DEC GKS C Binding Reference Manual

Order Number: AA–MJ30A–TE

April 1989

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

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

digital equipment corporation
maynard, massachusetts

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

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

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

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

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

The following are trademarks of Digital Equipment Corporation:

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

ZK4631

Contents

Examples

Tables

Preface

Manual Objectives

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

NOTE

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

```
$ HELP  GKS  RELEASE_NOTES RETURN
```

Intended Audience

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

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

Document Structure

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

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

Associated Documents

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

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

Conventions

Convention	Meaning
RETURN	The symbol RETURN represents a single stroke of the RETURN key on a terminal.
$ RUN GKSPROG RETURN	In interactive examples, the user's response to a prompt is printed in red; system prompts are printed in black.
INTEGER X . . . X = 5	A vertical ellipsis indicates that not all of the text of a program or program output is illustrated. Only relevant material is shown in the example.

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

Chapter 1

Introduction

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

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

DEC GKS implements two syntactical language bindings: the GKS C and GKS FORTRAN bindings. The language bindings in general, and specifically the C and FORTRAN bindings, provide standard function names and a standard number of function parameters. If you write programs to be transported across systems or across GKS implementations, you should use the appropriate language binding.

NOTE

The C binding conforms to a draft ANSI standard. Major changes are proposed to the ANSI standard draft. It is probable that the final ANSI draft will differ significantly from the implementation in this release.

1.1 GKS Levels

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

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

1.2 Programming Considerations

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

NOTE

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

1.2.1 Online Help

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

```
$ HELP GKS RETURN
```

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

```
$ HELP GKS RELEASE_NOTES RETURN
```

1.2.2 Capabilities of Supported Devices

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

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

1.2.3 Calling Sequences

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

- **Call Type**

  The DEC GKS C binding calls DEC GKS functions using function references.

- **Function Identifier**

  All DEC GKS C Binding function names begin with the prefix lowercase *g*. The remainder of the name indicates the operation performed by the function.

- **Argument List**

  Arguments that are passed to DEC GKS functions must be listed in the order shown in the syntax descriptions contained in this manual. The language binding functions may have a different argument list than the corresponding GKS$ function.

1.3 C Binding Function Syntax

The command section lists the following for each C binding function:

- C binding (descriptive) function name
- Valid DEC GKS operating states during which you can use the function
- Function syntax listing
- Brief description of the arguments

The argument description identifies the data type, indicates whether the argument is an input or output parameter, identifies the passing mechanism, and provides a brief textual description of the argument. The C binding uses the following data types:

- Gchar—to specify a null terminated character string
- Gfloat—to specify real values
- Gint—to specify integer values
- Gfile—to specify the file object type
- Guint—to specify unsigned integer values

1.4 C Binding Data Types

Independent C binding data type definitions are numerous, but are usually classified in one of the following three categories:

- Structures
- Unions
- Enumerated types

When these three independent data types are defined in a function, they are documented in the format of Example 1–1, which illustrates all three data types as they are used in function ginitchoice (INITIALIZE CHOICE).

INITIALIZE CHOICE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitchoice *(ws, dev, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
ws	Gint	I	By value	Workstation identifier
dev	Gint	I	By value	Choice device number
init	Gchoice (structure)	I	By reference	Initial choice and status pointer
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gchoicerec (structure)	I	By reference	Choice data record structure

Example 1–1: Example of C Binding Data Types

```
❶ struct Gchoice {                    /*I Choice data*/
       enum Gcstat {                  /*I Choice data - Enumerated type*/
       } status;                      /*I Choice status*/
       Gint choice;                   /*I Choice number*/
} init;
```

(continued on next page)

Example 1–1 (Cont.): Example of C Binding Data Types

```
struct Glimit {                     /*I Coordinate limits*/
    Gfloat xmin;                    /*I X minimum limit*/
    Gfloat xmax;                    /*I X maximum limit*/
    Gfloat ymin;                    /*I Y minimum limit*/
    Gfloat ymax;                    /*I Y maximum limit*/
} area;

❷ struct Gchoicerec {               /*I Choice data record structure*/
    union {
        struct Gchoicepet0001 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gint  *lengths;         /*I Lengths of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet1_datarec;
        struct Gchoicepet0002 {     /*I */
            Gint  number;           /*I Number of alternatives*/
            enum Gprflag *enable;   /*I Array of prompt flags - enumerated type*/
            Gchar *title_string;    /*I The title string */
        } choicepet2_datarec;
        struct Gchoicepet0003 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet3_datarec;
        struct Gchoicepet0004 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet4_datarec;
        struct Gchoicepet0005 {     /*I */
            Gint  seg;              /*I Segment name*/
            Gint  number;           /*I Number of alternatives*/
            Gint  *pickids;         /*I Array of pick indentifiers*/
            Gchar *title_string;    /*I The title string */
        } choicepet5_datarec;
```

(continued on next page)

Example 1–1 (Cont.): Example of C Binding Data Types

```
        struct Gchoicepet_0001 {   /*I */
            Gint  number;          /*I Number of choice strings*/
            Gint  *lengths;        /*I Lengths of choice strings*/
            Gchar **strings;       /*I Array of strings*/
            Gchar *title_string;   /*I The title string */
        } choicepet_1_datarec;
    }
} record;
```

Data type	Constant	Description
❸ Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Input is NOCHOICE

Data type	Constant	Description
Gprflag	GPROFF	Prompt off
	GPRON	Prompt on

❶ Structure Gchoice includes two variables, or members, (init, choice) of different data types (Gcstat, Gint). Structure Gchoice includes two variables because they are related and easily handled together.

❷ Structure Gchoicerec is a union, which holds objects of different types and sizes so that the objects can be manipulated within a single area of storage.

❸ Structure Gcstat is an enumerated type. Enumerated types hold several constant values for each declared structure.

An I or O in the commented section indicates an input or output argument, respectively.

An M indicates a modifiable parameter, both input and output.

1.5 C Binding Passing Mechanism

All arguments are passed by value or reference, where reference means by the argument's address.

To show that passing by reference means passing at the address of the argument, consider the following example for gawaitevent:

```
Gfloat timeout;
Struct Gevent event;

gawaitevent ( timeout, &event)
```

The following tables and code example show the function description for gawaitevent.

Argument	Data Type	I/O	Mechanism	Description
timeout	Gfloat	I	By value	Timeout value (in seconds)
event	Gevent (structure)	O	By reference	Event data

```
struct Gevent {                          /*I Event*/
    Gint ws;                             /*O Workstation identifier*/
    Gint dev;                            /*O Logical input device number*/
    struct Giclass {                     /*O Input device class - Enumerated type*/
    } class;
} event;
```

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

1.6 Standard Escape/GDP Data Records

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

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

1. Look up the escape or GDP description in Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual.*
2. Determine the size and contents of the required data record (if one is required).
3. Declare the data record as determined by your particular programming language. Each of the seven components of the data record is an integer value. The record is read only, passed by reference.
4. Pass to gescape or ggdp only the data record components required by the escape or GDP. For instance, if an escape or GDP only requires 5 data record components, omit values from components 6 and 7.
5. Pass to gescape or ggdp the exact size of the valid portion of the data record, as specified in Appendix I, DEC GKS GDPs and Escapes, in the *DEC GKS Reference Manual.* For instance, if an escape or GDP requires 5 valid components to the data record, then pass the value 20 as the data record size (each component being a longword in length).

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

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

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

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

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

NOTE

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

1.7 C Binding Function Organization

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

- Control functions
- Output functions
- Attribute functions
- Transformation functions
- Input functions
- Segment functions
- Metafile functions
- Error-handling functions
- Utility functions
- Inquiry functions
  - GKS Description Table functions
  - Workstation Description functions
  - GKS State List functions
  - Workstation State List functions
  - Segment functions
  - Pixel functions

For quick reference, use one of the following means to locate a C binding function description:

- Locate a C binding category. For example, locate the category attribute functions. Within that category find the specific function name and page number.
- Refer to Appendix A, which lists the DEC GKS function names with their corresponding C binding function names. Function names are organized according to functional category.

Chapter 2

Compiling, Linking, and Running DEC GKS Programs on VMS

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

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

The C binding provides the GKS functions and data types in both a natural and efficient manner using C facilities, without violating the style or design philosophy of C.

2.1 Binding Function Names

The C binding specifies a set of standard function identifiers. Each of the DEC GKS function identifiers that use the GKS$ interface correspond to one or more of the C binding identifiers. In Appendix A, Table A–1 lists the mapping from the DEC GKS function names to the C binding function names.

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

The C binding names abbreviate the names commonly found in the DEC GKS standard. For instance, all functions begin with lowercase g (for GKS); the binding abbreviation for workstation is ws. Consequently, the C binding function name for ACTIVATE WORKSTATION is gactivatews.

2.2 Compiling, Linking, and Running

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

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

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

If you choose to make reference to the C binding constant values, you need to include the appropriate definitions file into your source program. See Section 2.3 for more information on including definition files.

2.3 Including Definition Files

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

- The C Binding definition file located in SYS$LIBRARY is GKS.H. You may need some additional definitions from SYS$LIBRARY GKSDEFS.H
- The C Binding completion status code definition file located in SYS$LIBRARY is GKSMSGS.H.

Each of these files includes comments that describe the exact method for using them.

