

This section describes all the escape functions that affect the generation of specific output primitives. For more information concerning DEC GKS output and the corresponding output attributes, refer to Chapter 3, Control Functions, and to Chapter 5, Output Attribute Functions.

Some of the escape functions described in this section refer to “entries.” In all instances, these refer to DEC GKS state list entries. For more information concerning the DEC GKS state list, refer to Chapter 3, Control Functions.

-150 Set Writing Mode

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_WRITING_MODE

Supporting workstations: The VAXstations, ReGIS, VT output-only, and LCG01 workstations.

This escape sets the current *writing mode* entry for all subsequently drawn primitives that use this facility. An example of a writing mode is complement mode, which reverses the foreground and background colors when text is generated.

The initial writing mode is mode 1, which is workstation dependent. If a workstation cannot implement a specified writing mode, DEC GKS uses mode number 1.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|---|
| function_id | -150 |
| in_data | (4 components) 1 0 0 (address of) wr_mode |

Escape functions

Output, Attribute, and Transformation Escape Functions

| Argument | Required Value |
|-----------------------------------|----------------|
| <code>in_data_size</code> | 16 bytes |
| <code>out_buffer</code> | null |
| <code>record_buffer_length</code> | NA |
| <code>record_size</code> | 0 bytes |

The integer array contains the single element *wr_mode*, which can be one of the following values:

| Mode | Description |
|------|---|
| <=1 | Workstation dependent. |
| 2 | Complement mode (GKS\$K_WRT_MODE_COMPLEMENT). |
| 3 | Erase underlying characters (GKS\$K_WRT_MODE_ERASE). |
| 4 | Overlay on underlying characters (GKS\$K_WRT_MODE_OVERLAY). |
| >=5 | Reserved for future use. |

-151 Set Line Cap Style

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_LINE_CAP

Supporting workstations: The PostScript workstations.

This escape sets the current *line cap style* entry for all subsequently drawn polylines that use this facility. The line cap style determines the appearance of the polyline endpoints.

The initial line cap style is style 1, which is workstation dependent. If a workstation cannot implement a specified style, DEC GKS uses style number 1.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -151 |
| in_data | (4 components) 1 0 0 (address of) cap_style |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the single element *cap_style*, which can be one of the following values:

| Style | Description |
|-------|---|
| <=1 | Workstation dependent. |
| 2 | Butt, squared at the endpoint (GKS\$K_LINE_CAP_BUTT). |
| 3 | Round, semicircular arc (GKS\$K_LINE_CAP_ROUND). |
| 4 | Square, projecting square cap (GKS\$K_LINE_CAP_SQUARE). |
| >=5 | Reserved for future use. |

-152 Set Line Join Style

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_LINE_JOIN

Supporting workstations: The PostScript workstations.

This escape sets the current *line join style* entry for all subsequently drawn polylines that use this facility. The line join style determines the appearance of the polyline vertices.

Escape functions

Output, Attribute, and Transformation Escape Functions

The initial line join style is style 1, which is workstation dependent. If a workstation cannot implement a specified style, DEC GKS uses style number 1.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -152 |
| in_data | (4 components) 1 0 0 (address of) join_style |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the single element *join_style*, which can be one of the following values:

| Style | Description |
|-------|--|
| <=1 | Workstation dependent. |
| 2 | Mitre, outer edges meet at a sharp point (GKS\$K_LINE_JOIN_MITRE). |
| 3 | Round, circular arc at point (GKS\$K_LINE_JOIN_ROUND). |
| 4 | Beveled, a short, third line connecting lines not joined at ninety degrees (GKS\$K_LINE_JOIN_BEVEL). |
| >=5 | Reserved for future use. |

-153 Set Edge Control Flag

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_CTL

Supporting workstations: All DEC GKS-supported workstations.

Escape functions

Output, Attribute, and Transformation Escape Functions

This escape sets the *current edge control flag* entry. If the *current edge control flag ASF* is set to GKS\$K_ASF_INDIVIDUAL, then the Fill Area Set GDP (see the GDP description in the previous section) uses an edge for subsequently generated fill areas.

The initial edge control setting is GKS\$K_EDGE. (For more information concerning ASFs, refer to Chapter 5, Output Attribute Functions.)

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -153 |
| in_data | (4 components) 1 0 0 (address of) edge_flag |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the single element *edge_flag*, which can be one of the following values:

| Style | Description |
|-------|--|
| 0 | Fill Area Set does not use an edge (GKS\$K_NOEDGE). |
| 1 | Fill Area Set uses an edge if the <i>current edge control flag ASF</i> is GKS\$K_ASF_INDIVIDUAL (GKS\$K_EDGE). |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

Escape functions

Output, Attribute, and Transformation Escape Functions

-154 Set Edge Type

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_TYPE

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the *current edge type flag* entry. If the *current edge type flag ASF* is set to GKS\$K_ASF_INDIVIDUAL, then the Fill Area Set GDP (see the GDP description in the previous section) uses the current edge type for subsequently generated edges.

The initial edge type is type 1, which is a solid line. If a workstation cannot implement a specified type, DEC GKS uses type number 1.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -154 |
| in_data | (4 components) 1 0 0 (address of) edge_type |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the single element *edge_type*, which can be one of the following values:

| Style | Description |
|-------|-----------------------------------|
| <=0 | Workstation dependent. |
| 1 | Solid edge (GKS\$K_EDGE_SOLID). |
| 2 | Dashed edge (GKS\$K_EDGE_DASHED). |

Escape functions

Output, Attribute, and Transformation Escape Functions

| Style | Description |
|-------|---|
| 3 | Dotted edge (GKS\$K_EDGE_DOTTED). |
| 4 | Dashed-dotted edge (GKS\$K_EDGE_DASHED_DOTTED). |
| >=5 | Reserved for future use. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-155 Set Edge Width Scale Factor

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_WIDTH

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the *current edge width factor flag* entry. If the *current edge width scale factor flag ASF* is set to GKS\$K_ASF_INDIVIDUAL, then the Fill Area Set GDP (see the GDP description in the previous section) uses the current edge scale factor for subsequently generated edges.

The initial edge width is 1.0. An edge width scale factor of 0.0 produces the thinnest edge available on a workstation. DEC GKS multiplies the edge width scale factor with the nominal edge width and rounds to the nearest width available on the workstation.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -155 |
| in_data | (5 components) 0 1 0 0 (address of) scale_factor |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-156 Set Edge Color Index

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_COLOR_INDEX

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the *current edge color index* entry. If the *current edge color index ASF* is set to GKS\$K_ASF_INDIVIDUAL, then the Fill Area Set GDP (see the GDP description in the previous section) uses the current edge color index for subsequently generated edges.

The initial edge color index is 1. If a workstation cannot implement a specified color index, the device handler uses a workstation-dependent color.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -156 |
| in_data | (4 components) 1 0 0 (address of) color_index |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-157 Set Edge Index

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_INDEX

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the *current edge index* entry. The Fill Area Set GDP uses the index as a pointer into the edge bundle table whenever an edge attribute ASF is set to GKS\$K_ASF_BUNDLED. All device handlers must provide at least five edge bundle indexes. (To review your device's edge bundle table, refer to the device-specific appendixes in this manual.)

The initial edge index is 1. If a workstation cannot locate a specified edge index in the edge bundle table, the device handler uses a workstation-dependent index number.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -157 |
| in_data | (4 components) 1 0 0 (address of) bundle_index |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$ _ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-158 Set Edge Aspect Source Flag (ASF)

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_ASF

Supporting workstations: All DEC GKS-supported workstations.

This escape establishes the aspect source flag (ASF) setting for each of the edge attributes. Each flag can either be GKS\$K_ASF_BUNDLED (value 0—use the bundled attributes), or GKS\$K_ASF_INDIVIDUAL (value 1—use the individual attributes). All initial ASF flags are set to GKS\$K_ASF_INDIVIDUAL.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -158 |
| in_data | (4 components) 4 0 0 (address of) asf_flags |
| in_data_size | 16 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the elements *asf_array*, each of which contain either GKS\$K_ASF_BUNDLED or GKS\$K_ASF_INDIVIDUAL for each of the edge attribute settings, in the following order:

1. Edge control flag (see the Set Edge Control Flag escape in this section)
2. Edge type flag (see the Set Edge Type escape in this section)
3. Edge width scale factor flag (see the Set Edge Width Scale Factor escape in this section)
4. Edge color index flag (see the Set Edge Color Index escape in this section)

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

Escape functions

Output, Attribute, and Transformation Escape Functions

-160 Begin Transformation Block

Operating states: WSOP, WSAC

Constant: GKS\$K_ESC_BEGIN_TRANS_BLOCK

Supporting workstations: All DEC GKS-supported workstations.

This escape applies the specified transformation to all subsequently drawn primitives not contained in segments. The transformation continues until you call the End Transformation Block escape function (see the escape description in this section) or until you open a segment.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -160 |
| in_data | (5 components) 1 6 0 (address of) ws_id (address of) xform |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The real array contains the elements *xform*, which are the values for the segment transformation matrix. For more information, refer to the description of GKS\$EVAL_XFORM_MATRIX in Chapter 8, Segment Functions.

Escape functions

Output, Attribute, and Transformation Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 6 | GKS\$_ERROR_6 | GKS not in proper state: GKS shall be either in the state WSOP or in the state WSAC in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-161 End Transformation Block

Operating states: WSOP, WSAC

Constant: GKS\$_K_ESC_END_TRANS_BLOCK

Supporting workstations: All DEC GKS-supported workstations.

This escape ends the transformation process initiated by the call to the Begin Transformation Block escape function (see the escape description in this section).

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|---|
| function_id | -161 |
| in_data | (4 components) 1 0 0 (address of) ws_id |

Escape functions

Output, Attribute, and Transformation Escape Functions

| Argument | Required Value |
|----------------------|----------------|
| in_data_size | 0 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 6 | GKS\$_ERROR_6 | GKS not in proper state: GKS shall be either in the state WSOP or in the state WSAC in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-162 Set Segment Highlighting Method

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_SEG_HIGH_METHOD

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the segment highlighting method, but it does not change the highlighted state of a segment. Use the SET HIGHLIGHTING function to change the segment highlighted state. If the segment is currently highlighted when this escape is called, and the segment highlighting method or attributes are different, the segment is unhighlighted and then highlighted again with the new attributes. This function may also cause a regeneration of the workstation display, depending on the workstation regeneration mode.

Escape functions

Output, Attribute, and Transformation Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -162 |
| in_data | (5 components) 6 2 0 (address of) segment name, highlighting method, highlighting_color_index, highlighting_line_type, highlighting_fill_style, highlighting_fill_index (address of) highlighting_line_width, expand_extent_factor |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The integer array contains the element *segment_name*, which is the name of the segment for which the highlighting attributes are to be set. The *highlighting_method* element is also part of this integer array, and is one of the following constants:

| Constant | Value | Description |
|----------------------------|-------|---|
| GKS\$K_HIGH_METHOD_DEFAULT | 0 | Use the workstation-dependent default highlighting method. |
| GKS\$K_HIGH_METHOD_COMP | 1 | Highlight the segment by drawing it in complement mode. |
| GKS\$K_HIGH_METHOD_COLOR | 2 | Highlight the segment by drawing it using the color index specified in the integer array. |

Escape functions

Output, Attribute, and Transformation Escape Functions

| Constant | Value | Description |
|-------------------------|-------|---|
| GKS\$K_HIGH_METHOD_LINE | 3 | Highlight the segment by drawing an extent box around it, using the line attributes specified in the integer and float arrays. The extent box is normally drawn using complement mode. The extent will be expanded by <i>expand_extent_factor</i> times the nominal line width. |
| GKS\$K_HIGH_METHOD_FILL | 4 | Highlight the segment by drawing a complement mode fill area around it, using the fill area attributes specified in the integer array. The extent box will be expanded by <i>expand_extent_factor</i> times the nominal line width. |
| GKS\$K_HIGH_METHOD_DUAL | 5 | Highlight the segment by drawing both a line and fill area around it, using the attributes specified. The extent box will be expanded by <i>expand_extent_factor</i> times the nominal line width. If the highlighting method is not available on the workstation on which the segment is displayed, the default value of 1 is used. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 120 | GKS\$_ERROR_120 | Specified segment name is invalid in routine **** |

Escape functions

Output, Attribute, and Transformation Escape Functions

-163 Set Highlighting Method

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_HIGH_METHOD

Supporting workstations: All DEC GKS-supported workstations.

This escape sets the primitive highlighting method to be used for pick highlighting. This information is meaningful only on an OUTIN workstation. All subsequent primitives until the next usage of this escape are highlighted in the manner specified.

If you use pick prompt and echo type 1, the information applies to each primitive and is stored when they are created. If you use pick prompt and echo type 2, the information applies to the group of primitives with the same pick identifier. The information is stored the first time a primitive with a different pick identifier than any other primitive in a particular segment is stored. If you use pick prompt and echo type, the information is stored when a segment is created and applies to all primitives within a particular segment.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -163 |
| in_data | (5 components) 5 2 0 (address of) highlighting_method, highlighting_color_index, highlighting_line_type, highlighting_fill_style, highlighting_fill_index (address of) highlighting_line_width, expand_extent_factor |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

Escape functions

Output, Attribute, and Transformation Escape Functions

The integer array contains the element *highlighting_method*, which is the method to use for highlighting all subsequently stored primitives, and is one of the following constants:

| Constant | Value | Description |
|----------------------------|-------|--|
| GKS\$K_HIGH_METHOD_DEFAULT | 0 | Use the workstation-dependent default highlighting method. |
| GKS\$K_HIGH_METHOD_COMP | 1 | Highlight the primitive by drawing it in complement mode. |
| GKS\$K_HIGH_METHOD_COLOR | 2 | Highlight the primitive by drawing it using the color index specified in the integer array. |
| GKS\$K_HIGH_METHOD_LINE | 3 | Highlight the primitive by drawing an extent box around it, using the line attributes specified in the integer and float arrays. The extent box is normally drawn using complement mode. The extent will be expanded by <i>expand_extent_factor</i> times the nominal line width. |
| GKS\$K_HIGH_METHOD_FILL | 4 | Highlight the primitive by drawing a complement mode fill area around it, using the fill area attributes specified in the integer array. The extent box will be expanded by <i>expand_extent_factor</i> times the nominal line width. |
| GKS\$K_HIGH_METHOD_DUAL | 5 | Highlight the primitive by drawing both a line and fill area around it, using the attributes specified. The extent box will be expanded by <i>expand_extent_factor</i> times the nominal line width. If the highlighting method is not available on the workstation on which the primitive is displayed, the default value of 1 is used. |

Escape functions

Output, Attribute, and Transformation Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-200 Set Edge Representation

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_EDGE_REP

Supporting workstations: All DEC GKS-supported workstations.