2.3.1 Buffer Management for Inquiry and Input Functions

The application program is responsible for providing the memory buffer that the inquiry and input functions require. This buffer must be large enough to accommodate any requested returned data. If the binding data in any of the inquiry or input functions cannot fit into the application's buffer, GKS may return an error condition. In some cases, GKS returns the size of the buffer required to return the actual data.

Inquiry and input functions that return data to an application buffer fall into three categories:

1. Functions that return structures of known size

   When these functions are returned, they assume the application has a buffer of adequate size for the returned data. These returned functions are inquiries to a single object of fixed size. The user determines the size of the object by using the standard C binding **sizeof** function.

   Input functions in this category include get, sample, and request functions. The application program knows the buffer size to accommodate the returned data in one of two ways:

   - Through inquiry
   - By setting the buffer size when the device is initialized

2. Functions that return lists of objects

   Many of the lists of arrays of objects on which you can inquire are virtually unbounded. The mechanism that allows you to inquire a finite subset of the objects, starting at a particular object within the list, requires two input parameters:

   - The number of objects that can fit into the user's buffer
   - An object starting index

   If the start parameter on which you inquire is out of the range of these two input parameters, GKS returns the error **EOUTRANG**.

3. Functions that return a structure which contains one or more variable size members application program provides the buffer and its size in units, as a returned value. The inquiry proceeds normally if the buffer size is sufficient. If the buffer size in insufficient, the inquiry fails. The application program must increase buffer size and retry the inquiry.

2.4 Using User-Defined Error-Handling Functions

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

If you do replace gerrorhand with an external function of your own, you need to link your object files as follows:

```
$ LINK file_name.OBJ, error_handler.OBJ RETURN
```

Chapter 3

Compiling, Linking, and Running DEC GKS Programs on ULTRIX

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

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

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

3.1 ULTRIX Programming Considerations

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

3.1.1 Including Definition Files

You use DEC GKS software primarily by placing calls to DEC GKS functions in your program. However, when using DEC GKS, you need statements in your program other than calls to GKS functions. The specific statements that are needed depend on the VAX language you use.

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

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

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

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

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

- gksmsgs.h for VAX C

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

3.1.2 Compiling and Linking C Programs

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

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

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

NOTE

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

3.1.3 Environment Variables and DEC GKS Programming

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

```
❶       CALL GKS$OPEN_GKS( stderr )

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

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

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

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

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

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

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

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

```
csh> setenv  GKSconid  /dev/tty RETURN
csh> setenv  GKSwstype  13
csh> # VT241 Color RETURN
csh> application RETURN
   .
   .
   .
csh> setenv  GKSconid  /dev/tt00 RETURN
csh> setenv  GKSwstype  13
csh> # VT241 Color RETURN
csh> application RETURN
   .
   .
   .
```

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

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

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

3.1.3.1 Specifying Bit Masks as Workstation Type Values

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

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

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

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

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

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

Chapter 4

Control Functions

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

Category	GKS Functions
GKS Environment	gopengks, gclosegks
Workstation Environment	gopenws, gactivatews, gdeactivatews, gclosews
Display Surface Control	gclearws, gredrawsegws, gsetdeferst, gupdatews
Additional Control	gescape, gmessage

ACTIVATE WORKSTATION

Operating States: GWSOP, GWSAC

Syntax

gactivatews *(workstation_id)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier

CLEAR WORKSTATION

Operating States: GWSOP, GWSAC

Syntax

gclearws *(workstation_id, clearflag)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
clearflag	Gclrflag (enumerated)	I	By value	Clear control flag

Data type	Constant	Description
Gclrflag	GCONDITIONALLY	Clear if the surface is not empty
	GALWAYS	Clear the workstation

CLOSE GKS

Operating States: GGKOP

Syntax

gclosegks *( )*

CLOSE WORKSTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gclosews *(workstation_id)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier

DEACTIVATE WORKSTATION

Operating States: GWSAC

Syntax

gdeactivatews *(workstation_id)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier

ESCAPE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gescape *(function, indata, bufsize, outdata, escout_size)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
function	Gint	I	By value	Function identifier
indata	Gescin (union)	I	By reference	The escape input data record
bufsize	Gint	I	By value	Size of input data record, in bytes
outdata	Gescout (union)	O	By reference	The escape output data record
escout_size	Gint	M	By reference	Size of the output data record, in bytes

```
struct Gescin {                          /*I Escape input structure*/
    union {
        struct Guesc_idatarec {          /*I ESCAPE dependent data record*/
            Gint number_integer;         /*I Number of integers in list of integers*/
            Gint number_float;           /*I Number of floats in list of floats*/
            Gint number_strings;         /*I Number of strings in list of strings*/
            Gint *list_integers;         /*I List of integers*/
            Gfloat *list_floats;         /*I List of floats*/
            Gint *list_string_lengths;   /*I List of string lengths*/
            Gchar **list_strings;        /*I List of strings*/
        } esc_idatarec;
    }
} indata;

struct Gescout {                         /*O Escape output structure/*
    union {
        struct Guesc_odatarec {          /*O ESCAPE dependent data record*/
            Gint number_integer;         /*O Number of integers in list of integers*/
            Gint number_float;           /*O Number of floats in list of floats*/
            Gint number_strings;         /*O Number of strings in list of strings*/
            Gint *list_integers;         /*O List of integers*/
            Gfloat *list_floats;         /*O List of floats*/
            Gint *list_string_lengths;   /*O List of string lengths*/
            Gchar **list_strings;        /*O List of strings*/
        } esc_odatarec;
    }
} outdata;
```

NOTE

If you pass the value 0 as the argument escout_size (size of the output data record), the function gescape checks for errors, returns the size of the output data record in escout_size, but does *not* perform the escape. You can use this functionality to check the size of the returned output data record without performing the escape action.

MESSAGE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gmessage *(workstation_id, message)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
message	Gchar	I	By reference	Message text

OPEN GKS

Operating States: GGKCL

Syntax

gopengks *(errfile, memory)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
errfile	Gfile	I	By reference	Error file pointer
memory	Glong	I	By value	Amount of memory available for buffer space

OPEN WORKSTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gopenws *(workstation_id, conid, type)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
conid	Gconn (Gchar)	I	By reference	Connection identifier
type	Gwstype (Gint)	I	By reference	Workstation type

REDRAW ALL SEGMENTS ON WORKSTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gredrawsegws *(workstation_id)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier

SET DEFERRAL STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetdeferst *(workstation_id, defmode, irgmode)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
defmode	Gdefmode (enumerated)	I	By value	Deferral mode
irgmode	Girgmode (enumerated)	I	By value	Implicit regeneration mode

Data type	Constant	Description
Gdefmode	GASAP	As soon as possible
	GBNIG	Before next interaction globally
	GBNIL	Before next interaction locally
	GASTI	At some time

Data type	Constant	Description
Girgmode	GSUPPRESSED	Image regeneration is suppressed
	GALLOWED	Image regeneration is allowed

UPDATE WORKSTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gupdatews *(workstation_id, regenflag)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
regenflag	Gregen (enumerated)	I	By value	Update regeneration flag

Data type	Constant	Description
Gregen	GPOSTPONE	Supress regeneration of images
	GPERFORM	Perform regeneration

Chapter 5

Output Functions

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

Category	GKS Functions
Draw connected lines.	gpolyline
Mark locations with symbols.	gpolymarker
Draw text.	gtext
Fill a polygon.	gfillarea
Color cells of a rectangle.	gcellarray
Draw generalized drawing primitive.	ggdp

CELL ARRAY

Operating States: GWSAC, GSGOP

Syntax

gcellarray *(rectangle, dimensions, color)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
rectangle	Grect (structure)	I	By reference	Coordinate rectangle
dimensions	Gidim (structure)	I	By reference	Color index array dimensions pointer
color	Gint	I	By reference	Color index array

```
struct Grect {                          /*I Coordinate rectangle pointer*/
    struct Gpoint {                     /*I Coordinate point*/
        Gfloat x;                       /*I Upper left X coordinate*/
        Gfloat y;                       /*I Upper left Y coordinate*/
    } ul;
    struct Gpoint {                     /*I Coordinate point*/
        Gfloat x;                       /*I Lower right X coordinate*/
        Gfloat y;                       /*I Lower right Y coordinate*/
    } lr;
} rectangle;

struct Gidim {                          /*I Dimension in integer values*/
    Guint x_dim;                        /*I X dimension*/
    Guint y_dim;                        /*I Y dimension*/
} dimensions;
```

FILL AREA

Operating States: GWSAC, GSGOP

Syntax

gfillarea *(number_of_points, points)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
number_of_ points	Gint	I	By value	Number of points
points	Gpoint (structure)	I	By reference	Array of Gpoint structure

```
struct Gpoint {                 /*I Coordinate point*/
    Gfloat x;                   /*I X coordinate*/
    Gfloat y;                   /*I Y coordinate*/
} points;
```

GENERALIZED DRAWING PRIMITIVE

Operating States: GWSAC, GSGOP

Syntax

ggdp *(number_of_points, points, function, data)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
number_of_ points	Gint	I	By value	Number of points
points	Gpoint (structure)	I	By reference	Array of Gpoint structure
function	Gint	I	By value	GDP identifier
data	Ggdprec (structure)	I	By reference	GDP data record pointer

```
struct Gpoint {                          /*I Coordinate point*/
    Gfloat x;                            /*I X coordinate*/
    Gfloat y;                            /*I Y coordinate*/
} points;

struct Ggdprec {                         /*I GDP data record/*
    union {
        struct Gugdp_datarec {           /*I GDP dependent data record*/
            Gint number_integer;         /*I Number of integers in list of integers*/
            Gint number_float;           /*I Number of floats in list of floats*/
            Gint number_strings;         /*I Number of strings in list of strings*/
            Gint *list_integers;         /*I List of integers*/
            Gfloat *list_floats;         /*I List of floats*/
            Gint *list_string_lengths;   /*I List of string lengths*/
            Gchar **list_strings;        /*I List of strings*/
        } gugdp_datarec;
    }
} data;
```

POLYLINE

Operating States: GWSAC, GSGOP

Syntax

gpolyline *(number_of_points, points)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
number_of_ points	Gint	I	By value	Number of points
points	Gpoint (structure)	I	By reference	Array of Gpoint structure

```
struct Gpoint {                 /*I Coordinate point*/
    Gfloat x;                   /*I X coordinate*/
    Gfloat y;                   /*I Y coordinate*/
} points;
```

POLYMARKER

Operating States: GWSAC, GSGOP

Syntax

gpolymarker *(number_of_points, points)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
number_of_ points	Gint	I	By value	Number of points
points	Gpoint (structure)	I	By reference	Array of Gpoint structure

```
struct Gpoint {                 /*I Coordinate point*/
    Gfloat x;                   /*I X coordinate*/
    Gfloat y;                   /*I Y coordinate*/
} points;
```

TEXT

Operating States: GWSAC, GSGOP

Syntax

gtext *(position, text)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
position	Gpoint (structure)	I	By reference	Array of Gpoint structure
text	Gchar	I	By reference	The text string

```
struct Gpoint {                    /*I Coordinate point*/
    Gfloat x;                      /*I X coordinate*/
    Gfloat y;                      /*I Y coordinate*/
} position;
```

Chapter 6

Output Attributes

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

Category	GKS Functions
Fill Area Attributes	gsetfillcolourind, gsetfillind, gsetfillintstyle, gsetfillstyleind, gsetpatrefpt, gsetpatsize
Polyline Attributes	gsetlinecolourind, gsetlineind, gsetlinetype, gsetlinewidth
Polymarker Attributes	gsetmarkercolourind, gsetmarkerind, gsetmarkersize, gsetmarkertype
Primitive Attributes	gsetpickid
Text Attributes	gsettextalign, gsettextcolourind, gsetcharexpan, gsettextfontprec, gsetcharheight, gsettextind, gsettextpath, gsetcharspace, gsetcharup
Aspect Source Flags	gsetasf
Representations	gsetcolourrep, gsetfillrep, gsetpatrep, gsetlinerep, gsetmarkerrep, gsettextrep

SET ASPECT SOURCE FLAGS

Operating States: GGKOP, GWSAC, GSGOP

Syntax

gsetasf *(asfs)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
asfs	Gasfs (structure)	I	By reference	Array of 13 aspect source flags

```
struct Gasfs {              /*I Aspect source flags structure*/
     enum Gasf ln_type;     /*I Line type - Enumerated type*/
     enum Gasf ln_width;    /*I Line width - Enumerated type*/
     enum Gasf ln_colour;   /*I Line colour - Enumerated type*/
     enum Gasf mk_type;     /*I Marker type - Enumerated type*/
     enum Gasf mk_size;     /*I Marker size - Enumerated type*/
     enum Gasf mk_colour;   /*I Marker colour - Enumerated type*/
     enum Gasf tx_fp;       /*I Text font and precision - Enumerated type*/
     enum Gasf tx_exp;      /*I Text expansion - Enumerated type*/
     enum Gasf tx_space;    /*I Text character spacing - Enumerated type*/
     enum Gasf tx_colour;   /*I Text colour - Enumerated type*/
     enum Gasf fl_inter;    /*I Fill area interior style - Enumerated type*/
     enum Gasf fl_style;    /*I Fill area style index - Enumerated type*/
     enum Gasf fl_colour;   /*I Fill area colour - Enumerated type*/
} asfs;
```

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

SET CHARACTER EXPANSION FACTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetcharexpan *(exp)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
exp	Gfloat	I	By value	Character expansion factor

SET CHARACTER HEIGHT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetcharheight *(height)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
height	Gfloat	I	By value	Character height in world coordinates

SET CHARACTER SPACING

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetcharspace *(spacing)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
spacing	Gfloat	I	By value	Character spacing

SET CHARACTER UP VECTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetcharup *(charup)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
charup	Gpoint (structure)	I	By reference	Character up vector pointer

```
struct Gpoint {                 /*I Coordinate point*/
    Gfloat x;                   /*I X coordinate*/
    Gfloat y;                   /*I Y coordinate*/
} charup;
```

SET COLOUR REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetcolourrep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Color index
rep	Gcobundl (structure)	I	By reference	Color representation pointer

```
struct Gcobundl {                   /*I Color bundle*/
    Gfloat red;                     /*I Red intensity*/
    Gfloat green;                   /*I Green intensity*/
    Gfloat blue;                    /*I Blue intensity*/
} rep;
```

SET FILL AREA COLOUR INDEX

Operating States: GGKOP, GWSAC, GSGOP

Syntax

gsetfillcolourind *(colour)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
colour	Gint	I	By value	Fill area color index

SET FILL AREA INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetfillind *(index)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	I	By value	Fill area index

SET FILL AREA INTERIOR STYLE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetfillintstyle *(style)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
style	Gflinter (enumerated)	I	By value	Fill area interior style

Data type	Constant	Description
Gflinter	GHOLLOW	An outline
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Crossed or parallel lines

SET FILL AREA REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetfillrep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Pattern index
rep	Gflbundl (structure)	I	By reference	Pattern representation pointer

```
struct Gflbundl {              /*I Fill area bundle*/
    enum Gflinter inter;       /*I Fill area interior style - enumerated*/
    Gint style;                /*I Fill area style index*/
    Gint colour;               /*I Fill area colour index*/
} rep;
```

Data type	Constant	Description
Gflinter	GHOLLOW	An outline
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Crossed or parallel lines

SET FILL AREA STYLE INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetfillstyle *(index)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	I	By value	Fill area style index

SET LINETYPE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetlinetype *(type)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
type	Glntype (enumerated)	I	By reference	Line type

Data type	Constant	Value	Description
Glntype	GLN_SOLID	1	Solid line
	GLN_DASHED	2	Dashed line
	GLN_DOTTED	3	Dotted line
	GLN_DASHDOT	4	Dashed and dotted line
	GLN_TRIPLE_DOT	–8	Triple dotted line
	GLN_DOUBLE_DOT	–7	Double dotted line
	GLN_SPACED_DOT	–6	Spaced dotted line
	GLN_SPACED_DASH	–5	Spaced dashed line
	GLN_LONG_SHORT_DASH	–4	Long and short dashed line

Data type	Constant	Value	Description
	GLN_LONG_ DASH	–3	Long dashed line
	GLN_DASH_3_ DOT	–2	Dashed line with three dots
	GLN_DASH_2_ DOT	–1	Dashed line with two dots

NOTE

Beginning at GLN_TRIPLE_DOT, the enumerated type changes value to –8. Do not use the value. Use instead the proper name for the enumerated type. Your program may not compile correctly if you use the enumerated type value.

SET LINEWIDTH SCALE FACTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetlinewidth *(width)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
width	Gfloat	I	By value	Linewidth scale factor

SET MARKER SIZE SCALE FACTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetmarkersize *(size)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
size	Gfloat	I	By value	Polymarker size scale factor

SET MARKER TYPE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetmarkertype *(type)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
type	Gmktype (enumerated)	I	By value	Polymarker type

Data type	Constant	Value	Description
Gmktype	GMK_POINT	1	Point
	GMK_PLUS	2	Plus sign
	GMK_STAR	3	Star
	GMK_O	4	Circle
	GMK_X	5	X (Cross)
	GMK_SOLID_ DIAMOND	–13	Solid Diamond
	GMK_DIAMOND	–12	Diamond
	GMK_SOLID_ HGLASS	–11	Solid Hourglass
	GMK_HOURGLASS	–10	Hourglass
	GMK_SOLID_ BOWTIE	–9	Solid Bowtie
	GMK_BOWTIE	–8	Bowtie

Data type	Constant	Value	Description
	GMK_SOLID_ SQUARE	–7	Solid Square
	GMK_SQUARE	–6	Square
	GMK_SOLID_TRI_ DOWN	–5	Solid Triangle Down
	GMK_TRIANGLE_ DOWN	–4	Triangle Down
	GMK_SOLID_TRI_ UP	–3	Solid Triangle Up
	GMK_TRIANGLE_ UP	–2	Triangle Up
	GMK_SOLID_ CIRCLE	–1	Solid Circle

NOTE

Beginning at GMK_SOLID_DIAMOND, the enumerated type changes value to –13. Do not use the value. Use instead the proper name for the enumerated type. Your program may not compile correctly if you use the enumerated type value.

SET PATTERN REFERENCE POINT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetpatrefpt *(patref)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
patref	Gpoint (structure)	I	By reference	Reference point pointer

```
struct Gpoint {                 /*I Coordinate point*/
    Gfloat x;                   /*I X coordinate*/
    Gfloat y;                   /*I Y coordinate*/
} patref;
```

SET PATTERN REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetpatrep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Pattern index
rep	Gptbundl (structure)	I	By reference	Pattern representation pointer