This escape function sets or resets the representation for a given edge bundle index. For more information concerning representations, refer to Chapter 5, Output Attribute Functions.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -200 |
| in_data | (5 components) 5 1 0 (address of) ws_id, edge_index, edge_atts (address of) scale_factor |
| in_data_size | 20 bytes |
| out_buffer | null |
| record_buffer_length | NA |
| record_size | 0 bytes |

The real array contains the single element *scale_factor*, which is the edge width scale factor to be associated with the specified edge bundle index.

Escape functions

Output, Attribute, and Transformation Escape Functions

The integer array contains the element *edge_index*, which is the bundle index that you wish to define or redefine; the element *scale_factor*, which is the edge width scale factor to be associated with the bundle index; and the elements *edge_atts*, which must contain the following:

1. Edge control flag (see the Set Edge Control Flag escape in this section)
2. Edge type flag (see the Set Edge Type escape in this section)
3. Edge color index flag (see the Set Edge Color Index escape in this section)

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 92 | DECGKS\$_ERROR_NEG_92 | Color index is less than zero in routine **** |

Escape functions

Output, Attribute, and Transformation Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| -150 | DECGKS\$_ERROR_NEG_150 | Edge index is less than 1 in routine **** |
| -151 | DECGKS\$_ERROR_NEG_151 | Edge width scale factor is less than zero in routine **** |
| -162 | DECGKS\$_ERROR_NEG_162 | Edge index is invalid in routine **** |
| -163 | DECGKS\$_ERROR_NEG_163 | Specified edge type is not supported on this workstation in routine **** |

DEC GKS DECwindows Escape Functions

This section describes all the escape functions that affect the DECwindows device. For more information concerning DEC GKS and DECwindows, refer to Chapter 12, DECwindows Workstation Specifics, in the *DEC GKS Device Specifics Reference Manual*.

-202 Set Window Title

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_WINDOW_TITLE

Supporting workstations: All workstations.

This escape changes the string displayed in the title bar. This change applies to workstation types 210 and 211.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -202 |
| in_data | (7 components) 1 0 1 (address of) ws_id 0 (address of) length of new title (address of) new title |
| in_data_size | 28 bytes |
| out_buffer | null |
| record_buffer_length | N/A |
| record_size | 0 bytes |

Escape functions

DEC GKS DECwindows Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-203 Set Reset String

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_RESET_STRING

Supporting workstations: All workstations.

This escape changes the string displayed in the reset button on the menu bar, and applies only to workstation types 210 and 211.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -203 |
| in_data | (7 components) 1 0 1 (address of) ws_id 0 (address of) length of new string (address of) new string for reset button |
| in_data_size | 28 bytes |
| out_buffer | null |
| record_buffer_length | N/A |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Escape functions

DEC GKS DECwindows Escape Functions

-204 Set Cancel String

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_CANCEL_STRING

Supporting workstations: All workstations.

This escape sets the string used by the cancel buttons of input devices. It has no effect on input devices presently displayed, and applies only to workstation type 211.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -204 |
| in_data | (7 components) 1 0 1 (address of) ws_id 0 (address of) length of new string (address of) new string for cancel buttons |
| in_data_size | 28 bytes |
| out_buffer | null |
| record_buffer_length | N/A |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |

Escape functions DEC GKS DECwindows Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-205 Set Enter String

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_ENTER_STRING

Supporting workstations: All workstations.

This escape sets the string used by the enter buttons of input devices. It has no effect on input devices presently displayed, and applies only to workstation type 211.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|---|
| function_id | -205 |
| in_data | (7 components) |
| | 1 |
| | 0 |
| | 1 |
| | (address of) ws_id |
| | 0 |
| | (address of) length of new string |
| | (address of) new string for enter buttons |

Escape functions

DEC GKS DECwindows Escape Functions

| Argument | Required Value |
|----------------------|----------------|
| in_data_size | 28 bytes |
| out_buffer | null |
| record_buffer_length | N/A |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-206 Set Icon Bitmaps

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_ICON_BITMAPS

Supporting workstations: DECwindows Workstations 210 and 211.

Icon bitmaps are defined one integer per pixel. The integer values are used in a device-dependent manner to determine the icon appearance. Where possible, the integers will specify the GKS color indexes to be used for each pixel. The pixels are specified in row-major order, with pixel (0,0) being the upper left corner of the icon (left-to-right, then top-to-bottom).

Escape functions

DEC GKS DECwindows Escape Functions

If the icon height and width are specified as 0, then the default icon bitmap will be used instead, and the Bitmap Definition string for that icon must not be specified.

Some devices may not need more than one icon bitmap, in which case only the Small Icon should be specified. The Large Icon Height and Large Icon Width parameters should be set to 0.

This escape is currently supported on DECwindows workstations only. For the DECwindows workstations, the normal bitmap sizes are 17x17 and 32x32. Pixels specified with color 0 appear in the icon background color. Pixels specified as non-zero appear in the icon foreground color.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|--|
| function_id | -206 |
| int_data | (4 components) $5 + \text{Small_Icon_Width} * \text{Small_Icon_Height} +$ $\text{Large_Icon_Width} * \text{Large_Icon_Height}$ 0 0 (address of) ws_id, Small_Icon_Width (in pixels), Small_Icon_Height (in pixels), Large_Icon_Width (in pixels), Large_Icon_Height (in pixels), Small_Icon_Data (Small_Icon_Width * Small_Icon_Height integers), Large_Icon_Data (Large_Icon_Width * Large_Icon_Height integers) |
| in_data_size | 16 bytes |
| out_buffer | null |

Escape functions

DEC GKS DECwindows Escape Functions

| Argument | Required Value |
|----------------------|----------------|
| record_buffer_length | NA |
| record_size | 0 |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-304 Inquire Window Identifiers

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_WINDOW_IDS

Supporting workstations: All workstations.

This escape returns the display and window identifiers of the GKS output window.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -304 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 2 0 0 (address of) X Display id, X Window id |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Escape functions

DEC GKS DECwindows Escape Functions

-307 Inquire Pasteboard Identifier

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_PASTEBOARD_ID

Supporting workstations: DECwindows Workstations 210 and 211.

This escape is for VMS only.

This escape returns the widget identifier of the GKS pasteboard widget. The pasteboard is a dialog box that contains the GKS output window widget and all input widgets. You should not change the size of this widget. This escape applies only to workstation types 210 and 211.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -307 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 1 0 0 (address of) Pasteboard Widget id |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Escape functions

DEC GKS DECwindows Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-308 Inquire Menu Bar Identifier

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_MENU_BAR_ID

Supporting workstations: DECwindows Workstations 210 and 211.

This escape is for VMS only.

This escape returns the widget identifier of the menu bar widget, and it applies only to workstation types 210 and 211.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -308 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 1 0 0 (address of) Menu Bar Widget id |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

Escape functions

DEC GKS DECwindows Escape Functions

-309 Inquire Shell Identifier

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_SHELL_ID

Supporting workstations: DECwindows Workstations 210 and 211.

This escape is for VMS only.

This escape returns the widget identifier of the GKS application shell widget, and it applies only to workstation types 210 and 211.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -309 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 1 0 0 (address of) Shell Widget id |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Escape functions

DEC GKS DECwindows Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-500 Set Double Buffering

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_DOUBLE_BUFFER

Supporting Workstations: DECwindows workstations.

This escape sets the double buffering state to either ON or OFF.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|--|
| function_id | -500 |
| in_data | (4 components) 2 0 0 (address of) wsid, control_flag |
| in_data_size | 16 bytes |

Escape functions

DEC GKS DECwindows Escape Functions

| Argument | Required Value |
|----------------------|----------------|
| out_buffer | null |
| return_buffer_length | NA |
| record_size | 0 bytes |

The following list describes the integer array contents:

| Element | Description |
|--------------|--|
| wsid | This element is the workstation identifier. |
| control_flag | This element is the double buffering control flag. If it is GKS\$K_TRUE(1), double buffering is enabled. If it is GKS\$K_FALSE(0), double buffering is disabled. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

-501 Set Background Pixmap

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_SET_BCKGRND_PIXMAP

Supporting Workstations: DECwindows workstations.

For DECwindows workstations, the pixmap is defined as the background image. When you specify a background pixmap, GKS creates its own pixmap and copies the application-specified background pixmap to its own pixmap. After this call, GKS never references the application pixmap.

To modify the background pixmap, inquire the GKS background pixmap identifier (GKS\$K_ESC_INQ_BCKGRND_PIXMAP) and then modify it.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -501 |
| in_data | (4 components) 2 0 0 (address of) wsid, pixmap_id |
| in_data_size | 16 bytes |
| out_buffer | null |
| return_buffer_length | NA |
| record_size | 0 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

-502 Inquire Double Buffer Pixmap

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_DBUFFER_PIXMAP

Supporting Workstations: DECwindows workstations.

For the DECwindows workstations, when double buffering is enabled, GKS writes to a pixmap. This escape queries the X pixmap ID so the application can write to that pixmap.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -502 |
| in_data | (4 components) 1 0 0 (address of) wsid |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 1 0 0 (address of) pixmap_id |
| return_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

-503 Inquire Background Pixmap

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_BCKGRND_PIXMAP

Supporting Workstations: DECwindows workstations.

For the DECwindows workstations, when double buffering is enabled and a background pixmap is defined, this escape allows the application to query the pixmap identifier.

Escape functions

DEC GKS DECwindows Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -503 |
| in_data | (4 components) 1 0 0 (address of) wsid |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 1 0 0 (address of) pixmap_id |
| return_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

Escape functions

DEC GKS State List Inquiry Escape Functions

DEC GKS State List Inquiry Escape Functions

This section describes all the escape functions that inquire about information in the DEC GKS state list. All the inquiry functions in this section write an integer value called *error_status* to the output data record. If *error_status* is the value 0, then the rest of the output data record is valid. If *error_status* is not the value 0, then the rest of the output data record is invalid.

For more information concerning the DEC GKS state list, refer to Chapter 3, Control Functions. For more information concerning the DEC GKS inquiry functions and the *error_status* argument, refer to Chapter 11, Inquiry Functions.

-251 Inquire Current Writing Mode

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_WRITING_MODE

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the value of the current writing mode (see the Set Writing Mode escape description in this chapter) to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -251 |
| in_data | null |
| in_data_size | 0 bytes |
| out_buffer | (4 components) 2 0 0 (address of) error_status, writing_mode |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Escape functions

DEC GKS State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$ _ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-252 Inquire Current Line Cap Style

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_LINE_CAP

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the value of the current line cap style (see the Set Line Cap Style escape description in this chapter) to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -252 |
| in_data | null |
| in_data_size | 0 bytes |
| out_buffer | (4 components) 2 0 0 (address of) error_status, cap_style |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Escape functions

DEC GKS State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-253 Inquire Current Line Join Style

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_LINE_JOIN

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the value of the current line join style (see the Set Line Join Style escape description in this chapter) to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -253 |
| in_data | null |
| in_data_size | 0 bytes |
| out_buffer | (4 components) 2 0 0 (address of) error_status, join_style |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

Escape functions

DEC GKS State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-254 Inquire Current Edge Attributes

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$_K_ESC_INQ_EDGE_ATTR

Supporting workstations: All DEC GKS-supported workstations.

This escape writes all the value of the current edge attributes to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -254 |
| in_data | null |
| in_data_size | 0 bytes |
| out_buffer | (5 components) 9 1 0 (address of) error_status, edge_atts (address of) scale_factor |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Escape functions

DEC GKS State List Inquiry Escape Functions

The real array contains the single element *scale_factor*, which is the current edge width scale factor. The integer array contains the elements *edge_atts*, which must contain the following:

1. Current edge index (see the Set Edge Index escape in this chapter)
2. Current edge control flag (see the Set Edge Control Flag escape in this chapter)
3. Current edge type flag (see the Set Edge Type escape in this chapter)
4. Current edge color index flag (see the Set Edge Color Index escape in this chapter)

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| -160 | DECGKS\$_ERROR_NEG_160 | Insufficient space in escape output data record arrays in routine **** |

Escape functions

Workstation State List Inquiry Escape Functions

Workstation State List Inquiry Escape Functions

This section describes all the escape functions that inquire about information in the workstation state list. All the inquiry functions in this section write an integer value called *error_status* to the output data record. If *error_status* is the value 0, then the rest of the output data record is valid. If *error_status* is not the value 0, then the rest of the output data record is invalid.

For more information concerning the workstation state list, refer to Chapter 3, Control Functions. For more information concerning the DEC GKS inquiry functions and the *error_status* argument, refer to Chapter 11, Inquiry Functions.

-255 Inquire Viewport Data

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_VIEWPORT_DATA, GEIVD (FORTRAN)

Supporting workstations: DECwindows workstations.

This escape returns the region of the workstation that has been changed or exposed since the last time this escape was called.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|---|
| function_id | -255 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |

Escape functions

Workstation State List Inquiry Escape Functions

| Argument | Required Value |
|----------------------|---|
| out_buffer | (5 components) 1 4 0 (address of) error_status (address of) viewport |
| record_buffer_length | 20 |
| record_size | 20 bytes |

The input integer array contains the single element *ws_id*, which is the workstation identifier.

The following list describes the contents of the output integer array:

| Component | Description |
|--------------|--|
| error_status | This element is the inquiry error status. |
| viewport | This element is the viewport defining the region changed or exposed, in device coordinates (XMIN, XMAX, YMIN, YMAX). |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 182 | GKS\$ _ERROR_182 | Contents of escape data record are invalid in routine **** |

Escape functions

Workstation State List Inquiry Escape Functions

-300 Inquire Current Display Speed

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_SPEED

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the current display speed to the output data record. (See the Set Display Speed escape in this chapter.)