```
struct Gptbundle {                  /*I Pattern representation bundle*/
    struct Gipoint {                /*I Integer point*/
        Gint x;                     /*I X coordinate*/
        Gint y;                     /*I Y coordinate*/
    } size;
    Gint *array;                    /*I Pattern array*/
} rep;
```

SET PATTERN SIZE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetpatsize *(patsize)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
patsize	Gpoint (structure)	I	By reference	Array of Gpoint structure

```
struct Gpoint {                    /*I Size of pattern in world coordinates*/
    Gfloat x;                      /*I Pattern width*/
    Gfloat y;                      /*I Pattern height*/
} patsize;
```

SET PICK IDENTIFIER

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetpickid *(pickid)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
pickid	Gint	I	By value	Pick identifier

SET POLYLINE COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetlinecolourind *(colour)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
colour	Gint	I	By value	Polyline color index

SET POLYLINE INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetlineind *(index)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	I	By value	Polyline index

SET POLYLINE REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetlinerep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Polyline index value
rep	Glnbundl (structure)	I	By reference	Polyline representation pointer

```
struct Glnbundl {                    /*I Polyline bundle*/
    Gint type;                       /*I Line type*/
    Gfloat width;                    /*I Linewidth scale factor*/
    Gint colour;                     /*I Polyline colour index*/
} rep;
```

SET POLYMARKER COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetmarkercolourind *(colour)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
colour	Gint	I	By value	Polymarker color index

SET POLYMARKER INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetmarkerind *(index)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	I	By value	Polymarker index

SET POLYMARKER REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetmarkerrep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Polymarker index
rep	Gmkbundl (structure)	I	By reference	Polymarker representation pointer

```
struct Gmkbundl {                    /*I Polymarker bundle*/
    Gint type;                       /*I Marker type*/
    Gfloat size;                     /*I Marker size scale factor*/
    Gint colour;                     /*I Polymarker colour index*/
} rep;
```

SET TEXT ALIGNMENT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsettextalign *(txalign)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
txalign	Gtxalign (structure)	I	By reference	Text alignment pointer

```
struct Gtxalign {           /*I Text alignment*/
    enum Gtxhor {           /*I Horizontal alignment - Enumerated type*/
    } hor;
    enum Gtxver {           /*I Vertical alignment - Enumerated type*/
    } ver;
} txalign;
```

Data type	Constant	Description
Gtxhor	GAH_NORMAL	Normal
	GAH_LEFT	Left
	GAH_CENTRE	Center
	GAH_RIGHT	Right

Data type	Constant	Description
Gtxver	GAV_NORMAL	Normal
	GAV_TOP	Top
	GAV_CAP	Cap
	GAV_HALF	Half
	GAV_BASE	Base
	GAV_BOTTOM	Bottom

SET TEXT COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsettextcolourind *(colour)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
colour	Gint	I	By value	Color index

SET TEXT FONT AND PRECISION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsettextfontprec *(txfp)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
txfp	Gtxfp (structure)	I	By reference	Text and font precision pointer

```
struct Gtxfp {                  /*I Text facilities*/
    Gint font;                  /*I Text font*/
    enum Gtxprec {              /*I Text precision - Enumerated type*/
    } prec;
} txfp;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

SET TEXT INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsettextind *(index)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	I	By value	Text index value

SET TEXT PATH

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsettextpath *(text_path)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
text_path	Gtxpath (enumerated)	I	By value	Text path

Data type	Constant	Description
Gtxpath	GTP_RIGHT	Right
	GTP_LEFT	Left
	GTP_UP	Up
	GTP_DOWN	Down

SET TEXT REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsettextrep *(workstation_id, index, rep)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Polymarker index
rep	Gtxbundl (structure)	I	By reference	Text representation pointer

```
struct Gtxbundl {                   /*I Text bundle*/
    Gtxfp fp;                       /*I Font and precision*/
    Gfloat exp;                     /*I Character expansion*/
    Gfloat space;                   /*I Character spacing*/
    Gint colour;                    /*I Polymarker colour index*/
} rep;

struct Gtxfp {                      /*I Text facilities*/
    Gint font;                      /*I Text font*/
    enum Gtxprec {                  /*I Text precision - Enumerated type*/
    } prec;
} fp;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

Chapter 7

Transformation Functions

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

Category	GKS Functions
Normalization	gseltran, gsetclip, gsetviewport, gsetviewportinputpri, gsetwindow
Workstation	gsetwsviewport, gsetwswindow

SELECT NORMALIZATION TRANSFORMATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gselntran *(transform)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
transform	Gint	I	By value	Normalization transformation number

SET CLIPPING INDICATOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetclip *(indicator)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
indicator	Gclip (enumerated)	I	By value	Clipping indicator

Data type	Constant	Description
Gclip	GNOCLIP	Clipping is off
	GCLIP	Clipping is on

SET WINDOW

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetwindow *(transform, window)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
transform	Gint	I	By value	Normalization transformation number
window	Glimit (structure)	I	By reference	World coordinate window limits pointer

```
struct Glimit {                    /*I Coordinate limits*/
    Gfloat xmin;                   /*I X minimum limit*/
    Gfloat xmax;                   /*I X maximum limit*/
    Gfloat ymin;                   /*I Y minimum limit*/
    Gfloat ymax;                   /*I Y maximum limit*/
} window;
```

SET VIEWPORT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetviewport *(transform, viewport)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
transform	Gint	I	By value	Normalization transformation number
viewport	Glimit (structure)	I	By reference	NDC coordinate viewport limits pointer

```
struct Glimit {                    /*I Coordinate limits*/
    Gfloat xmin;                   /*I X minimum limit*/
    Gfloat xmax;                   /*I X maximum limit*/
    Gfloat ymin;                   /*I Y minimum limit*/
    Gfloat ymax;                   /*I Y maximum limit*/
} viewport;
```

SET VIEWPORT INPUT PRIORITY

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gsetviewportinputpri *(transform, reference, priority)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
transform	Gint	I	By value	Normalization transformation number
reference	Gint	I	By value	Reference normalization transformation number
priority	Gvpri (enumerated)	I	By value	Relative priority limits pointer

Data type	Constant	Description
Gvpri	GHIGHER	Higher priority
	GLOWER	Lower priority

SET WORKSTATION WINDOW

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetwswindow *(workstation_id, window)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
window	Glimit (structure)	I	By reference	Points that determine the rectangular area of the workstation window within the NDC space

```
struct Glimit {                    /*I Coordinate limits*/
    Gfloat xmin;                   /*I X minimum limit*/
    Gfloat xmax;                   /*I X maximum limit*/
    Gfloat ymin;                   /*I Y minimum limit*/
    Gfloat ymax;                   /*I Y maximum limit*/
} window;
```

SET WORKSTATION VIEWPORT

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetwsviewport *(workstation_id, viewport)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
viewport	Glimit (structure)	I	By reference	Points that determine the rectangular area of the workstation viewport within the device coordinate space

```
struct Glimit {                    /*I Coordinate limits*/
    Gfloat xmin;                   /*I X minimum limit*/
    Gfloat xmax;                   /*I X maximum limit*/
    Gfloat ymin;                   /*I Y minimum limit*/
    Gfloat ymax;                   /*I Y maximum limit*/
} window;
```

Chapter 8

Input Functions

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

Category	GKS Functions
Initialization	ginitchoice, ginitloc, ginitpick, ginitstring, ginitstroke, ginitval
Mode Control	gsetchoicemode, gsetlocmode, gsetpickmode, gsetstringmode, gsetstrokemode, gsetvalmode
Request Mode	greqchoice, greqloc, greqpick, greqstring, greqstroke, greqval
Sample Mode	gsamplechoice, gsampleloc, gsamplepick, gsamplestring, gsamplestroke, gsampleval
Event Mode	gawaitevent, gflushevents, ggetchoice, ggetloc, ggetpick, ggetstring, ggetstroke, ggetval

AWAIT EVENT

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gawaitevent *(timeout, event)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
timeout	Gfloat	I	By value	Timeout value (in seconds)
event	Gevent (structure)	O	By reference	Event data

```
struct Gevent {                 /*O Event*/
    Gint ws;                    /*O Workstation identifier*/
    Gint dev;                   /*O Logical input device number*/
    enum Giclass {              /*O Input device class - Enumerated type*/
    } class;
} event;
```

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

FLUSH DEVICE EVENTS

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gflushevents *(workstation_id, class, device_number)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
class	Giclass (enumerated)	I	By value	Device class
device_number	Gint	I	By value	Logical input device number

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

GET CHOICE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetchoice *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gchoice (structure)	O	By reference	Choice event

```
struct Gchoice {                    /*O Choice data*/
     enum Gcstat {                  /*O Choice data - Enumerated type*/
     } status;                      /*O Choice status*/
     Gint choice;                   /*O Choice number*/
} response;
```

Data type	Constant	Description
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Input is NOCHOICE
	GC_NONE	No input obtained

GET LOCATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetloc *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gloc (structure)	O	By reference	Locator event

```
struct Gloc {                   /*O Locator data*/
     Gint transform;            /*O Normalization transformation number*/
     struct Gpoint {            /*O Locator position*/
         Gfloat x;              /*O X coordinate*/
         Gfloat y;              /*O Y coordinate*/
     } position;
} response;
```

GET PICK

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetpick *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gpick (structure)	O	By reference	Pick event