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -300 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 1 1 0 (address of) error_status (address of) display_speed |
| record_buffer_length | 20 |
| record_size | 20 bytes |

Escape functions

Workstation State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-302 Inquire List of Edge Indexes

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$_K_ESC_INQ_LIST_EDGE_INDEXES

Supporting workstations: All DEC GKS-supported workstations.

This escape returns the list of indexes supported by a given workstation.
(See the Set Edge Index escape in this chapter.)

Escape functions

Workstation State List Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -302 |
| in_data | (4 components) 1 0 0 (address of) ws_id |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 3 + num_indexes 0 0 (address of) error_status, total_indexes, returned_indexes, index_list |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

The following list describes the integer array contents:

| Component | Description |
|------------------|---|
| error_status | This element is the inquiry error status. |
| total_indexes | This element is the total number of edge indexes supported by the workstation. |
| returned_indexes | This element is the number of edge indexes written to the remaining elements of the output data record's integer array. |
| index_list | These elements are the edge indexes. |

Escape functions

Workstation State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|-------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$ _ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| -160 | DECGKS\$ _ERROR_NEG_160 | Insufficient space in escape output data record arrays in routine **** |

-303 Inquire Segment Extent

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_SEGMENT_EXTENT

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the coordinate range of the segment extent rectangle corresponding to the specified segment name. For more information concerning segment names, refer to Chapter 8, Segment Functions.

This escape will not report correct values if the segment is off the display surface. This is a design restriction.

Escape functions

Workstation State List Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -303 |
| in_data | (4 components) 2 0 0 (address of) ws_id, segment_id |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 1 4 0 (address of) error_status (address of) coord_range |
| record_buffer_length | 20 |
| record_size | 20 bytes |

The real array contains the elements *coord_range*, which are the four world coordinate values of the segment's extent rectangle, using the current transformation values.

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$ _ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$ _ERROR_25 | Specified workstation is not open in routine **** |

Escape functions

Workstation State List Inquiry Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |

-305 Inquire Segment Highlighting Method

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_INQ_SEG_HIGH_METHOD

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the values of the segment highlighting method and attributes to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|--|
| function_id | -305 |
| in_data | (4 components) 1 0 0 (address of) segment name |
| in_data_size | 16 bytes |

Escape functions

Workstation State List Inquiry Escape Functions

| Argument | Required Value |
|----------------------|---|
| out_buffer | (5 components) 6 2 0 (address of) error_status, highlighting method, highlighting_color_index, highlighting_line_type, highlighting_fill_style, highlighting_fill_index (address of) highlighting_line_width, expand_extent_factor |
| record_buffer_length | 20 |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$ _ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 120 | GKS\$ _ERROR_120 | Specified segment name is invalid in routine **** |

-306 Inquire Highlighting Method

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_INQ_HIGH_METHOD

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the values of the current primitive highlighting method and attributes to its output data record.

Escape functions

Workstation State List Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -306 |
| in_data | null |
| in_data_size | 0 bytes |
| out_buffer | (5 components) 6 2 0 (address of) error_status, highlighting method, highlighting_color_index, highlighting_line_type, highlighting_fill_style, highlighting_fill_index (address of) highlighting_line_width, expand_extent_factor |
| record_buffer_length | 20 |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-358 Inquire List of Highlighting Method

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_INQ_LIST_HIGH

Supporting workstations: All DEC GKS-supported workstations.

This escape returns the list of supported segment and primitive highlighting methods.

Escape functions

Workstation State List Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -358 |
| in_data | (4 components) 1 0 0 (address of) ws_type |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 3 + total_num_high_methods 0 0 (address of) error_status, total_num_high_methods, returned_high_methods, high_methods_list |
| record_buffer_length | 16 |
| record_size | 16 bytes |

The following list describes the integer array contents:

| Component | Description |
|------------------------|--|
| error_status | This element is the inquiry error status. |
| total_num_high_methods | This element is the total number of highlighting methods supported by the workstation type. |
| return_high_methods | This element is the number of highlighting methods written to the remaining elements of the output data record's integer array |
| hig_methods_list | These elements are the highlighting methods. |

Escape functions

Workstation State List Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 8 | GKS\$ _ERROR_8 | GKS not in proper state: GKS must be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |

-359 Inquire Edge Representation

Operating states: WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_EDGE_REP

Supporting workstations: All DEC GKS-supported workstations.

This escape returns the edge bundle representation for a given workstation.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -359 |
| in_data | (4 components) 3 0 0 (address of) ws_id, bundle_index, set_realized |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 4 1 0 (address of) error_status, control_flag, edge_type, color_index (address of) edge_width |
| return_buffer_length | 20 bytes |
| record_size | 20 bytes |

Escape functions

Workstation State List Inquiry Escape Functions

The following list describes the integer array contents:

| Component | Description |
|--------------|--|
| wsid | This element is the workstation identifier. |
| bundle_index | This element is the edge bundle index. |
| set_realized | This element determines the return type of the values. It can be either GKS\$K_VALUE_SET or GKS\$K_VALUE_REALIZED. |
| error_status | This element is the inquire error status. |
| control_flag | This element is the edge control. |
| edge_type | This element is the edge type. |
| color_index | This element is the edge color index. |

The following list describes the float array contents:

| Component | Description |
|------------|--|
| edge_width | This element is the edge width scale factor. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |

Escape functions

Workstation State List Inquiry Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |
| 36 | GKS\$_ERROR_36 | Specified workstation is Workstation Independent Segment Storage in routine **** |
| 92 | GKS\$_ERROR_92 | Color index is less than zero in routine **** |
| -90 | DECGKS\$_ERROR_NEG_92 | Internal GKS error: Bad memory address freed in routine **** |
| -154 | DECGKS\$_ERROR_NEG_154 | Length of initial string is greater than the buffer size in routine **** |
| -162 | DECGKS\$_ERROR_NEG_162 | Edge index is invalid in routine **** |

-404 Inquire Extent of a GDP

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_GDP_EXTENT

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the coordinate range, representing the GDP extent rectangle, to its output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|----------------------|
| function_id | -404 |
| in_data | (4 components) |
| | 7 |
| | 0 |
| | 0 |
| | (address of) in_data |

Escape functions

Workstation State List Inquiry Escape Functions

| Argument | Required Value |
|----------------------|--|
| in_data_size | 16 bytes |
| out_buffer | (5 components) 1 4 0 (address of) error_status (address of) coord_range |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

The real array contains the elements *coord_range*, which are the four world coordinate values of the segment's extent rectangle, using the current transformation values.

The following list describes the integer array contents of *in_data*:

| Component | Description |
|-------------|---|
| ws_id | This element is the workstation identifier. |
| num_points | This element is the number of points that define the GDP. |
| x_points | This element is the address of the array containing the GDP X point values. |
| y_points | This element is the address of the array containing the GDP Y point values. |
| GDP_id | This element is the GDP identifier. |
| d_r_size | This element is the size of the GDP data record in bytes. |
| d_r_address | This element is the address of the GDP data record. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |

Escape functions

Workstation State List Inquiry Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|---|
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |

Escape functions

Workstation Description Table Inquiry Escape Functions

Workstation Description Table Inquiry Escape Functions

This section describes all the escape functions that inquire about information in the workstation description table. All the inquiry functions in this section write an integer value called *error_status* to the output data record. If *error_status* is the value 0, then the rest of the output data record is valid. If *error_status* is not the value 0, then the rest of the output data record is invalid.

The escapes in this section require a workstation type (*ws_type*) instead of a workstation identifier (*ws_id*). For more information concerning the workstation type value, refer to Chapter 3, Control Functions.

For more information concerning the workstation description table, refer to Chapter 3, Control Functions. For more information concerning the DEC GKS inquiry functions and the *error_status* argument, refer to Chapter 11, Inquiry Functions.

-350 Inquire List of Available Escapes

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_LIST_ESC

Supporting workstations: All DEC GKS-supported workstations.

This escape returns the list of escapes supported by a specified workstation.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|---|
| function_id | -350 |
| in_data | (4 components) 1 0 0 (address of) ws_type |

Escape functions

Workstation Description Table Inquiry Escape Functions

| Argument | Required Value |
|----------------------|--|
| in_data_size | 16 bytes |
| out_buffer | (4 components) 3 + num_escapes 0 0 (address of) error_status, total_escapes, returned_escapes, escape_list |
| record_buffer_length | 16 bytes |
| record_size | 16 bytes |

The following list describes the integer array contents:

| Component | Description |
|------------------|---|
| error_status | This element is the inquiry error status. |
| total_escapes | This element is the total number of escapes supported by the workstation type. |
| returned_escapes | This element is the number of escape identifiers written to the remaining elements of the output data record's integer array. |
| escape_list | These elements are the identifiers of the supported escapes. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |

-351 Inquire Default Display Speed

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_DEF_SPEED

Supporting workstations: All DEC GKS-supported workstations.

Escape functions

Workstation Description Table Inquiry Escape Functions

This escape writes the default speed, for the specified workstation type, to its output data record. (See the Set Display Speed escape description in this chapter.)

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -351 |
| in_data | (4 components) 1 0 0 (address of) ws_type |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 1 1 0 (address of) error_status (address of) def_speed |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

Escape functions

Workstation Description Table Inquiry Escape Functions

-352 Inquire Line Cap and Join Facilities

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_LINE_CAP_JOIN_FAC

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the line cap and line join facilities, for the specified workstation type, to the output data record. (See the Set Line Cap Style and Set Line Join Style escapes in this chapter.)

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -352 |
| in_data | (4 components) 1 0 0 (address of) ws_type |
| in_data_size | 16 bytes |
| out_buffer | (4 components) 5 + ret_cap_styles + ret_join_styles 0 0 (address of) error_status, cap_join_data |
| record_buffer_length | 16 |
| record_size | 16 bytes |

The following list describes the integer array contents of *cap_join_data*:

| Component | Description |
|----------------|--|
| num_cap_styles | This element is the total number of line cap styles supported by the workstation type. |
| ret_cap_styles | This element is the number of cap styles written to the elements <i>cap_style_list</i> . |

Escape functions

Workstation Description Table Inquiry Escape Functions

| Component | Description |
|-----------------|--|
| num_join_styles | This element is the total number of line join styles supported by the workstation type. |
| ret_join_styles | This element is the number of join styles written to the elements <i>join_style_list</i> . |
| cap_list | These elements are the list of supported cap styles. |
| join_list | These elements are the list of supported join styles. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|---|
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |

-354 Inquire Edge Facilities

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$_K_ESC_INQ_EDGE_FAC

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the edge facilities available, for the specified workstation type, to the output data record.

Escape functions

Workstation Description Table Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -354 |
| in_data | (4 components) 1 0 0 (address of) ws_type |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 5 + number_edge_types 3 0 (address of) error_status, total_edge_types, number_edge_types, number_edge_widths, number_indices (address of) nominal_edge_width, minimum_edge_width, maximum_edge_width |
| return_buffer_length | 20 bytes |
| record_size | 20 bytes |

The following list describes the integer array contents:

| Component | Description |
|--------------------|---|
| ws_type | This element is the workstation type. |
| error_status | This element is the inquire error status. |
| total_edge_types | This element is the total number of edge types available. |
| number_edge_types | This element is the number of edge types returned. |
| number_edge_widths | This element is the number of edge widths available. |
| number_indices | This element is the number of predefined edge bundle table entries. |

Escape functions

Workstation Description Table Inquiry Escape Functions

The following list describes the float array contents:

| Component | Description |
|--------------------|---|
| nominal_edge_width | This element is the nominal edge width. |
| minimum_edge_width | This element is the minimum edge width. |
| maximum_edge_width | This element is the maximum edge width. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_22 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| -160 | DECGKS\$_ERROR_NEG_160 | Insufficient space in escape output data record arrays in routine **** |

-355 Inquire Predefined Edge Representation

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_INQ_PREDEF_EDGE_REP

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the predefined edge bundle information, for the specified workstation type, to the output data record.

Escape functions

Workstation Description Table Inquiry Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|---|
| function_id | -355 |
| in_data | (4 components) 2 0 0 (address of) ws_type, bundle_index |
| in_data_size | 16 bytes |
| out_buffer | (5 components) 4 1 0 (address of) error_status, control_flag, edge_type, color_index (address of) edge_width |
| return_buffer_length | 20 bytes |
| record_size | 20 bytes |

The following list describes the integer array contents:

| Component | Description |
|--------------|---|
| ws_type | This element is the workstation type. |
| bundle_index | This element is the edge bundle index. |
| error_status | This element is the inquire error status. |
| control_flag | This element is the edge control. |
| edge_type | This element is the edge type. |
| color_index | This element is the edge color index. |

The following list describes the float array contents:

| Component | Description |
|------------|--|
| edge_width | This element is the edge width scale factor. |

Escape functions

Workstation Description Table Inquiry Escape Functions

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| -150 | DECGKS\$_ERROR_NEG_150 | Edge index is less than zero in routine **** |
| -160 | DECGKS\$_ERROR_NEG_160 | Insufficient space in escape output data record arrays in routine **** |

-356 Inquire Maximum Number of Edge Bundles

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$_K_ESC_INQ_MAX_EDGE_BUNDLE

Supporting workstations: All DEC GKS-supported workstations.

This escape writes the maximum number of edge bundle table entries, for the specified workstation type, to the output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|-------------|---|
| function_id | -356 |
| in_data | (4 components) 1 0 0 (address of) ws_type |

Escape functions

Workstation Description Table Inquiry Escape Functions

| Argument | Required Value |
|----------------------|--|
| in_data_size | 16 bytes |
| out_buffer | (4 components) 2 0 0 (address of) error_status, max_number_bundles |
| return_buffer_length | 16 bytes |
| record_size | 16 bytes |

The following list describes the integer array contents:

| Component | Description |
|--------------------|--|
| ws_type | This element is the workstation type. |
| error_status | This element is the inquire error status. |
| max_number_bundles | This element is the maximum number of edge bundle table entries. |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 8 | GKS\$_ERROR_8 | GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP in routine **** |
| 22 | GKS\$_ERROR_22 | Specified workstation type is invalid in routine **** |
| 23 | GKS\$_ERROR_23 | Specified workstation type does not exist in routine **** |
| 39 | GKS\$_ERROR_39 | Specified workstation is neither of category OUTPUT nor of category OUTIN in routine **** |
| -160 | DECGKS\$_ERROR_NEG_160 | Insufficient space in escape output data record arrays in routine **** |

Escape functions

Utility Escape Functions

Utility Escape Functions

This section describes all the escape functions that provide you with utilities to assist you in programming. For instance, many of the utility functions translate the mapping of a point from one of the DEC GKS coordinate planes to another. (For more information concerning transformations, refer to Chapter 6, Transformation Functions.)

-400 Evaluate NDC Mapping of a WC Point

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_MAP_NDC_OF_WC

Supporting workstations: All DEC GKS-supported workstations.

This escape accepts a world coordinate point and a normalization transformation number, and writes the corresponding normalized device coordinate (NDC) point value to the output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|---|
| function_id | -400 |
| in_data | (5 components) 1 2 0 (address of) norm_xform (address of) world_x_value, world_y_value |
| in_data_size | 20 bytes |

Escape functions Utility Escape Functions

| Argument | Required Value |
|----------------------|--|
| out_buffer | (5 components) 0 2 0 null address (address of) NDC_x_value, NDC_y_value |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 50 | GKS\$ _ERROR_50 | Transformation number is invalid in routine **** |

-401 Evaluate DC Mapping of an NDC Point

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_MAP_DC_OF_NDC

Supporting workstations: All DEC GKS-supported workstations.

This escape accepts a normalized device coordinate (NDC) point, calculates the corresponding device coordinate point using the current workstation transformation, and writes the device coordinate value to the output data record.

Escape functions

Utility Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -401 |
| in_data | (5 components) 1 2 0 (address of) ws_id (address of) NDC_x_value, NDC_y_value |
| in_data_size | 20 bytes |
| out_buffer | (5 components) 0 2 0 null address (address of) DC_x_value, DC_y_value |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |

Escape functions Utility Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 33 | GKS\$_ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$_ERROR_35 | Specified workstation is of category INPUT in routine **** |

–402 Evaluate WC Mapping of NDC Point

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$_K_ESC_MAP_WC_OF_NDC

Supporting workstations: All DEC GKS-supported workstations.

This escape accepts a normalized device coordinate (NDC) point and a normalization transformation number, calculates the corresponding world coordinate point, and writes the world coordinate value to the output data record.

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|--------------|---|
| function_id | –402 |
| in_data | (5 components) 1 2 0 (address of) norm_xform (address of) NDC_x_value, NDC_y_value |
| in_data_size | 20 bytes |

Escape functions

Utility Escape Functions

| Argument | Required Value |
|----------------------|--|
| out_buffer | (5 components) 0 2 0 null address (address of) world_x_value, world_y_value |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 50 | GKS\$_ERROR_50 | Transformation number is invalid in routine **** |

-403 Evaluate NDC Mapping of DC Point

Operating states: GKOP, WSOP, WSAC, SGOP

Constant: GKS\$K_ESC_MAP_NDC_OF_DC

Supporting workstations: All DEC GKS-supported workstations.

This escape accepts a device coordinate point, calculates the corresponding normalized device coordinate (NDC) point using the current workstation transformation, and writes the device coordinate value to the output data record.

Escape functions Utility Escape Functions

GKS\$ESCAPE Arguments:

| Argument | Required Value |
|----------------------|--|
| function_id | -403 |
| in_data | (5 components) 1 2 0 (address of) ws_id (address of) DC_x_value, DC_y_value |
| in_data_size | 20 bytes |
| out_buffer | (5 components) 0 2 0 null address (address of) NDC_x_value, NDC_y_value |
| record_buffer_length | 20 bytes |
| record_size | 20 bytes |

Error Messages:

| Error Number | Completion Status Code | Message/Meaning |
|--------------|------------------------|--|
| 7 | GKS\$_ERROR_7 | GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP in routine **** |
| 20 | GKS\$_ERROR_20 | Specified workstation identifier is invalid in routine **** |
| 25 | GKS\$_ERROR_25 | Specified workstation is not open in routine **** |

Escape functions

Utility Escape Functions

| Error Number | Completion Status Code | Message/Meaning |
|---------------------|-------------------------------|--|
| 33 | GKS\$ _ERROR_33 | Specified workstation is of category MI in routine **** |
| 35 | GKS\$ _ERROR_35 | Specified workstation is of category INPUT in routine **** |

Appendix J

DEC GKS-Specific Input Values

This appendix provides input information that is applicable to all the DEC GKS GKS\$K_WSCAT_OUTIN workstations. You should review this appendix before working with the DEC GKS input functions. If you need further workstation-specific input information, refer to the device-specific appendixes in this manual.

This appendix describes the following input values that are available for all DEC GKS-supported devices:

- Input devices
- Prompt and echo types
- Data records
- Keypad functionality

Logical Input Device Numbers

Logical Input Device Numbers

The following section specifies which DEC GKS-supported workstations implement which logical input devices. Logical input device numbers determine the physical device (such as a keypad or a mouse) used to control the DEC GKS logical input devices. You pass the device numbers described in this section to the DEC GKS input functions, as follows:

```
      .  
      .  
C    Declare the device number...  
      INTEGER DEVICE_NUM  
  
      DATA DEVICE_NUM / 3 /  
      .  
      .  
C    Request input from the device...  
      CALL GKS$REQUEST_CHOICE( WS_ID, DEVICE_NUM, INPUT_STATUS,  
* INPUT_CHOICE )  
      .  
      .
```

DEC GKS defines at least four logical input device numbers for each input class (some workstations support eight choice devices). If the workstation does not support the device number you specify, the workstation implements the device in the same manner as it implements device number 1.

Several of the input devices use special sections of the keyboard available to users of specific workstations. If you use these devices, you should remember that you need to provide the user with the information necessary to operate them. For further information concerning input keypad functionality, refer to the Keypad Functionality section in this appendix.

Logical Input Device Numbers

To allow you to use several logical input devices of the same class during sample or event mode, DEC GKS defines different echo areas for devices of a single class. The appropriate chapters in the DEC GKS Device Specifics Reference Manual list the default echo area for the default logical input device of a given class. To determine the default echo area for other devices of the same class, call one of the `INQ_DEF_class_DATA` inquiry functions and pass it the appropriate device number. For more information, refer to Chapter 11, Inquiry Functions.

For complete information concerning logical input devices, physical input devices, and the DEC GKS input process, refer to Chapter 7, Input Functions.

Logical Input Devices

Choice Devices

Choice Devices

The following sections describe the choice logical input devices and specify which DEC GKS workstations support each device.

VAXstations: Using one of the VAXstations, you can use the LOCK Key feature for any of the choice logical input devices. For more information, refer to the Keypad Functionality section in this appendix.

Choice 1

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

For workstations that do not have a mouse or puck, this device requires that the user press the arrow keys to highlight various choices. To trigger this device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

For workstations that do have a mouse or puck, this device requires that the user move the tracking device to highlight various choices. To trigger this device, the user must press the left button. To cause a break during request mode, the user must press the middle button on the mouse and the top button on the puck.

Choice 2

Supporting Workstations: The VAXstation, VT125, VT240, VT330, VT340, and Tektronix—4107 workstations.

This device activates both the arrow keys and the numeric keypad keys to highlight the various choices. (For more information concerning the numeric keypad, refer to the Keypad Functionality section in this appendix.) By pressing any of the arrow or numeric keys, the user immediately triggers the device and the measure corresponds to the number assigned to the pressed key. To break input during request mode, the user must press CTRL/U.

Logical Input Devices

Choice Devices

NOTE

For all other DEC GKS workstations, the handlers accept this input device number, but they implement the device in the same manner as they implement choice device 1.

Choice 3

Supporting Workstations: The VAXstation, VT240, VT330, and VT340 workstations.

This device activates the top six keys of the auxiliary keypad and the keys F7 to F20 to highlight choices 1 through 20. (For more information concerning the auxiliary keypad or the keys F7 through F20, refer to the Keypad Functionality section.) By pressing any of the arrow or numeric keys, the user immediately triggers the device and the measure corresponds to the number assigned to the pressed key. To break input during request mode, the user must press CTRL/U.

NOTE

For all other DEC GKS workstations, the handlers accept this input device number, but they implement the device in the same manner as they implement choice device 1.

Choice 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

This device is implemented in the same manner as choice device number 1.

VAXstations/VT330/VT340 (with mouse): This device can display only up to four choices and does not react to the tracking device of the mouse or puck. (If you use a mouse, you should initialize the device for three choices; if you use a puck, you should initialize it for four choices.) The user triggers the device by depressing a mouse or puck button.

Logical Input Devices

Choice Devices

The measure is the choice number corresponding to the button pushed. The left button corresponds to choice 1; the middle button corresponds to choice 2; the right button corresponds to choice 3. If you use a puck, the bottom button corresponds to choice 4.

Choice 5

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

This device is implemented in the same manner as choice device number 1.

VAXstations/VT330/VT340 (with mouse): This device can display only up to four choices and does not react to the tracking device of the mouse or puck. (If you use a mouse, you should initialize the device for three choices; if you use a puck, you should initialize it for four choices.) The user triggers the device by releasing a mouse or puck button.

The measure is the choice number corresponding to the button pushed. The left button corresponds to choice 1; the middle button corresponds to choice 2; the right button corresponds to choice 3. If you use a puck, the bottom button corresponds to choice 4.

Choice 6, 7, and 8

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

These devices are implemented in the same manner as choice device number 1.

Locator Devices

The following subsection describes the locator logical input devices and specifies which DEC GKS workstations support each device.

VAXstations: Using one of the VAXstations, you can use the LOCK Key feature for any of the locator logical input devices. For more information, refer to the Keypad Functionality section in this appendix.

Locator 1, 2, 3, and 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

For workstations that do not have a mouse or puck, these devices require that the user press the arrow keys to move the locator prompt. To trigger the device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

For workstations that do have a mouse or puck, these devices require that the user move the tracking device to move the locator prompt. To trigger the device, the user must press the left button. To cause a break during request mode, the user must press the middle button on the mouse and the top button on the puck.

VT125/240/330/340 and Tektronix—4107: Using these workstations, you can use the numeric keypad as a zoning mechanism using device numbers 1 and 2. (For more information concerning the numeric keypad, refer to the Keypad Functionality section in this appendix.)

Logical Input Devices

Pick Devices

Pick Devices

The following subsection describes the pick logical input devices and specifies which DEC GKS workstations support each device.

Pick 1, 2, 3, and 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

For workstations that do not have a mouse or puck, these devices require that the user press the arrow keys to move the pick aperture. The workstation marks the currently picked segments (or portions of segments) by outlining the extent rectangle of all or part of the segment. To trigger the device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

For workstations that do have a mouse or puck, these devices require that the user move the tracking device to move the pick aperture. To trigger the device, the user must press the left button. To cause a break during request mode, the user must press the middle button on the mouse and the top button on the puck.

VT125/240 and Tektronix—4107: Using a VT240 or a VT125, you can use the numeric keypad as a zoning mechanism using device numbers 1 and 2. (For more information concerning the numeric keypad, refer to the Keypad Functionality section in this appendix.)

String Devices

The following sections describe the string logical input devices and specify which DEC GKS workstations support each device.

String 1 and 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

This device returns a DEC multinational text string to the calling program. The device requires the user to enter the text string using the keyboard. To trigger this device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

To edit the string while entering input (on all workstations except the Tektronix—4014), the user can use the following keys:

- DELETE, to delete the last character of the input string.
- CTRL/H, to move the cursor to the beginning of the string.
- CTRL/E, to move the cursor to the end of the string.
- CTRL/B, to recall only the initial string.
- CTRL/A, to toggle insert and overstrike modes.
- Left arrow, to move the cursor to the left.
- Right arrow, to move the cursor to the right.

String 2

Supporting Workstations: The VAXstation workstations.

This device returns an SMG Encoded Key value. DEC GKS ignores any prompt and echo type specified for this device. By pressing a key, you trigger the device; the measure of the device is the single character. For information concerning this type of text string, refer to the *VMS Run-Time Library Routines Reference Manual*.

Logical Input Devices

String Devices

NOTE

For all other DEC GKS-supported devices, the handlers accept this device number, but they implement the device in the same manner as they implement string devices 1 and 4.

String 3

Supporting Workstations: The VT240, VT125, and Tektronix—4107 workstations.

This device returns the ASCII value associated with the specified character. This device requires that the user press a single key on the keyboard. When the user presses a key, the device accepts the keystroke without a trigger. To cause a break during request mode, the user must press CTRL/U. DEC GKS ignores any prompt and echo type specified for this device.

NOTE

For all other DEC GKS-supported devices, the handlers accept this device number, but they implement the device in the same manner as they implement string devices 1 and 4.

Stroke Devices

The following subsection describes the stroke logical input devices and specifies which DEC GKS workstations support each device.

VAXstations: Using one of the VAXstations, you can use the LOCK Key feature for any of the stroke logical input devices. For more information, refer to the Keypad Functionality section in this appendix.

Stroke 1, 2, 3, and 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

For workstations that do not have a mouse or puck, these devices require that the user press the arrow keys to move the stroke prompt. To trigger the device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

For workstations that do have a mouse or puck, these devices require that the user move the tracking device to move the stroke prompt. To trigger the device, the user must press the left button. To cause a break during request mode, the user must press the middle button on the mouse and the top button on the puck.

VT125/240/330/340 and Tektronix—4107: Using these workstations, you can use the numeric keypad as a zoning mechanism when using device numbers 1 and 2. (For more information concerning the numeric keypad, refer to the Keypad Functionality section in this appendix.)

VT330/340: Using these workstations, you use the right mouse button to trigger a point in the stroke.

Logical Input Devices

Valuator Devices

Valuator Devices

The following subsection describes the valuator logical input devices and specifies which DEC GKS workstations support each device.

VAXstations: Using one of the VAXstations, you can use the LOCK Key feature for any of the valuator logical input devices. For more information, refer to the Keypad Functionality section in this appendix.