```
struct Gpick {                    /*O Pick data*/
     enum Gpstat {                /*O Pick data - Enumerated type*/
     } status;                    /*O Pick status*/
     Gint seg;                    /*O Pick segment*/
     Gint pickid;                 /*O Pick identifier*/
} response;
```

Data type	Constant	Description
Gpstat	GP_OK	Input obtained
	GP_NOPICK	Input is NOPICK
	GP_NONE	No input obtained

GET STRING

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetstring *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gchar	M	By reference	String event

NOTE

Initialize the response parameter with a string, before the call. GKS will do a strlen to get the size, and will use this size as the maximum size of the returned string.

GET STROKE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetstroke *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gstroke (structure)	O	By reference	Stroke event

```
struct Gstroke {              /*O Stroke data*/
     Gint transform;          /*O Normalization transformation number*/
     Gint n_points;           /*O Number of points in stroke*/
     struct Gpoint {          /*O Points in stroke (user supplied stroke
                                buffer)*/
          Gfloat x;           /*O X coordinate*/
          Gfloat y;           /*O Y coordinate*/
     } *points;
} response;
```

NOTE

N_points should be initialized with the number representing the maximum number of points that will fit in the points buffer. It will be updated to indicate the actual number of points returned in the buffer.

GET VALUATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggetval *(response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
response	Gfloat	O	By reference	Valuator event

INITIALIZE CHOICE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitchoice *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Choice device number
init	Gchoice (structure)	I	By reference	Initial choice and status pointer
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gchoicerec (structure)	I	By reference	Choice data record structure

```
struct Gchoice {                    /*I Choice data*/
     enum Gcstat {                  /*I Choice data - Enumerated type*/
     } status;                      /*I Choice status*/
     Gint choice;                   /*I Choice number*/
} init;

struct Glimit {                     /*I Coordinate limits*/
    Gfloat xmin;                    /*I X minimum limit*/
    Gfloat xmax;                    /*I X maximum limit*/
    Gfloat ymin;                    /*I Y minimum limit*/
    Gfloat ymax;                    /*I Y maximum limit*/
} area;

struct Gchoicerec {                 /*I Choice data record structure*/
    union {
        struct Gchoicepet0001 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gint  *lengths;         /*I Lengths of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet1_datarec;
        struct Gchoicepet0002 {     /*I */
            Gint  number;           /*I Number of alternatives*/
            enum Gprflag *enable;   /*I Array of prompt flags - enumerated type*/
            Gchar *title_string;    /*I The title string */
        } choicepet2_datarec;
        struct Gchoicepet0003 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet3_datarec;
        struct Gchoicepet0004 {     /*I */
            Gint  number;           /*I Number of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet4_datarec;
        struct Gchoicepet0005 {     /*I */
            Gint  seg;              /*I Segment name*/
            Gint  number;           /*I Number of alternatives*/
            Gint  *pickids;         /*I Array of pick indentifiers*/
            Gchar *title_string;    /*I The title string */
        } choicepet5_datarec;
        struct Gchoicepet_0001 {    /*I */
            Gint  number;           /*I Number of choice strings*/
            Gint  *lengths;         /*I Lengths of choice strings*/
            Gchar **strings;        /*I Array of strings*/
            Gchar *title_string;    /*I The title string */
        } choicepet_1_datarec;
    }
} record;
```

Data type	Constant	Description
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Input is NOCHOICE

Data type	Constant	Description
Gprflag	GPROFF	Prompt off
	GPRON	Prompt on

INITIALIZE LOCATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitloc *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Locator device number
init	Gloc (structure)	I	By reference	Initial locator pointer
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Glocrec (structure)	I	By reference	Locator data record structure

```
struct Gloc {                       /*I Locator data*/
     Gint transform;                /*I Normalization transformation number*/
     struct Gpoint {                /*I Coordinate point*/
         Gfloat x                   /*I X coordinate*/
         Gfloat y                   /*I Y coordinate*/
     } position;                    /*I Locator position*/
} init;                             /*I Initial locator pointer*/

struct Glimit {                     /*I Coordinate limits*/
    Gfloat xmin;                    /*I X minimum limit*/
    Gfloat xmax;                    /*I X maximum limit*/
    Gfloat ymin;                    /*I Y minimum limit*/
    Gfloat ymax;                    /*I Y maximum limit*/
} area;                             /*I Echo area structure*/

struct Glocrec {                    /*I Locator data record structure*/
    union {
        struct Glocpet0001 {        /*I */
            Gchar *data;            /*I Device/implementation dependent data*/
        } locpet1_datarec;
        struct Glocpet0002 {        /*I */
            Gchar *data;            /*I Device/implementation dependent data*/
        } locpet2_datarec;
        struct Glocpet0003 {        /*I */
            Gchar *data;            /*I Device/implementation dependent data*/
        } locpet3_datarec;
        struct Glocpet0004 {        /*I */
            Gacf  acf;              /*I Attribute control flag*/
            Glnattr ln;             /*I Polyline attributes*/
        } locpet4_datarec;
        struct Glocpet0005 {        /*I */
            enum Gpfcf pfcf;        /*I Polyline/fill area control flag -
                                    /*  Enumerated type*/
            enum Gacf  acf;         /*I Attribute control flag - Enumerated type*/
            union {
                Glnattr  ln;        /*I Polyline attributes*/
                Gflattr  fl;        /*I Fill area attributes*/
            } attr;
        } locpet5_datarec;
        struct Glocpet0006 {        /*I */
            Gchar *title_string;    /*I The title string */
        } locpet6_datarec;
        struct Glocpet_0001 {       /*I */
            Gfloat box_x;           /*I Size of the box in x */
            Gfloat box_y;           /*I Size of the box in y */
        } locpet_1_datarec;
        struct Glocpet_0002 {       /*I */
            enum Gpfcf pfcf;        /*I Polyline/fill area control flag -
                                    /*  Enumerated type*/
            enum Gacf  acf;         /*I Attribute control flag - Enumerated type*/
            union {
                Glnattr  ln;        /*I Polyline attributes*/
                Gflattr  fl;        /*I Fill area attributes*/
```

```
    } attr;
} locpet_2_datarec;
struct Glocpet_0003 {     /*I */
    Gacf  acf;            /*I Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*I Point 1 for echo */
          Gpoint  point2; /*I Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /* Polyline attributes */
          Gpoint point1;  /* Point 1 for echo */
          Gpoint point2;  /* Point 2 for echo */
       } lnecho;
    } attr;
} locpet_3_datarec;
struct Glocpet_0004 {    /*I */
    Gacf  acf;           /*I Attribute control flag*/
    Glnattr ln;          /*I Polyline attributes*/
} locpet_4_datarec;
struct Glocpet_0005 {    /*I */
    Gacf  acf;           /*I Attribute control flag*/
    Glnattr ln;          /*I Polyline attributes*/
} locpet_5_datarec;
struct Glocpet_0006 {     /*I */
    Gacf  acf;            /*I Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*I Point 1 for echo */
          Gpoint  point2; /*I Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /* Polyline attributes */
          Gpoint point1;  /* Point 1 for echo */
          Gpoint point2;  /* Point 2 for echo */
       } lnecho;
    } attr;
} locpet_6_datarec;
struct Glocpet_0007 {     /*I */
    Gacf  acf;            /*I Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*I Point 1 for echo */
          Gpoint  point2; /*I Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /* Polyline attributes */
          Gpoint point1;  /* Point 1 for echo */
          Gpoint point2;  /* Point 2 for echo */
       } lnecho;
    } attr;
} locpet_7_datarec;
struct Glocpet_0008 {     /*I */
    Gacf  acf;            /*I Attribute control flag */
```

```
            union {
               struct {
                  Gpoint  point1; /*I Point 1 for echo */
                  Gpoint  point2; /*I Point 2 for echo */
               } echo;
                struct {
                  Glnattr ln;     /* Polyline attributes */
                  Gpoint point1;  /* Point 1 for echo */
                  Gpoint point2;  /* Point 2 for echo */
               } lnecho;
            } attr;
        } locpet_8_datarec;
        struct Glocpet_0009 {    /*I */
            Gacf  acf;           /*I Attribute control flag*/
            Glnattr ln;          /*I Polyline attributes*/
        } locpet_9_datarec;
        struct Glocpet_0010 {    /*I */
            Gacf  acf;           /*I Attribute control flag*/
            Glnattr ln;          /*I Polyline attributes*/
        } locpet_10_datarec;
        struct Glocpet_0011 {   /*I */
            Gchar *data;        /*I Device/implementation dependent data*/
        } locpet_11_datarec;
        struct Glocpet_0012 {    /*I */
            Gacf  acf;           /*I Attribute control flag*/
            Glnattr ln;          /*I Polyline attributes*/
        } locpet_12_datarec;
    }
} record;                       /*I Locator data record*/

struct Glnattr {                /*I Polyline attributes*/
     enum Gasf {                /*I Aspect control flag - Enumerated type*/
     } type|width|color;        /*I Linetype ASF|linewidth ASF|linecolour ASF*/
     Gint line;                 /*I Line index*/
     struct Glnbundl {          /*I Polyline bundle*/
         Gint type;             /*I Line type*/
         Gfloat width;          /*I Linewidth scale factor*/
         Gint colour;           /*I Polyline colour index*/
     } bundl;                   /*I Line bundle*/
} ln;                           /*I Polyline attributes*/

struct Gflattr {                /*I Fill area attributes*/
     enum Gasf {                /*I Aspect control flag - Enumerated type*/
     } inter|style|color;       /*I Fill interior ASF|fill style ASF|linecolour ASF*/
     Gint fill;                 /*I Fill color*/
     struct Gflbundl {          /*I Fill area bundle*/
         enum Gflinter;         /*I Fill area interior*/
         Gint style;            /*I Fill area style*/
         Gint colour;           /*I Fill area colour index*/
     } bundl;                   /*I Fill area bundle*/
} fl;                           /*I Fill area attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