Valuator 1, 2, 3, and 4

Supporting Workstations: All DEC GKS-supported GKS\$K_WSCAT_OUTIN workstations.

For workstations that do not have a mouse or puck, these devices require that the user press the arrow keys to move the valuator prompt. To trigger the device, the user must press the RETURN key. To cause a break during request mode, the user must press CTRL/U.

For workstations that do have a mouse or puck, these devices require that the user move the tracking device to move the valuator prompt. To trigger the device, the user must press the left button. To cause a break during request mode, the user must press the middle button on the mouse and the top button on the puck.

Input Devices and Echo Area Titles

For all choice, string, and valuator devices, and for locator devices using prompt and echo type 6, you can specify a character string that the workstation places at the top of the echo area. In this manner, you can place an application-specific title at the top of the echo area.

To take advantage of this feature, allow for two extra longwords at the end of your input data record. For instance, if you use a string device with a prompt and echo type of 1, you normally declare the data record as follows:

```
      .  
      .  
C      String data record.  
      INTEGER DATA_RECORD( 2 )  
  
C      Enter the buffer size and cursor position...  
      DATA_RECORD( 1 ) = 30  
      DATA_RECORD( 2 ) = 0  
  
C      Specify the size of the data record...  
      RECORD_BUFFER_LENGTH = 8  
  
      CALL GKSS$INIT_STRING( WS_ID, DEVICE_NUM, ' ',  
* PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,  
* RECORD_BUFFER_LENGTH )  
      .  
      .  
      .
```

If you want to place a title at the top of the string echo area, you can declare the data record as follows:

```
      .  
      .  
      .  
C      String data record.  
      INTEGER DATA_RECORD( 4 )  
  
C      Enter the buffer size and cursor position...  
      DATA_RECORD( 1 ) = 30  
      DATA_RECORD( 2 ) = 0  
  
C      In the last two longwords, enter the address and length of  
C      the string to be used as a title for the echo area...  
      DATA_RECORD( 3 ) = %LOC( 'Enter Your Name' )  
      DATA_RECORD( 4 ) = LEN( 'Enter Your Name' )  
  
C      Specify the NEW size of the data record...  
      RECORD_BUFFER_LENGTH = 16
```

Logical Input Devices

Input Devices and Echo Area Titles

```
CALL GKS$INIT_STRING( WS_ID, DEVICE_NUM, ' ',  
* PROMPT_ECHO_TYPE, ECHO_AREA, DATA_RECORD,  
* RECORD_BUFFER_LENGTH )
```

VAXstations: If you do not pass the extra components of the data record, DEC GKS always places a banner at the top of the input window; in this case, you cannot eliminate the banner. If you want to create an input window that does not contain a banner, pass a title length of 0 to the first of the extra components of the input data record. For more information concerning the VAXstation window banners and borders, refer to the Programming Consideration section in Chapter 1, VAXstation Workstation Specifics, in the *DEC GKS Device Specifics Reference Manual*.

Prompt and Echo Types, and Data Records

The following sections describe the DEC GKS-supported prompt and echo types for each class of logical input device. After describing the available prompt and echo types, these sections describe the DEC GKS required input data records for each prompt and echo type. These data records are for GKS\$ functions *only*. See the FORTRAN and C Bindings for information about FORTRAN and C data records.

Prompt and Echo Types, and Data Records

Choice Input Class

Choice Input Class

The choice class input devices support the following equivalent prompt and echo types:

| Echo Type Number | Description |
|-------------------------|--|
| -1 | Highlight the current choice using a hollow rectangle. |
| 1 | Display the list of choice strings within the echo area. |
| 3 | Display the list of choice strings within the echo area. |

Choice Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

For more information about specifying a character string at the top of the workstation echo area, see *Input Devices and Echo Area Titles*.

Prompt and Echo Types, and Data Records Choice Input Class

Choice Class: All Prompt and Echo Types

If you specify either of these prompt and echo types, the workstations expect a data record of size 12. If you call INITIALIZE CHOICE, the *record_buffer_length* argument must be the value 12.

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 1 | Integer | U | Number of choice strings. |
| 2 | Integer | U | Address of array containing choice string lengths. |
| 3 | Integer | U | Address of array containing addresses of choice string lengths. |

Prompt and Echo Types, and Data Records

Locator Input Class

Locator Input Class

The locator class input devices support the following prompt and echo types:

| Echo Type Number | Description |
|-----------------------------|--|
| -12 | Mark the current location using an ellipse centered at the initial point and the current location at the corner of the bounding rectangle. |
| -11 | Mark the current location with the world coordinate translation of the device coordinate position. |
| -10 | Mark the current location using a circle centered at the midpoint of the initial position and the current location. |
| -9 | Mark the current location using a circle centered at the initial position, with the current location on the circumference. |
| -8 | Mark the current location using an open type arc defined by the current location and two points supplied in the data record. |
| -7 | Mark the current location using a pie type arc defined by the current location and two points supplied in the data record. |
| -6 | Mark the current location using a chord type arc defined by the current location and two points supplied in the data record. |
| -5 | Mark the current location using a horizontal line drawn from the initial position to the current location. |
| -4 | Mark the current location using a vertical line drawn from the initial position to the current location. |
| -3 | Mark the current location using two lines connected to two fixed points supplied in the data record. |
| -2 | Mark the current location using a rectangle that is centered at the initial points and has a corner at the current location. |
| -1 | Mark the current location with a marker shaped like a box. |
| 1 | Mark the current location with a tracking plus sign. |
| 2 | Mark the current location by using a vertical and a horizontal line as a crosshair. |
| 3 | Mark the current location using a tracking cross. |

Prompt and Echo Types, and Data Records Locator Input Class

| Echo Type Number | Description |
|------------------|--|
| 4 | Mark the current location using a line connecting the current location to the initial location (rubber-band line). |
| 5 | Mark the current location using a rectangle whose diagonal is the current location and the initial location (rubber-band box). |
| 6 | Mark the current location by displaying a digital representation of the location. |

Locator Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

For more information about specifying a character string at the top of the workstation echo area for locator devices using prompt and echo type 6, see Input Devices and Echo Area Titles.

Locator Class: Prompt and Echo Types -1

If you specify this prompt and echo type, the workstations expect a data record of size 8 bytes. If you call INITIALIZE LOCATOR, the *record_buffer_length* argument must be the value 8.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Real | U | X dimension of the box in world coordinates. |
| 2 | Real | U | Y dimension of the box in world coordinates. |

Prompt and Echo Types, and Data Records Locator Input Class

NOTE

Since you express the X and Y dimensions of the box, the current normalization transformation affects the size and shape of this cursor. DEC GKS centers this box around the initial position.

Locator Class: Prompt and Echo Types 1, 2, 3, 6, and -11

If you specify any of these prompt and echo types, the workstations expect a null data record of size 0 bytes. If you call INITIALIZE LOCATOR, the *record_buffer_length* argument must be the value 0.

Locator Class: Prompt and Echo Type 4, -12, -10, -9, -5, and -4

If you specify this prompt and echo type, the workstations expect a data record of size 4 or 32 bytes, depending on the value of the attribute control flag. If you call INITIALIZE LOCATOR, the *record_buffer_length* argument must be the value 4 or 32.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Integer | U | Attribute control flag. GKS\$K_ACF_CURRENT (0) or GKS\$K_ACF_SPECIFIED (1). Use the currently set output attributes or specify new attributes in this data record. |

If component 1 is GKS\$K_ACF_SPECIFIED, you must pass the following components:

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 2 | Integer | I | Line type aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |

Prompt and Echo Types, and Data Records Locator Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 3 | Integer | I | Line width scale factor aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 4 | Integer | I | Polyline color index aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 5 | Integer | I | Polyline index. |
| 6 | Integer | U | Line type index. |
| 7 | Real | U | Line width scale factor. |
| 5 | Integer | I | Polyline color index. |

Locator Class: Prompt and Echo Type 5 and -2

If you specify either of these prompt and echo types, the workstations expect a data record of size 8 or 36 bytes, depending on the value of the attribute control flag. If you call INITIALIZE LOCATOR, the *record_buffer_length* argument must be the value 8 or 36.

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 1 | Integer | I | Polyline/fill area control flag. GKS\$K_ACF_POLYLINE (0) or GKS\$K_ACF_FILL_AREA (1). Use a polyline or a filled area to draw the rectangle whose diagonal connects the current and initial points. |
| 2 | Integer | U | Attribute control flag. GKS\$K_ACF_CURRENT (0) or GKS\$K_ACF_SPECIFIED (1). Use the currently set output attributes or specify new attributes in this data record. |

If component 1 is GKS\$K_ACF_POLYLINE and component 2 is GKS\$K_ACF_SPECIFIED, then you must pass the following data record components:

Prompt and Echo Types, and Data Records Locator Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 3 | Integer | I | Line type aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 4 | Integer | I | Line width scale factor aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 5 | Integer | I | Polyline color index aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 6 | Integer | I | Polyline index. |
| 7 | Integer | U | Line type index. |
| 8 | Real | U | Line width scale factor. |
| 9 | Integer | I | Polyline color index. |

If component 1 is GKS\$K_ACF_FILL_AREA and component 2 is GKS\$K_ACF_SPECIFIED, then you must pass the following record components:

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 3 | Integer | I | Fill area interior style aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 4 | Integer | I | Fill area style index aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 5 | Integer | I | Fill area color index aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 6 | Integer | I | Fill area index. |

Prompt and Echo Types, and Data Records Locator Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 7 | Integer | I | Fill area interior style. GKS\$K_INTSTYLE_HOLLOW (0), GKS\$K_INTSTYLE_SOLID (1), GKS\$K_INTSTYLE_PATTERN (2), or GKS\$K_INTSTYLE_HATCH (3). |
| 8 | Integer | I | Fill area style index. |
| 9 | Integer | I | Fill area color index. |

Locator Class: Prompt and Echo Type -8, -7, -6, and -3

If you specify any of these prompt and echo types, the workstations expect a data record of size 20 or 48 bytes, depending on the value of the attribute control flag. If you call INITIALIZE LOCATOR, the *record_buffer_length* argument must be the value 20 or 48.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Integer | U | Attribute control flag. GKS\$K_ACF_CURRENT(0) or GKS\$K_ACF_SPECIFIED. Use the currently set output attributes or specify new attributes in the data record. |

If component 1 is GKS\$K_ACF_CURRENT, then you must pass the following data record components:

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 2 | Real | U | X component of the first world coordinate point. |
| 3 | Real | U | Y component of the first world coordinate point. |
| 4 | Real | U | X component of the second world coordinate point. |
| 5 | Real | U | Y component of the second world coordinate point. |

If component 1 is GKS\$K_ACF_SPECIFIED, then you must pass the following record components:

Prompt and Echo Types, and Data Records Locator Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 2 | Integer | I | Line type aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 3 | Integer | I | Line width scale factor aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 4 | Integer | I | Polyline color aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 5 | Integer | I | Polyline bundle index. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 6 | Integer | U | Line type index. |
| 7 | Real | U | Line width scale factor. |
| 8 | Integer | I | Polyline color index. |
| 9 | Real | U | X component of the first world coordinate point. |
| 10 | Real | U | Y component of the first world coordinate point. |
| 11 | Real | U | X component of the second world coordinate point. |
| 12 | Real | U | Y component of the second world coordinate point. |

Additionally, the data record fields described as `line_type_index` and `line_width_scale_factor` are now used by some workstations, primarily VAXstations running UIS.

Prompt and Echo Types, and Data Records

Pick Input Class

Pick Input Class

The pick logical input devices support the following prompt and echo types:

| Echo Type Number | Description |
|------------------|---|
| 1 | Highlight the extent rectangle of the picked output primitive. |
| 2 | Highlight the extent rectangle of all the output primitives that share the pick identifier of the picked primitive. |
| 3 | Highlight the extent rectangle of the picked segment. |

Pick Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

Pick Class: All Prompt and Echo Types

If you specify any of these prompt and echo types, the workstations expect a data record of size 4. If you call INITIALIZE PICK, the *record_buffer_length* argument must be the value 4.

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 1 | Real | U | Size of the pick aperture (prompt) in device coordinates. |

Prompt and Echo Types, and Data Records

String Input Class

String Input Class

The string logical input devices support the following prompt and echo type:

| Echo Type Number | Description |
|------------------|--|
| 1 | Display the current string value in the echo area. |

String Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

For more information about specifying a character string at the top of the workstation echo area, see Input Devices and Echo Area Titles.

String Class: Prompt and Echo Type 1

If you specify this prompt and echo type, the workstations expect a data record of size 8 bytes, depending on the value of the attribute control flag. If you call INITIALIZE STRING, the *record_buffer_length* argument must be the value 8.

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 1 | Integer | U | Input buffer size in number of characters. |
| 2 | Integer | I | Initial cursor position within the string. The initial position must follow the formula: $1 \leq \text{initial_position} \leq \text{length_initial_string}$ |

Prompt and Echo Types, and Data Records

Stroke Input Class

Stroke Input Class

The stroke class input devices support the following equivalent prompt and echo type values:

| Echo Type Number | Description |
|------------------|---|
| 1 | Display a line joining successive points of the current stroke. |
| 3 | Display a polymarker at each successive stroke point. |
| 4 | Display a line joining successive points of the current stroke. |

Stroke Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

Stroke Class: Prompt and Echo Type 1

If you specify this prompt and echo type, the workstations expect a data record of size 20 bytes. If you call INITIALIZE STROKE, the *record_buffer_length* argument must be the value 20.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Integer | U | Input buffer size, in number of stroke points. |
| 2 | Integer | I | Editing position expressed as a stroke point. |
| 3 | Real | U | X world coordinate change vector. |

Prompt and Echo Types, and Data Records

Stroke Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|-----------------------------------|
| 4 | Real | U | Y world coordinate change vector. |
| 5 | Real | I | Time interval, in seconds. |

Stroke Class: Prompt and Echo Type 3

If you specify this prompt and echo type, the workstations expect a data record of size 24 or 52 bytes, depending on the value of the attribute control flag. If you call INITIALIZE STROKE, the *record_buffer_length* argument must be the value 24 or 52.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Integer | U | Input buffer size, in number of stroke points. |
| 2 | Integer | I | Editing position expressed as a stroke point. |
| 3 | Real | U | X world coordinate change vector. |
| 4 | Real | U | Y world coordinate change vector. |
| 5 | Real | I | Time interval, in seconds. |
| 6 | Integer | U | Attribute control flag. GKS\$K_ACF_CURRENT (0) or GKS\$K_ACF_SPECIFIED (1). Use the currently set output attributes or specify new attributes in this data record. |

If component 6 is GKS\$K_ACF_SPECIFIED, you must pass the following components:

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 7 | Integer | I | Polymarker type aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 8 | Integer | I | Polymarker size factor aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |

Prompt and Echo Types, and Data Records Stroke Input Class

| Position | Data Type | Used | Description |
|----------|-----------|------|---|
| 9 | Integer | I | Polymarker color aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 10 | Integer | I | Polymarker bundle index. |
| 11 | Integer | U | Polymarker type index. |
| 12 | Real | U | Polymarker scale factor. |
| 13 | Integer | I | Polymarker color index. |

Stroke Class: Prompt and Echo Type 4

If you specify this prompt and echo type, the workstations expect a data record of size 24 or 52 bytes, depending on the value of the attribute control flag. If you call INITIALIZE STROKE, the *record_buffer_length* argument must be the value 24 or 52.

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 1 | Integer | U | Input buffer size, in number of stroke points. |
| 2 | Integer | I | Editing position expressed as a stroke point. |
| 3 | Real | U | X world coordinate change vector. |
| 4 | Real | U | Y world coordinate change vector. |
| 5 | Real | I | Time interval, in seconds. |
| 6 | Integer | U | Attribute control flag. GKS\$K_ACF_CURRENT (0) or GKS\$K_ACF_SPECIFIED (1). Use the currently set output attributes or specify new attributes in this data record. |

Prompt and Echo Types, and Data Records Stroke Input Class

If component 6 is GKS\$K_ACF_SPECIFIED, you must pass the following components:

| Position | Data Type | Used | Description |
|----------|-----------|------|--|
| 7 | Integer | I | Line type aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 8 | Integer | I | Line width scale factor aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 9 | Integer | I | Polyline color index aspect source flag. GKS\$K_ASF_BUNDLED (0) or GKS\$K_ASF_INDIVIDUAL (1). |
| 10 | Integer | I | Polyline index. |
| 11 | Integer | U | Line type index. |
| 12 | Real | U | Line width scale factor. |
| 13 | Integer | I | Polyline color index. |

Prompt and Echo Types, and Data Records

Valuator Input Class

Valuator Input Class

The valuator class input devices support the following prompt and echo types:

| Echo Type Number | Description |
|-------------------------|--|
| -3 | Display the range of values in a circular dial (for use only with the VAXstations). |
| -2 | Display the range of values on a horizontal sliding scale (for use only with the VAXstations). |
| -1 | Display the range of values on a vertical sliding scale (for use only with the VAXstations). |
| 1 | Display a graphical representation of the current value (such as a dial or a pointer). |
| 2 | Display a graphical representation of the current value (such as a dial or a pointer). |
| 3 | Display a digital representation of the current value. |

Valuator Data Records

The DEC GKS workstations require the following data records for the specified prompt and echo types. The introduction at the beginning of each subsection specifies the data record size requirements. The column marked Used specifies whether the handler uses (U) or ignores (I) the data record component.

For more information about specifying a character string at the top of the workstation echo area, see Input Devices and Echo Area Titles.

Prompt and Echo Types, and Data Records

Valuator Input Class

Valuator Class: Prompt and Echo Types -1, -2, and -3

These prompt and echo types are only for use with the VAXstation workstations.

If you specify any of these prompt and echo types, the workstations expect a data record of size 8 bytes. If you call INITIALIZE VALUATOR, the *record_buffer_length* argument must be the value 8.

| Position | Data Type | Used | Description |
|----------|-----------|------|----------------------------------|
| 1 | Real | U | Low value of the numeric range. |
| 2 | Real | U | High value of the numeric range. |

Valuator Class: Prompt and Echo Types 1, 2, and 3

If you specify any of these prompt and echo types, the workstations expect a data record of size 8 bytes. If you call INITIALIZE VALUATOR, the *record_buffer_length* argument must be the value 8.

| Position | Data Type | Used | Description |
|----------|-----------|------|----------------------------------|
| 1 | Real | U | Low value of the numeric range. |
| 2 | Real | U | High value of the numeric range. |

Keypad Functionality

DEC GKS allows the user to press keys other than the arrow keys to control the input prompt. This section describes how the user can use the various keypad tablets during input. If you use logical input devices that take advantage of these keypads, remember to provide the user with the information necessary to operate the device.

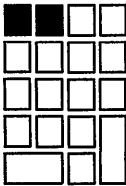
Keypad Functionality

Cycling Logical Input Devices

Cycling Logical Input Devices

Supporting Devices: All logical input devices used on a single workstation.

Supporting Workstations: The VT125, VT240, VT330, VT340, Tektronix—4014, and the Tektronix—4107 workstations.



The shaded key to the left is the PF1 key. This key cycles through the devices present on a single workstation, in a workstation-determined order. The shaded key to the right is the PF2 key. This key ends the cycling process and activates the prompts of all logical input devices present on a workstation. (If you are using the Tektronix—4107 terminal, these keys are labeled F5 and F6.)

When you use more than one logical input device at a time, the workstations change the measures of all devices that use a physical device, by default. For instance, if you simultaneously use two devices that use the arrow keys to alter the prompt, the user moves both prompts when pressing the arrow keys.

In order to provide the user with the ability to choose which device's measure to alter, the workstations allow the user to activate the prompts of each device individually, in a workstation-specific cycle. In this way, the user can change the measure of only one device at a time.

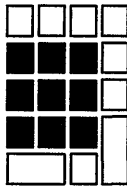
The only restriction placed on the cycling of logical input devices is that cycling only affects those devices whose prompts are enabled. If you use a device on a workstation whose prompt is disabled (by setting the value `GKS$K_NOECHO` in one of the `SET_class_MODE` functions), that device's prompt is always active. You cannot cycle past a device whose echo is disabled.

Keypad Functionality Numeric Keypad (Zoning Mechanism)

Numeric Keypad (Zoning Mechanism)

Supporting Devices: Locator, pick, and stroke device numbers 1 and 2.

Supporting Workstations: VT125, VT240, VT330, VT340, and the Tektronix—4107 workstations.



The workstations move the cursor to the position on the rectangular input echo area that corresponds to the position of the pressed key within the rectangular set of shaded keys. For instance, if the user presses the shaded key in the upper left corner, the cursor moves to the upper left corner of the current echo area. If the user presses the shaded key in the exact center, the cursor moves to the center of the echo area. If the user presses the rightmost shaded key in the second shaded row of keys, the cursor moves to the middle of the right border of the rectangular echo area.

Keypad Functionality

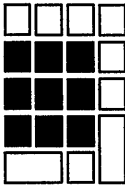
Numeric Keypad (Choice)

Numeric Keypad (Choice)

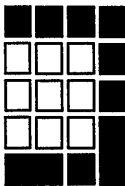
Supporting Devices: Choice device number 2.

Supporting Workstations: The VAXstation, VT125, VT240, VT330, VT340, and Tektronix—4107 workstations.

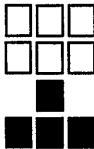
Key Set 1



Key Set 2



Key Set 3



The workstations trigger the choice that corresponds to the number assigned to the shaded keys. The number assignments are as follows:

Keypad Functionality Numeric Keypad (Choice)

| Key Set | Numbering Assignments |
|-----------|--|
| Key Set 1 | On most supporting workstations, the numbers on these shaded keys correspond to the choice numbers 1 through 9. Incrementing from left to right, the bottom row contains keys 1, 2, and 3; the middle row contains keys 4, 5, and 6; and, the top row contains keys 7, 8, and 9. |
| Key Set 2 | Beginning with the shaded key in the lower left corner, the corresponding choice numbers increment as you move clockwise around the key set. The shaded key in the lower left corner corresponds to choice number 10; the key in the upper left corner corresponds to choice number 11, the next key (moving clockwise) in the top row corresponds to choice number 12, and so forth. The middle key on the bottom row corresponds to choice number 18. |
| Key Set 3 | <p>These shaded keys are the arrow keys. The up arrow key corresponds to choice number 19; the down arrow key corresponds to choice number 20; the left arrow key corresponds to choice number 21; and, the right arrow key corresponds to choice number 22.</p> <p>Tektronix—4107: The keys F1 through F4 and the joydisk return valid choice numbers when using this device.</p> <p>VT125: The arrow keys are located in a row in the top right portion of the keyboard.</p> |

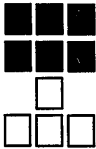
Keypad Functionality

Auxiliary Keypad (Choice)

Auxiliary Keypad (Choice)

Supporting Devices: Choice device number 3.

Supporting Workstations: The VAXstation, VT240, VT330, and VT340 workstations.



These keys operate in the same manner as the numeric keypad for choice input. The upper left shaded key is equivalent to choice prompt 1, the upper right to choice prompt 3, the lower left to choice prompt 4, and the lower right to choice prompt 6.

In addition, the keys located at the top of the keyboard labeled F7 through F20 correspond to the equivalent choice prompt. The workstation triggers the choice prompt of the number pressed by the user. You can use this keypad (choice device number 3 on the VT240) if you have up to 20 choices. If you have nine or less choices, you can use the numeric keypad, for choice device number 2, on either the VT125 or the VT240.

The LOCK Key

Supporting Devices: Choice, locator, stroke, and valuator.

Supporting Workstations: The VAXstation workstations.

When you use several logical input devices at one time, the measure of a device can change if the user moves the mouse's tracking cursor across the device. If the device is in sample mode and if the application happens to sample from that device as the user moves the tracking cursor across the device's echo area, inappropriate values may be returned to the application program.

DEC GKS allows the user to lock a logical input device so that its measure cannot be altered until the user unlocks the device. If a device is locked, the user can still trigger the device (if in request or event mode), but the measure cannot be altered by moving the tracking cursor across the device's echo area.

To lock a device, depress the LOCK key (this activates the red LOCK light at the top of the keyboard), move the cursor into the device's echo area, and press any mouse button. Once the device is locked, press the LOCK key again (the LOCK light turns off) and continue to enter input values in other devices. The locked device always returns the same measure.

To unlock the device, depress the LOCK key (activating the LOCK light), move the cursor into the locked device's echo area, and press any mouse button. Once the device is unlocked, press the LOCK key again (the LOCK light turns off), and you can now change the measure of the device.

A

- Addresses
 - GDP and escape data records, I-2
- Angles
 - GDPs, I-5
- ANSI
 - CGM standard, E-1
 - GKSM standard, E-1
- Applications
 - programming information, F-1
- Arcs
 - direction of formation, I-6
- Arguments
 - See also Inquiry functions
 - inquiry error status, 11-5
 - inquiry value type argument, 11-6
 - passing by descriptor, F-1
- Arrays
 - descriptors, F-1
- ASCII
 - VT125/240 string input, J-10
- Attributes
 - color chart, H-1
 - DEC GKS specific line types, C-5
 - initial values, C-1 to C-4
 - output
 - list of errors, D-26 to D-33
- Auxiliary keypad
 - choice input, J-38

B

- Base line, G-1
 - See also Fonts
- BASIC programming information, F-3

- Binding
 - list of GKS\$ constants, B-1 to B-21
- Bit masks
 - See also Workstations
 - constants, A-5
 - workstation types, A-4
- Bottom line, G-1
 - See also Fonts
- Bundles
 - See also Attributes

C

- Calling sequences, F-3
- Cap line, G-1
 - See also Fonts
- CGM
 - ANSI standard, E-1
 - CGM metafiles, E-9
- Character descriptor, G-9
 - elements, G-10
- Characters
 - fonts, G-1
- Choice
 - data records required, J-16
 - keypad functionality, J-36, J-38
 - logical input device numbers, J-4
 - prompt and echo types supported, J-16
- Circumference
 - See also GDPs
 - ellipses, I-6
- Clipping flag
 - initial value, C-5
- COBOL programming information, F-3 to F-7

Colors

- chart, H-1 to H-5
- reservation
 - VSII/GPX, A-4

Completion status codes, D-1

Components

- GDP and escape data records, I-2

Conditions

- error, D-1 to D-43

Connection identifiers

- file specifications, A-4
- hardcopy workstation types, A-4

Constants

- See also Workstations
- bit masks, A-5
- for supported workstations, A-1 to A-6
- GDPs and escapes, I-1
- GKS\$ binding, B-1
- list of, B-1 to B-21

Coordinates

- See also Escapes
- escapes, I-48
- range translation, I-132

C programming information, F-3

Cycling

- logical input devices, J-34

D

Data records

- See also Escapes
- See also GDPs
- GDPs and escapes, I-2 to I-3
- input
 - echo area titles, J-13
 - required, J-15 to J-32
- internal metafile structure, E-4

DEC GKS

- fonts, G-1 to G-35
- input values, J-1

Declaring

- GKS functions
 - externally, F-1

DECwindows

- workstation type constant, A-3

DECwindows—drawable

- workstation type constant, A-3

DECwindows—output only

- workstation type constant, A-3

Defaults

- GKS\$K_WSTYPE_DEFAULT, A-1, A-3

Definition files

- declaring external functions, F-1
- list of, B-1

Description tables

- GKS, 11-1
- workstation, 11-1

Descriptors

- passing arguments, F-1
- problems passing, F-3 to F-7

Device independent, 11-1

- fonts, G-1

Device numbers

- DEC GKS logical input, J-2

Devices

- See also Workstations
- constants, A-1 to A-6
- DEC GKS available logical input, J-2
- DEC GKS specific input values, J-1 to J-39
- hardcopy, A-4
- output-only workstation types, A-4