Data type	Constant	Description
Gpfcf	GPF_POLYLINE	Polyline
	GPF_FILLAREA	Fill area

Data type	Constant	Description
Gflinter	GHOLLOW	Hollow
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Hatch

INITIALIZE PICK

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitpick *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Pick device number
init	Gpick (structure)	I	By reference	Initial pick pointer
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gpickrec (structure)	I	By reference	Pick data record structure

```
struct Gpick {                          /*I Pick data*/
     enum Gpstat {                      /*I Pick status - Enumerated type*/
     } status;                          /*I Pick status*/
     Gint seg;                          /*I Pick segment*/
     Gint pickid;                       /*I Pick number*/
} init;

struct Glimit {                         /*I Coordinate limits*/
    Gfloat xmin;                        /*I X minimum limit*/
    Gfloat xmax;                        /*I X maximum limit*/
    Gfloat ymin;                        /*I Y minimum limit*/
    Gfloat ymax;                        /*I Y maximum limit*/
} area;

struct Gpickrec {                       /*I Pick data record structure*/
    union {
        struct Gpickpet0001  {          /*I */
            Gfloat aperture;            /*I The pick aperture size */
        } pickpet1_datarec;
        struct Gpickpet0002 {           /*I */
            Gfloat aperture;            /*I The pick aperture size */
        } pickpet2_datarec;
        struct Gpickpet0003  {          /*I */
            Gfloat aperture;            /*I The pick aperture size */
        } pickpet3_datarec;
    }
} record;
```

Data type	Constant	Description
Gpstat	GP_OK	Input obtained
	GP_NOPICK	Input is NOPICK

INITIALIZE STRING

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitstring *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	String device number
init	Gchar	I	By reference	Initial string
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gstringrec (structure)	I	By reference	String data record structure

```
struct Glimit {                         /*I Coordinate limits*/
    Gfloat xmin;                        /*I X minimum limit*/
    Gfloat xmax;                        /*I X maximum limit*/
    Gfloat ymin;                        /*I Y minimum limit*/
    Gfloat ymax;                        /*I Y maximum limit*/
} area;

struct Gstringrec {                     /*I String data record structure*/
    union {
        struct Gstringpet0001 {         /*I */
            Gint  bufsiz;               /*I Buffer size*/
            Gint  position;             /*I Initial cursor position*/
            Gchar  *title_string;       /*I The title string */
        } stringpet1_datarec;
    }
} record;
```

INITIALIZE STROKE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitstroke *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Stroke device number
init	Gstroke (structure)	I	By reference	Initial stroke and status pointer
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gstrokerec (structure)	I	By reference	Stroke data record structure

```
struct Gstroke {                    /*I Stroke data*/
     Gint transform;                /*I Normalization transformation number*/
     Gint n_points;                 /*I Number of points in stroke*/
     struct Gpoint {                /*I Points in stroke*/
         Gfloat x;                  /*I X coordinate*/
         Gfloat y;                  /*I Y coordinate*/
     } *points;
} response;

struct Glimit {                      /*I Coordinate limits*/
    Gfloat xmin;                     /*I X minimum limit*/
    Gfloat xmax;                     /*I X maximum limit*/
    Gfloat ymin;                     /*I Y minimum limit*/
    Gfloat ymax;                     /*I Y maximum limit*/
} area;

struct Gstrokerec {                  /*I Stroke data record structure*/
    union {
        struct Gstrokepet0001 {      /*I */
            Gint  bufsiz;            /*I Input buffer size*/
            Gint  editpos;           /*I Editing position*/
            Gpoint interval;         /*I X, Y interval*/
            Gfloat time;             /*I Time interval*/
        } strokepet1_datarec;
        struct Gstrokepet0002  {     /*I */
            Gint  bufsiz;            /*I Input buffer size*/
            Gint  editpos;           /*I Editing position*/
            Gpoint interval;         /*I X, Y interval*/
            Gfloat time;             /*I Time interval*/
        } strokepet2_datarec;
        struct Gstrokepet0003  {     /*I */
            Gint  bufsiz;            /*I Input buffer size*/
            Gint  editpos;           /*I Editing position*/
            Gpoint interval;         /*I X, Y interval*/
            Gfloat time;             /*I Time interval*/
            enum Gacf acf;           /*I Attribute control flag _ Enumerated type*/
            Gmkattr mk;              /*I Marker attributes*/
        } strokepet3_datarec;
        struct Gstrokepet0004  {     /*I */
            Gint  bufsiz;            /*I Input buffer size*/
            Gint  editpos;           /*I Editing position*/
            Gpoint interval;         /*I X, Y interval*/
            Gfloat time;             /*I Time interval*/
            enum Gacf acf;           /*I Attribute control flag _ Enumerated type*/
            Glnattr ln;              /*I Line attributes*/
        } strokepet4_datarec;
    }
} record;

struct Gpoint {                      /*I Coordinate point*/
    Gfloat x;                        /*I X coordinate*/
    Gfloat y;                        /*I Y coordinate*/
} points;
```

```
struct Gmkattr {                    /*I Polymarker attributes*/
     enum Gasf {                    /*I Aspect control flag - Enumerated type*/
     } type|size|color;             /*I Markertype ASF|markerwidth ASF|markercolor ASF*/
     Gint marker;                   /*I Marker index*/
     struct Gmkbundl {              /*I Polymarker bundle*/
         Gint type;                 /*I Marker type*/
         Gfloat width               /*I Markerwidth scale factor*/
         Gint colour;               /*I Polymarker colour index*/
     } bundl;                       /*I Marker bundle*/
} mk;                               /*I Polymarker attributes*/

struct Glnattr {                    /*I Polyline attributes*/
     enum Gasf {                    /*I Aspect control flag - Enumerated type*/
     } type|width|color;            /*I Linetype ASF|linewidth ASF|linecolor ASF*/
     Gint line;                     /*I Line index*/
     struct Glnbundl {              /*I Polyline bundle*/
         Gint type;                 /*I Line type*/
         Gfloat width               /*I Linewidth scale factor*/
         Gint colour;               /*I Polyline colour index*/
     } bundl;                       /*I Line bundle*/
} ln;                               /*I Polyline attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INITIALIZE VALUATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginitval *(workstation_id, device_number, init, pet, area, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Valuator device number
init	Gfloat	I	By value	Initial value
pet	Gint	I	By value	Prompt and echo type
area	Glimit (structure)	I	By reference	Echo area structure
record	Gvalrec (structure)	I	By reference	Valuator data record structure

```
struct Glimit {                         /*I Coordinate limits*/
    Gfloat xmin;                        /*I X minimum limit*/
    Gfloat xmax;                        /*I X maximum limit*/
    Gfloat ymin;                        /*I Y minimum limit*/
    Gfloat ymax;                        /*I Y maximum limit*/
} area;

struct Gvalrec {                        /*I Valuator data record structure*/
    union {
        struct Gvalpet0001 {            /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet1_datarec;
        struct Gvalpet0002 {            /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet2_datarec;
        struct Gvalpet0003  {           /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet3_datarec;
        struct Gvalpet_0001  {          /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet_1_datarec;
        struct Gvalpet_0002  {          /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet_2_datarec;
        struct Gvalpet_0003  {          /*I */
            Gfloat low;                 /*I Low range limit*/
            Gfloat high;                /*I High range limit*/
            Gchar  *title_string;       /*I The title string */
        } valpet_3_datarec;
    }
} record;
```

REQUEST CHOICE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqchoice *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Choice device number
response	Gqchoice (structure)	O	By reference	Choice response

```
struct Gqchoice {                    /*O Choice data*/
     enum Gcstat {                   /*O Choice data - Enumerated type*/
     } status;                       /*O Request status*/
     Gint choice;                    /*O Choice data*/
} response;
```

Data type	Constant	Description
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Input is NOCHOICE

REQUEST LOCATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqloc *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Locator device number
response	Gqloc (structure)	O	By reference	Locator response

```
struct Gqloc {                      /*O Request locator*/
     enum Gistat {                  /*O Locator data - Enumerated type*/
     } status;                      /*O Request status*/
     struct Gloc {                  /*O Locator data*/
         Gint transform;            /*O Normalization transformation number*/
         Gpoint position;           /*O Locator position/*
         } loc;                     /*O */
} response;
```

Data type	Constant	Description
Gistat	GOK	Request
	GNONE	No request

```
struct Gpoint {                     /*O Coordinate point*/
    Gfloat x;                       /*O X coordinate*/
    Gfloat y;                       /*O Y coordinate*/
} points;
```

REQUEST PICK

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqpick *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Pick device number
response	Gqpick (structure)	O	By reference	Pick response