E

Echo

- cycling with disabled input echoing, J-34

Echo area

- titles, J-13

Echo types

- DEC GKS supported, J-15 to J-32

Ellipses

- focus points, I-6
- formation, I-6

Errors

- GDPs, I-4
- inquiry error status argument, 11-5
- messages, D-1 to D-43
 - escape functions, D-41
 - implementation-specific, D-2 to D-14
 - input, D-36 to D-39
 - metafiles, D-39 to D-41
 - miscellaneous, D-41
 - operating state, D-17 to D-19
 - output, D-33 to D-34
 - output attributes, D-26 to D-33
 - segments, D-34 to D-36
 - system, D-42 to D-43
 - transformations, D-25 to D-26
 - workstation, D-19 to D-25

Escape functions

- list of errors, D-41

Escapes, I-48 to I-138

Escapes (Cont.)

control, I-49 to I-61

GKS\$K_ESC_ASSOC_WSTYPE_CONID,
I-59
GKS\$K_ESC_BEEP, I-52
GKS\$K_ESC_POP_WORKSTATION, I-53
GKS\$K_ESC_PRINT, I-50
GKS\$K_ESC_PUSH_WORKSTATION, I-54
GKS\$K_ESC_SET_ERR_HANDLING_MODE,
I-56
GKS\$K_ESC_SET_SPEED, I-49
GKS\$K_ESC_SET_VIEWPORT_EVENT,
I-57
GKS\$K_SET_ICON_BITMAPS, I-88
GKS\$K_SET_SOFT_CLIP, I-60

coordinate ranges, I-48

coordinate range translation, I-132 to I-138

data records, I-2 to I-3

DECwindows, I-83 to I-100

GKS\$K_ESC_DOUBLE_BUFFER, I-96
GKS\$K_ESC_INQ_BCKGRND_PIXMAP, I-99
GKS\$K_ESC_INQ_DBUFFER_PIXMAP, I-98
GKS\$K_ESC_INQ_MENU_BAR_ID, I-93
GKS\$K_ESC_INQ_PASTEBOARD_ID, I-92
GKS\$K_ESC_INQ_SHELL_ID, I-95
GKS\$K_ESC_INQ_WINDOW_IDS, I-90
GKS\$K_ESC_SET_BCKGRND_PIXMAP,
I-97
GKS\$K_ESC_SET_CANCEL_STRING, I-86
GKS\$K_ESC_SET_ENTER_STRING, I-87
GKS\$K_ESC_SET_RESET_STRING, I-84
GKS\$K_ESC_SET_WINDOW_TITLE, I-83

GKS state list inquiries, I-101 to I-105

GKS\$K_ESC_INQ_EDGE_ATTR, I-104
GKS\$K_ESC_INQ_LINE_CAP, I-102
GKS\$K_ESC_INQ_LINE_JOIN, I-103
GKS\$K_ESC_INQ_WRITING_MODE, I-101

output related, I-62 to I-82

GKS\$K_ESC_BEGIN_TRANS_BLOCK, I-73
GKS\$K_ESC_END_TRANS_BLOCK, I-74
GKS\$K_ESC_SET_EDGE_ASF, I-71
GKS\$K_ESC_SET_EDGE_COLOR_INDEX,
I-69
GKS\$K_ESC_SET_EDGE_CTL, I-65
GKS\$K_ESC_SET_EDGE_INDEX, I-70
GKS\$K_ESC_SET_EDGE_REP, I-80
GKS\$K_ESC_SET_EDGE_TYPE, I-67
GKS\$K_ESC_SET_EDGE_WIDTH, I-68
GKS\$K_ESC_SET_HIGH_METHOD, I-78
GKS\$K_ESC_SET_LINE_CAP, I-63
GKS\$K_ESC_SET_LINE_JOIN, I-64

Escapes

output related (Cont.)

GKS\$K_ESC_SET_SEG_HIGH_METHOD,
I-75
GKS\$K_ESC_SET_WRITING_MODE, I-62
utility, I-132 to I-138
GKS\$K_ESC_MAP_DC_OF_NDC, I-133
GKS\$K_ESC_MAP_NDC_OF_DC, I-136
GKS\$K_ESC_MAP_NDC_OF_WC, I-132
GKS\$K_ESC_MAP_WC_OF_NDC, I-135
WS description table inquiries, I-122 to I-131
GKS\$K_ESC_INQ_DEF_SPEED, I-123
GKS\$K_ESC_INQ_EDGE_FAC, I-126
GKS\$K_ESC_INQ_LINE_CAP_JOIN_FAC,
I-125
GKS\$K_ESC_INQ_LIST_ESC, I-122
GKS\$K_ESC_INQ_MAX_EDGE_BUNDLE,
I-130
GKS\$K_ESC_INQ_PREDEF_EDGE_REP,
I-128
WS state list inquiries, I-106 to I-121
GEIVD, I-106
GKS\$K_ESC_INQ_EDGE_REP, I-117
GKS\$K_ESC_INQ_GDP_EXTENT, I-119
GKS\$K_ESC_INQ_LIST_EDGE_INDEXES,
I-109
GKS\$K_ESC_INQ_SEGMENT_EXTENT,
I-111
GKS\$K_ESC_INQ_SPEED, I-108
GKS\$K_ESC_INQ_VIEWPORT_DATA, I-106
GKS\$K_INQ_HIGH_METHOD, I-114
GKS\$K_INQ_LIST_HIGH, I-115
GKS\$K_INQ_SEG_HIGH_METHOD, I-113

External functions

declaring GKS functions, F-1

F

Fields

metafile structure, E-2

Files

connection identifiers, A-4
definition
list of, B-1

Fill areas

See also Attributes
initial attributes, C-4

Focus points, I-6

See also GDPs

Fonts

designing, G-2

Fonts (Cont.)

- GKS multinational, G-1
- lines, G-1
- list of, G-10 to G-35
- monospaced, G-1
- software, G-1
- stroke font file, G-4
- Stroke font file
 - character descriptor, G-9
 - header, G-5
- supported by DEC GKS, G-1 to G-35

Format

- font file, G-1
- metafiles, E-1

Functions

- arguments passed by descriptor, F-1
- declaring GKS functions, F-1
- inquiry, 11-1

G

GDPs, I-4 to I-47

- additional
 - GKS\$K_GDP_DISJOINT_PLINE, I-9
- angles, I-5
- arcs
 - direction of formation, I-6
 - GKS\$K_GDP_ARC_2PT_RAD, I-17
 - GKS\$K_GDP_ARC_3PT, I-15
 - GKS\$K_GDP_ARC_CTR_2PT, I-13
 - GKS\$K_GDP_ARC_CTR_2VEC_RAD, I-16
 - GKS\$K_GDP_ARC_CTR_PT_ANG, I-18
 - GKS\$K_GDP_FARC_2PT_RAD, I-35
 - GKS\$K_GDP_FARC_3PT, I-32
 - GKS\$K_GDP_FARC_CTR_2PT, I-31
 - GKS\$K_GDP_FARC_CTR_2VEC_RAD, I-34
 - GKS\$K_GDP_FARC_CTR_PT_ANG, I-36
- cell arrays, I-44 to I-45
 - GKS\$K_GDP_IMAGE_ARRAY, I-44
- circles
 - GKS\$K_GDP_CIRCLE_2PT_RAD, I-12
 - GKS\$K_GDP_CIRCLE_3PT, I-10
 - GKS\$K_GDP_CIRCLE_CTR_PT, I-10
 - GKS\$K_GDP_CIRCLE_CTR_RAD, I-11
 - GKS\$K_GDP_FCIRCLE_2PT_RAD, I-30
 - GKS\$K_GDP_FCIRCLE_3PT, I-28
 - GKS\$K_GDP_FCIRCLE_CTR_PT, I-27
 - GKS\$K_GDP_FCIRCLE_CTR_RAD, I-29
- data records, I-2 to I-3
- ellipses
 - formation, I-6

GDPs

ellipses (Cont.)

- GKS\$K_GDP_ELLIPSE_CTR_AXES, I-20
- GKS\$K_GDP_ELLIPSE_FOCII_PT, I-20
- GKS\$K_GDP_FELLIPSE_CTR_AXES, I-37
- GKS\$K_GDP_FELLIPSE_FOCII_PT, I-38

elliptic arcs

- GKS\$K_GDP_ELIARC_CTR_AXES_2VEC, I-21
- GKS\$K_GDP_ELIARC_FOCII_2PT, I-23
- GKS\$K_GDP_FELIARC_CTR_AXES_2VEC, I-39
- GKS\$K_GDP_FELIARC_FOCII_2PT, I-40

fill area sets

- GKS\$K_GDP_FILL_AREA_SET, I-26

filled, I-26 to I-43

- radians, I-5
- radius specifications, I-5
- rectangles

- GKS\$K_GDP_FRECT_2PT, I-42
- GKS\$K_GDP_RECT_2PT, I-24

rotation, I-4, I-5

text, I-46 to I-47

- GKS\$K_GDP_RESTRICTED_TEXT, I-46

transformations, I-4

- unfilled, I-9 to I-25
- vector origin point, I-5
- vectors, I-5

GKS

functions

- declared as external, F-1
- multinational font, G-1
- operating state
 - errors, D-17 to D-19

GKS\$ binding

- list of constants, B-1 to B-21

GKS\$CONID

hardcopy workstation types, A-4

- GKS\$INQ_ACTIVE_WS, 11-192 to 11-194
- GKS\$INQ_AVAIL_GDP, 11-96 to 11-98
- GKS\$INQ_CHOICE_STATE, 11-202 to 11-210
- GKS\$INQ_CLIP, 11-157 to 11-159
- GKS\$INQ_COLOR_FAC, 11-22 to 11-25
- GKS\$INQ_COLOR_INDEXES, 11-219 to 11-221
- GKS\$INQ_COLOR_REP, 11-211 to 11-214
- GKS\$INQ_CURRENT_XFORMNO, 11-167 to 11-168
- GKS\$INQ_DEF_CHOICE_DATA, 11-26 to 11-34
- GKS\$INQ_DEF_DEFER_STATE, 11-35 to 11-37
- GKS\$INQ_DEF_LOCATOR_DATA, 11-38 to 11-44
- GKS\$INQ_DEF_PICK_DATA, 11-45 to 11-50
- GKS\$INQ_DEF_STRING_DATA, 11-51 to 11-57

GK\$INQ_DEF_STROKE_DATA, 11-58 to 11-64
 GK\$INQ_DEF_VALUATOR_DATA, 11-65 to 11-71
 GK\$INQ_DYN_MOD_SEG_ATTB, 11-76 to 11-81
 GK\$INQ_DYN_MOD_WS_ATTB, 11-82 to 11-87
 GK\$INQ_FILL_FAC, 11-88 to 11-91
 GK\$INQ_FILL_INDEXES, 11-222 to 11-224
 GK\$INQ_FILL_REP, 11-215 to 11-218
 GK\$INQ_GDP, 11-92 to 11-95
 GK\$INQ_INDIV_ATTB, 11-160 to 11-166
 GK\$INQ_INPUT_DEV, 11-102 to 11-105
 GK\$INQ_INPUT_QUEUE_OVERFLOW, 11-175 to 11-177
 GK\$INQ_LEVEL, 11-10 to 11-12
 GK\$INQ_LOCATOR_STATE, 11-237 to 11-244
 GK\$INQ_MAX_DS_SIZE, 11-72 to 11-75
 GK\$INQ_MAX_WS_STATE_TABLE, 11-99 to 11-101
 GK\$INQ_MAX_XFORM, 11-16 to 11-17
 GK\$INQ_MORE_SIMUL_EVENTS, 11-181 to 11-182
 GK\$INQ_NAME_OPEN_SEG, 11-183 to 11-184
 GK\$INQ_OPEN_WS, 11-195 to 11-197
 GK\$INQ_OPERATING_STATE, 11-188 to 11-189
 GK\$INQ_PAT_FAC, 11-109 to 11-111
 GK\$INQ_PAT_INDEXES, 11-225 to 11-227
 GK\$INQ_PAT_REP, 11-245 to 11-248
 GK\$INQ_PICK_ID, 11-190 to 11-191
 GK\$INQ_PICK_STATE, 11-249 to 11-257
 GK\$INQ_PIXEL, 11-326 to 11-328
 GK\$INQ_PIXEL_ARRAY, 11-329 to 11-333
 GK\$INQ_PIXEL_ARRAY_DIM, 11-334 to 11-338
 GK\$INQ_PLINE_FAC, 11-112 to 11-115
 GK\$INQ_PLINE_INDEXES, 11-228 to 11-230
 GK\$INQ_PLINE_REP, 11-258 to 11-261
 GK\$INQ_PMARK_FAC, 11-116 to 11-120
 GK\$INQ_PMARK_INDEXES, 11-231 to 11-233
 GK\$INQ_PMARK_REP, 11-262 to 11-265
 GK\$INQ_PREDEF_COLOR_REP, 11-121 to 11-123
 GK\$INQ_PREDEF_FILL_REP, 11-124 to 11-127
 GK\$INQ_PREDEF_PAT_REP, 11-128 to 11-131
 GK\$INQ_PREDEF_PLINE_REP, 11-132 to 11-135
 GK\$INQ_PREDEF_PMARK_REP, 11-136 to 11-139
 GK\$INQ_PREDEF_TEXT_REP, 11-140 to 11-144
 GK\$INQ_PRIM_ATTB, 11-169 to 11-174
 GK\$INQ_SEG_ATTB, 11-318 to 11-321
 GK\$INQ_SEG_NAMES, 11-198 to 11-200
 GK\$INQ_SEG_NAMES_ON_WS, 11-266 to 11-268
 GK\$INQ_SEG_PRIORITY, 11-106 to 11-108
 GK\$INQ_SET_ASSOC_WS, 11-322 to 11-324