```
struct Gqpick {                    /*O Request pick*/
     enum Gpstat {                 /*O Pick data - Enumerated type*/
     } status;                     /*O Pick status*/
     Gint seg;                     /*O Segment*/
     Gint pickid;                  /*O Pick identifier*/
} response;
```

Data type	Constant	Description
Gpstat	GP_OK	Input obtained
	GP_NOPICK	Input is NOPICK
	GP_NONE	No input obtained

REQUEST STRING

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqstring *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	String device number
response	Gqstring (structure)	O	By reference	String response

```
struct Gqstring {                    /*O Request string*/
     enum Gistat {                   /*O Request status - Enumerated type*/
     } status;                       /*O Request status*/
     Gchar *string;                  /*M String data*/
} response;
```

Data type	Constant	Description
Gistat	GOK	Request
	GNONE	No request

NOTE

Initialize the string parameter with a string, before the call. GKS will do a strlen to get the size, and will use this size as the maximum size of the returned string.

REQUEST STROKE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqstroke *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Stroke device number
response	Gqstroke (structure)	O	By reference	Stroke response

```
struct Gqstroke {                   /*O Stroke data*/
     enum Gistat {                  /*O Request status - Enumerated type*/
     } status;                      /*O Request status*/
     Gstroke stroke;                /*O Stroke data*/
} response;

struct Gstroke {                    /*O Stroke data*/
     Gint transform;                /*O Normalization transformation number*/
     Gint n_points;                 /*M Number of points in stroke*/
     struct Gpoint {                /*O Points in stroke (user supplied stroke
                                        buffer)*/
         Gfloat x;                  /*O X coordinate*/
         Gfloat y;                  /*O Y coordinate*/
     } *points;
} response;
```

Data type	Constant	Description
Gistat	GOK	Request
	GNONE	No request

NOTE

N_points should be initialized with the number representing the maximum number of points that will fit in the points buffer. It will be updated to indicate the actual number of points returned in the buffer.

REQUEST VALUATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

greqval *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Valuator device number
response	Gqval (structure)	O	By reference	Valuator response

```
struct Gqval {                    /*O Valuator data*/
     enum Gistat {                /*O Request status - Enumerated type*/
     } status;                    /*O Request status*/
     Gfloat val;                  /*O Valuator data*/
} response;
```

Data type	Constant	Description
Gistat	GOK	Request
	GNONE	No request

SAMPLE CHOICE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsamplechoice *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Choice device number
response	Gchoice (structure)	O	By reference	Choice sample

```
struct Gchoice {                    /*O Choice data*/
     enum Gcstat {                  /*O Choice data - Enumerated type*/
     } status;                      /*O Choice status*/
     Gint choice;                   /*O Choice number*/
} response;
```

Data type	Constant	Description
Gcstat	GC_OK	Input obtained
	GC_NOCHOICE	Input is NOCHOICE
	GC_NONE	No input obtained

SAMPLE LOCATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsampleloc *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Locator device number
response	Gloc (structure)	O	By reference	Locator sample

```
struct Gloc {                    /*O Locator data*/
     Gint transform;             /*O Normalization transformation number*/
     struct Gpoint {             /*O Coordinate point*/
          Gfloat x;              /*O X coordinate*/
          Gfloat y;              /*O Y coordinate*/
     } position;                 /*O Locator position*/
} response;                      /*O Initial locator pointer*/
```

SAMPLE PICK

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsamplepick *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Pick device number
response	Gpick (structure)	O	By reference	Pick sample

```
struct Gpick {                          /*O Pick data*/
     enum Gpstat {                      /*O Pick status - Enumerated type*/
     } status;                          /*O Pick status*/
     Gint seg;                          /*O Pick segment*/
     Gint pickid;                       /*O Pick number*/
} init;
```

Data type	Constant	Description
Gpstat	GP_OK	Input obtained
	GP_NOPICK	Input is NOPICK
	GP_NONE	No input obtained

SAMPLE STRING

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsamplestring *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Sample device number
response	Gchar	M	By reference	String sample

NOTE

Initialize the response parameter with a string, before the call. GKS will do a strlen to get the size, and will use this size as the maximum size of the returned string.

SAMPLE STROKE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsamplestroke *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Stroke device number
response	Gstroke (structure)	O	By reference	Stroke sample

```
struct Gstroke {                    /*O Stroke data*/
     Gint transform;                /*O Normalization transformation number*/
     Gint n_points;                 /*M Points in a stroke*/
     struct Gpoint {                /*O Locator position*/
         Gfloat x;                  /*O X coordinate*/
         Gfloat y;                  /*O Y coordinate*/
     } *points;                     /*O Request status*/
} response;
```

NOTE

N_points should be initialized with the number representing the maximum number of points that will fit in the points buffer. It will be updated to indicate the actual number of points returned in the buffer.

SAMPLE VALUATOR

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsampleval *(workstation_id, device_number, response)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Valuator device number
response	Gfloat	O	By reference	Valuator sample

SET CHOICE MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetchoicemode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Choice device number
operating_ mode	Gimode (enu- merated)	I	By value	Operating mode
echo	Gesw (enu- merated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

SET LOCATOR MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetlocmode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Locator device number
operating_ mode	Gimode (enu- merated)	I	By value	Operating mode
echo	Gesw (enu- merated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

SET PICK MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetpickmode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Pick device number
operating_ mode	Gimode (enu- merated)	I	By value	Operating mode
echo	Gesw (enu- merated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

SET STRING MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetstringmode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	String device number
operating_ mode	Gimode (enu- merated)	I	By value	Operating mode
echo	Gesw (enu- merated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

SET STROKE MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetstrokemode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Stroke device number
operating_ mode	Gimode (enumerated)	I	By value	Operating mode
echo	Gesw (enumerated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

SET VALUATOR MODE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetvalmode *(workstation_id, device_number, operating_mode, echo)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_number	Gint	I	By value	Valuator device number
operating_ mode	Gimode (enumerated)	I	By value	Operating mode
echo	Gesw (enumerated)	I	By value	Echo switch

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	Noecho

Chapter 9

Segment Functions

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

Category	GKS Functions
Using Segments	gassocsegws, gcloseseg, gcopysegws, gcreateseg, gdelseg, gdelsegws, ginsertseg, grenameseg
Segment Attributes	gsetdet, gsethighlight, gsetsegpri, gsetvis
Segment Transformations	gaccumtran, gevaltran, gsetsegtran

ASSOCIATE SEGMENT WITH WORKSTATION

Operating States: GWSOP, GWSAC,

Syntax

gassocsegws *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
segment_name	Gint	I	By value	Segment name

CLOSE SEGMENT

Operating States: GSGOP

Syntax

gcloseseg *( )*

COPY SEGMENT TO WORKSTATION

Operating States: GWSOP, GWSAC

Syntax

gcopysegws *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
segment_name	Gint	I	By value	Segment name

CREATE SEGMENT

Operating States: GWSAC

Syntax

gcreateseg *(segment_name)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By value	Segment name

DELETE SEGMENT

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gdelseg *(segment_name)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_name	Gint	I	By value	Segment name

DELETE SEGMENT FROM WORKSTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gdelsegws *(workstation_id, segment_name)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
segment_name	Gint	I	By value	Segment name

INSERT SEGMENT

Operating States: GWSAC, GSGOP

Syntax

ginsertseg *(segment_name, segtran)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By value	Segment name
segtran[2][3]	Gfloat	I	By value	2 x 3 transformation matrix

RENAME SEGMENT

Operating States: GWSOP, GWSAC, GSGOP

Syntax

grenameseg *(old, new)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
old	Gint	I	By value	Segment's previous name
new	Gint	I	By value	Segment's new name

SET SEGMENT DETECTABILITY

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetdet *(segment_name, detectability)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_name	Gint	I	By value	Segment name
detectability	Gsegdet (enumerated)	I	By value	Detectability

Data type	Constant	Description
Gsegdet	GUNDETECTABLE	Cannot pick the segment.
	GDETECTABLE	Pick the segment, if visible.

SET SEGMENT HIGHLIGHTING

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsethighlight *(segment_name, highlighting)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By value	Segment name
highlighting	Gseghi (enumerated)	I	By value	Highlighting

Data type	Constant	Description
Gseghi	GNORMAL	DEC GKS does not highlight the segment.
	GHIGHLIGHTED	DEC GKS highlights the segment, if visible.

SET SEGMENT PRIORITY

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetsegpri *(segment_name, priority)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_name	Gint	I	By value	Segment name
priority	Gfloat	I	By value	Segment priority, infinite range [0.0,1.1]

SET SEGMENT TRANSFORMATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetsegtran *(segment_name, segtran)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_name	Gint	I	By value	Segment name
segtran[2][3]	Gfloat	I	By value	2 x 3 transformation matrix

SET SEGMENT VISIBILITY

Operating States: GWSOP, GWSAC, GSGOP

Syntax

gsetvis *(segment_name, visibility)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By value	Segment name
visibility	Gsegvis (enu- merated)	I	By value	Segment visibility

Data type	Constant	Description
Gsegvis	GVISIBLE	DEC GKS does not show the segment.
	GINVISIBLE	DEC GKS shows the segment on the workstation surface.

Chapter 10

Metafile, Error, and Utility Functions

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

- ggettypegksm
- ginterpret
- greadgksm
- gwritegksm

The DEC GKS error-handling functions provide a method for you to control the generation of messages to the user, and a method of exit when a DEC GKS function call generates an error. The following list presents the DEC GKS error-handling functions:

- gemergencyclosegks
- gerrorhand
- gerrorlog

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

- gevaltran
- gaccumtran

GET ITEM TYPE FROM GKSM

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ggettypegksm *(workstation_id, result)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
result	Ggksmit (structure)	O	By reference	Metafile item information