GK\$INQ_STRING_STATE, 11-269 to 11-275
 GK\$INQ_STROKE_STATE, 11-276 to 11-285
 GK\$INQ_TEXT_EXTENT, 11-286 to 11-289
 GK\$INQ_TEXT_FAC, 11-145 to 11-149
 GK\$INQ_TEXT_INDEXES, 11-234 to 11-236
 GK\$INQ_TEXT_REP, 11-290 to 11-294
 GK\$INQ_VALUATOR_STATE, 11-295 to 11-301
 GK\$INQ_WSTYPE_LIST, 11-13 to 11-15, A-1
 GK\$INQ_WS_CATEGORY, 11-150 to 11-152
 GK\$INQ_WS_CLASSIFICATION, 11-153 to 11-155
 GK\$INQ_WS_DEFER_AND_UPDATE, 11-302 to 11-306
 GK\$INQ_WS_MAX_NUM, 11-18 to 11-20
 GK\$INQ_WS_STATE, 11-310 to 11-312
 GK\$INQ_WS_TYPE, 11-307 to 11-309, A-1
 GK\$INQ_WS_XFORM, 11-313 to 11-316
 GK\$INQ_XFORM, 11-185 to 11-187
 GK\$INQ_XFORM_LIST, 11-178 to 11-180
 GK\$OPEN_WS, A-1
 GK\$WSTYPE, A-3
 GKSM
 ANSI standard, E-1
 GKSM metafiles, E-1

H

Half line, G-1
 See also Fonts
 Handlers
 See also Devices
 See also Inquiry functions
 See also Workstations
 set and realized values, 11-6
 Hardcopy
 workstation types, A-4
 Hatches
 See also Attributes
 Header
 metafile structure, E-3
 Hershey fonts
 See Fonts
 HP7475
 workstation type constant, A-2
 HP7550
 workstation type constant, A-2
 HP7580
 workstation type constant, A-2
 HP7585
 workstation type constant, A-2

Implementation specific errors
list of, D-2 to D-14

Initial position
string input cursor, J-26

Input
data records required, J-15 to J-32
DEC GKS logical input device numbers, J-2 to J-14
DEC GKS specific values, J-1 to J-39
echo area titles, J-13
inquiry functions, 11-4
keypad functionality, J-33 to J-39
keypad zoning mechanism, J-11
list of errors, D-36 to D-39
LOCK key, J-39
string input control characters, J-9
zoning mechanism, J-35

Input priority
initial value, C-5

INQUIRE CHOICE DEVICE STATE, 11-202

INQUIRE CLIPPING, 11-157

INQUIRE COLOR FACILITIES, 11-22

INQUIRE COLOR REPRESENTATION, 11-211

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES, 11-160

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER, 11-167

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES, 11-169

INQUIRE DEFAULT CHOICE DEVICE DATA, 11-26

INQUIRE DEFAULT DEFERRAL STATE VALUES, 11-35

INQUIRE DEFAULT LOCATOR DEVICE DATA, 11-38

INQUIRE DEFAULT PICK DEVICE DATA, 11-45

INQUIRE DEFAULT STRING DEVICE DATA, 11-51

INQUIRE DEFAULT STROKE DEVICE DATA, 11-58

INQUIRE DEFAULT VALUATOR DEVICE DATA, 11-65

INQUIRE DISPLAY SPACE SIZE, 11-72

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES, 11-76

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES, 11-82

INQUIRE FILL AREA FACILITIES, 11-88

INQUIRE FILL AREA REPRESENTATION, 11-215

INQUIRE GENERALIZED DRAWING PRIMITIVE, 11-92

INQUIRE INPUT QUEUE OVERFLOW, 11-175

INQUIRE LEVEL OF GKS, 11-10

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES, 11-96

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES, 11-13

INQUIRE LIST OF COLOR INDICES, 11-219

INQUIRE LIST OF FILL AREA INDICES, 11-222

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS, 11-178

INQUIRE LIST OF PATTERN INDICES, 11-225

INQUIRE LIST OF POLYLINE INDICES, 11-228

INQUIRE LIST OF POLYMARKER INDICES, 11-231

INQUIRE LIST OF TEXT INDICES, 11-234

INQUIRE LOCATOR DEVICE STATE, 11-237

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES, 11-99

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER, 11-16

INQUIRE MORE SIMULTANEOUS EVENTS, 11-181

INQUIRE NAME OF OPEN SEGMENT, 11-183

INQUIRE NORMALIZATION TRANSFORMATION, 11-185

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES, 11-102

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED, 11-106

INQUIRE OPERATING STATE VALUE, 11-188

INQUIRE PATTERN FACILITIES, 11-109

INQUIRE PATTERN REPRESENTATION, 11-245

INQUIRE PICK DEVICE STATE, 11-249

INQUIRE PICK IDENTIFIER VALUE, 11-190

INQUIRE PIXEL, 11-326

INQUIRE PIXEL ARRAY, 11-329

INQUIRE PIXEL ARRAY DIMENSIONS, 11-334

INQUIRE POLYLINE FACILITIES, 11-112

INQUIRE POLYLINE REPRESENTATION, 11-258

INQUIRE POLYMARKER FACILITIES, 11-116

INQUIRE POLYMARKER REPRESENTATION, 11-262

INQUIRE PREDEFINED COLOR REPRESENTATION, 11-121

INQUIRE PREDEFINED FILL AREA REPRESENTATION, 11-124

INQUIRE PREDEFINED PATTERN REPRESENTATION, 11-128

INQUIRE PREDEFINED POLYLINE REPRESENTATION, 11-132

INQUIRE PREDEFINED POLYMARKER REPRESENTATION, 11-136

INQUIRE PREDEFINED TEXT REPRESENTATION, 11-140

INQUIRE SEGMENT ATTRIBUTES, 11-318

INQUIRE SET OF ACTIVE WORKSTATIONS, 11-192
 INQUIRE SET OF ASSOCIATED WORKSTATIONS,
 11-322
 INQUIRE SET OF OPEN WORKSTATIONS, 11-195
 INQUIRE SET OF SEGMENT NAMES IN USE,
 11-198
 INQUIRE SET OF SEGMENT NAMES ON
 WORKSTATION, 11-266
 INQUIRE STRING DEVICE STATE, 11-269
 INQUIRE STROKE DEVICE STATE, 11-276
 INQUIRE TEXT EXTENT, 11-286
 INQUIRE TEXT FACILITIES, 11-145
 INQUIRE TEXT REPRESENTATION, 11-290
 INQUIRE VALUATOR DEVICE STATE, 11-295
 INQUIRE WORKSTATION CATEGORY, 11-150
 INQUIRE WORKSTATION CLASSIFICATION, 11-153
 INQUIRE WORKSTATION CONNECTION AND TYPE,
 11-307
 INQUIRE WORKSTATION DEFERRAL AND UPDATE
 STATES, 11-302
 INQUIRE WORKSTATION MAXIMUM NUMBERS,
 11-18
 INQUIRE WORKSTATION STATE, 11-310
 INQUIRE WORKSTATION TRANSFORMATION,
 11-313
 Inquiry functions, 11-1 to 11-338
 GKS description table, 11-9 to 11-20
 GKS state list, 11-156 to 11-200
 introduction to, 11-1 to 11-8
 list of, 11-8 to 11-338
 pixels, 11-325 to 11-338
 segment state list, 11-317 to 11-324
 workstation description table, 11-21 to 11-155
 workstation state list, 11-201 to 11-316
 Intensities
 color chart, H-2
 Internal structure
 of metafiles, E-1
 ISO standardization>6093 (GKSM metafiles), E-1
 Items
 metafile internal structure, E-1

K

Keypad functionality
 choice input, J-36, J-38
 cycling, J-34
 input, J-33 to J-39
 input zoning, J-35
 Keypad input functionality
 zoning mechanism, J-11

Keys
 input keypad functionality, J-33

L

%LOC
 GDP and escape data records, I-3
 LA100
 workstation type constant, A-2
 LA210
 workstation type constant, A-2
 LA34
 workstation type constant, A-2
 LA50
 aspect ratio, A-3
 workstation type constant, A-2
 Languages
 BASIC, F-3
 C, F-3
 COBOL, F-3
 declaring external functions, F-1
 Pascal, F-7
 programming information, F-1 to F-8
 Layout
 metafile structure, E-4
 LCG01
 workstation type constant, A-2
 Lengths
 metafile items, E-4
 Lines
 of fonts, G-1
 Line types
 DEC GKS specific, C-5
 Lists
 GKS state, 11-1
 segment state, 11-1
 workstation state, 11-1
 Locator
 data records required, J-18
 keypad zoning of cursor, J-35
 logical input device numbers, J-7
 prompt and echo types supported, J-18
 LOCK key
 input on VAXstations, J-39
 Logical input devices
 DEC GKS available numbers, J-2 to J-14
 keypad functionality, J-33
 LVP16
 workstation type constants, A-2

M

Marker types

DEC GKS specific, C-6

Messages

error, D-1 to D-43

Metafiles

CGM, E-9 to E-28

CGM RMS format, E-28

GKSM, E-1 to E-9

Internal structure, E-1

list of errors, D-39 to D-41

RMS format, E-9

workstation type constants, A-1

MicroVAX

workstation type constants, A-2

Monospaced fonts, G-1

MPS-2000 workstation type constant, A-2

N

Normalization transformations

See Transformations

Numbers

error, D-1

Numeric Keypad

choice input, J-36

O

Operating states

list of errors, D-17 to D-19

Output

list of errors, D-33 to D-34

Output attributes

See Attributes

Output-only workstations, A-4

P

Paper sizes, A-4

Pascal programming information, F-7

Passing by descriptor, F-3

problems, F-1

Pick

data records required, J-25

keypad zoning of cursor, J-35

logical input device numbers, J-8

prompt and echo types supported, J-25

Polylines

See also Attributes

Polylines (Cont.)

DEC GKS specific, C-5

initial attributes, C-1

Polymarkers

See also Attributes

DEC GKS specific, C-6

initial attributes, C-2

PostScript workstation type constant, A-3

Precision

See also Fonts

Precision text fonts, G-1

Programming

BASIC, F-3

C, F-3

language-specific information, F-1

Pascal, F-7

Programming COBOL, F-3

Prompts

See Echo area

R

Radians

GDPs, I-5

Radius

GDPs, I-5

Ranges

escapes, I-48

Realized values, 11-6

See also Inquiry Functions

Representations

See also Attributes

color chart, H-2

RMS

CGM metafile structure, E-28

metafile structure, E-9

Rotation

GDPs, I-4, I-5

S

Segments

initial attributes, C-4

list of errors, D-34 to D-36

Set values, 11-6

Sizes

paper, A-4

SMG encoded key

VAXstation string input, J-9

Software fonts, G-1

Standards

metafile structure, E-1

State lists

GKS, 11-1
segments, 11-1
workstation, 11-1

Status

inquiry error status argument, 11-5
values
VMS, D-1

String

data records required, J-26
prompt and echo types supported, J-26

Strings

control characters for input, J-9
initial position of input cursor, J-26
logical input device numbers, J-9
toggling overstrike/insert input, J-9

Stroke

data records required, J-27
keypad zoning mechanism, J-11
keypad zoning of cursor, J-35
logical input device numbers, J-11
prompt and echo types supported, J-27

Stroke font file, G-4

character descriptor, G-4, G-9
elements, G-10
header, G-4, G-5
elements, G-7

Stroke font file header, G-5

elements, G-7

Structure

metafiles, E-1

System errors

list of, D-42 to D-43

T

Tables

GKS description, 11-1
workstation description, 11-1

Tektronix-4014

workstation type constant, A-3

Tektronix-4107

cycling logical input devices, J-34
workstation type constant, A-3

Text

fonts, G-1
initial attributes, C-3

Toggling

logical input devices, J-34

Toggling (Cont.)

overstrike/insert string input, J-9

Top line, G-1

See also Fonts

Transformations

See also Escapes
coordinate range translation, I-132
GDPs, I-4
list of errors, D-25 to D-26
normalization
initial attributes, C-5
translating DC to NDC points, I-136
translating NDC to DC points, I-133
translating NDC to WC points, I-135
translating WC to NDC points, I-132
vector origin points, I-5

Types

hardcopy workstations, A-4
inquiry value type argument, 11-6
workstation, A-1

V

Valuator

data records required, J-31
logical input device numbers, J-12
prompt and echo types supported, J-31

Values

DEC GKS specific line types, C-5
DEC GKS specific marker types, C-6
initial attribute, C-1 to C-5
of constants, B-1 to B-21

VAX Languages, F-1

VAXstation

choice input device numbers, J-4, J-5

VAXstations

color chart, H-1
LOCK key, J-39
SMG encoded input string, J-9
string input device numbers, J-9
workstation type constants, A-2

Vector origin point, I-5

See also GDPs

Vector origin points

transformations, I-5

Vectors

GDPs, I-5

Viewports

normalization
initial value, C-5

VT125

- choice input device numbers, J-4
- choice keypad functionality, J-36
- color chart, H-1
- keypad zoning during input, J-35
- keypad zoning mechanism, J-11
- string input device numbers, J-10
- workstation type constants, A-2

VT240

- choice input device numbers, J-4, J-5
- choice keypad functionality, J-36, J-38
- color chart, H-1
- keypad zoning during input, J-35
- keypad zoning mechanism, J-11
- string input device numbers, J-10
- workstation type constants, A-2

VT330

- workstation type constant, A-2

VT340

- workstation type constant, A-2

W

Windows

- normalization
 - initial value, C-5

WISS

- workstation type constant, A-2

Workstations

- DEC GKS specific input values, J-1 to J-39
- hardcopy types, A-4
- list of errors, D-19 to D-25
- output-only types, A-4
- supported devices, A-1 to A-6
- types
 - as bit masks, A-4
 - bit mask constants, A-5

Z

Zoning

- input cursor, J-35

Reader's Comments

DEC GKS Reference Manual
Part 2
AA-HW44D-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) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Completeness (enough information) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Clarity (easy to understand) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Organization (structure of subject matter) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Figures (useful) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Examples (useful) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Index (ability to find topic) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Page layout (easy to find information) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

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** _____

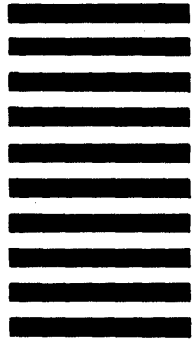
Mailing Address _____
_____ **Phone** _____

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

Reader's Comments

DEC GKS Reference Manual
Part 2
AA-HW44D-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) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Completeness (enough information) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Clarity (easy to understand) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Organization (structure of subject matter) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Figures (useful) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Examples (useful) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Index (ability to find topic) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Page layout (easy to find information) | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

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