```
struct Ggksmit {                    /*O GKS metafile item*/
    Gint type;                      /*O Item type*/
    Gint length;                    /*O Item length*/
} result;
```

INTERPRET ITEM

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginterpret *(typeandlength, date)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
typeandlength	Ggksmit (structure)	I	By reference	Item type and length pointer
data	Ggksmrec (structure)	I	By reference	Item data record pointer

```
struct Ggksmit {                    /*I GKS metafile item*/
    Gint type;                      /*I Item type*/
    Gint length;                    /*I Item length*/
} typeandlength;

struct Ggksmrec {                   /*I GKS metafile data record*/
    Gchar *gksmrec;                 /*I Metafile data record*/
} data;
```

READ ITEM FROM GKSM

Operating States: GWSOP, GWSAC, GSGOP

Syntax

readgksm *(workstation_id, length, record)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
length	Gint	I	By value	Maximum item data record length
record	Ggksmrec (structure)	O	By reference	Metafile Item data record

```
struct Ggksmrec {                    /*O GKS metafile data record*/
    Gchar *gksmrec;                  /*O Metafile data record*/
} record;
```

WRITE ITEM TO GKSM

Operating States: GWSAC, GSGOP

Syntax

gwritegksm *(workstation_id, type, length, data)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
type	Gint	I	By value	Item type
length	Gint	I	By value	Item length
data	Ggksmrec (structure)	I	By reference	Data record pointer

```
struct Ggksmrec {                    /*I GKS metafile data record*/
    Gchar *gksmrec;                  /*I Metafile data record*/
} data;
```

EMERGENCY CLOSE GKS

Operating States: GGKCL, GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gemergencyclosegks *( )*

ERROR HANDLING

Operating States: GGKCL, GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gerrorhand *(error-number, funcname, perrfile)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
errnum	Gint	I	By value	Error number
funcname	Gint	I	By value	GKS function number
perrfile	Gfile	I	By reference	Error file pointer

ERROR LOGGING

Operating States: GGKCL, GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gerrorlog *(error_number, funcname, perrfile)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
errnum	Gint	I	By value	Error number
funcname	Gint	I	By value	GKS function number
perrfile	Gfile	I	By reference	Error file pointer

EVALUATE TRANSFORMATION MATRIX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gevaltran *(ppoint, pshift, angle, pscale, coord, result)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
ppoint	Gpoint (structure)	I	By reference	Fixed point pointer
pshift	Gpoint (structure)	I	By reference	Shift vector pointer
angle	Gfloat	I	By value	Rotation in radians
pscale	Gscale (structure)	I	By reference	Scale factors pointer
coord	Gcsw (enumerated)	I	By value	Coordinate switch
result[2][3]	Gfloat	O	By value	2 x 3 transformation matrix

```
struct Gpoint {                    /*I Coordinate point*/
    Gfloat x;                      /*I X coordinate*/
    Gfloat y;                      /*I Y coordinate*/
} ppoint;

struct Gpoint {                    /*I Coordinate point*/
    Gfloat x;                      /*I X coordinate*/
    Gfloat y;                      /*I Y coordinate*/
} pshift;

struct Gscale {                    /*I Scale vector*/
    Gfloat x_scale;                /*I X scale*/
    Gfloat y_scale;                /*I Y scale*/
} pscale;
```

Data type	Constant	Description
Gcsw	GWC	The fixed point and translation vectors are world coordinate values.
	GNDC	The fixed point and translation vectors are normalized device coordinate values.

ACCUMULATE TRANSFORMATION MATRIX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

gaccumtran *(segtran, ppoint, pshift, angle, pscale, coord, result)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segtran	Gfloat	I	By value	2 x 3 segment transformation matrix
ppoint	Gpoint (structure)	I	By reference	Fixed point pointer
pshift	Gpoint (structure)	I	By reference	Shift vector pointer
angle	Gfloat	I	By value	Rotation in radians
pscale	Gscale (structure)	I	By reference	Scale factors pointer
coord	Gcsw (enumerated)	I	By value	Coordinate switch
result	Gfloat	O	By value	2 x 3 transformation matrix

```
struct Gpoint {                     /*I Coordinate point*/
    Gfloat x;                       /*I X coordinate*/
    Gfloat y;                       /*I Y coordinate*/
} ppoint;

struct Gpoint {                     /*I Coordinate point*/
    Gfloat x;                       /*I X coordinate*/
    Gfloat y;                       /*I Y coordinate*/
} pshift;

struct Gscale {                     /*I Scale vector*/
    Gfloat x_scale;                 /*I X scale*/
    Gfloat y_scale;                 /*I Y scale*/
} pscale;
```

Data type	Constant	Description
Gcsw	GWC	The fixed point and translation vectors are world coordinate values.
	GNDC	The fixed point and translation vectors are normalized device coordinate values.

Chapter 11

Inquiry Functions

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

INQUIRE OPERATING STATE VALUE

Operating States: GGKCL, GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqopst *(state)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
state	Gopst (enumerated)	O	By reference	GKS operating state

Data type	Constant	Description
Gopst	GGKCL	GKS closed
	GGKOP	GKS open
	GWSOP	Workstation Open
	GWSAC	Workstation Active
	GSGOP	Segment open

INQUIRE LEVEL OF GKS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqlevelgks *(level, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
level	Glevel (enumerated)	O	By reference	Level of GKS
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Glevel	GLMA	Minimal output, No input
	GLMB	Minimal output, Request input
	GLMC	Minimal output, Full input
	GL0A	All primitives and attributes, No input
	GL0B	All primitives and attributes, Request input
	GL0C	All primitives and attributes, Full input
	GL1A	Basic segmentation with full output, No input
	GL1B	Basic segmentation with full output, Request input
	GL1C	Basic segmentation with full output, Full input

GKS Description Table Inquiry Functions
INQUIRE LEVEL OF GKS

Data type	Constant	Description
	GL2A	Workstation Independent & Segment Storage, No input
	GL2B	Workstation Independent & Segment Storage, Request input
	GL2C	Workstation Independent & Segment Storage, Full input

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
bufsize	Gint	I	By value	Maximum number of workstation types that fit into buffer wstypes
start	Gint	I	By value	Start position of inquiry
wstypes	Gstrlist (structure)	O	By reference	List of available workstation types
actual_types	Gint	O	By reference	Number of workstation types available
error_status	Gint	O	By reference	Error status

```
struct Gstrlist {                    /*O Workstation type buffer*/
     Gint n_points;                  /*O Number of workstation types*/
     Gchar **strings;                /*O Array of pointers to workstation type
                                         strings*/
} wstypes;
```

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmaxntrannum *(maxtran, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
maxtran	Gint	O	By reference	Maximum normalization transformation number
error_status	Gint	O	By reference	Error status

INQUIRE WORKSTATION MAXIMUM NUMBERS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqwsmaxnum *(maxws, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
maxws	Gwsmax (structure)	O	By reference	Workstation maximum numbers
error_status	Gint	O	By reference	Error status

```
struct Gwsmax {                    /*O Workstation maximum numbers*/
     Gint open;                    /*O Number of open workstations*/
     Gint active;                  /*O Number of active workstations*/
     Gint assoc;                   /*O Number of associated
                                       workstations*/
} maxws;
```

INQUIRE COLOUR FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Buffer size of buffer facility
fac_size	Gint	O	By reference	Size of color facilities structure (unused)
fac	Gcofac (structure)	O	By reference	Color facilities structure
error_status	Gint	O	By reference	Error status

```
struct Gcofac {                    /*O Color facilities*/
    Gint colours;                  /*O Number of colors*/
    enum Gcoavail coavail;         /*O Color availability -
                                       Enumerated type*/
    Gint predefined;               /*O Number of predefined bundles*/
} fac;
```

Data type	Constant	Description
Gcoavail	GCOLOUR	Color
	GMONOCHROME	Monochrome

INQUIRE DEFAULT CHOICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit into buffer pets of data
data_size	Gint	M	By reference	Size of the choice data record (size of the structure record of parameter data)
data	Gdefchoice (structure)	O	By reference	Default choice device data structure
error_status	Gint	O	By reference	Error status

Workstation Description Table Inquiry Functions
INQUIRE DEFAULT CHOICE DATA

```
struct Gdefchoice {                 /*O Default choice data*/
    Gint choices;                   /*O Maximum number of choices*/
    Gintlist *pets;                 /*O List of prompt and echo types*/
    Glimit e_area;                  /*O Default echo area*/
    Gchoicerec record;              /*O Default choice data record*/
} data;

struct Gintlist {                   /*O Prompt and echo type buffer*/
    Gint number;                    /*O Prompt and echo types*/
    Gint *integers;                 /*O List of prompt and echo types
                                        (in user supplied integer array)*/
} pets;

struct Glimit {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gchoicerec {                 /*O Choice data record structure*/
    union {
        struct Gchoicepet0001 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gint  *lengths;         /*O Lengths of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet1_datarec;
        struct Gchoicepet0002 {     /*O */
            Gint  number;           /*O Number of alternatives*/
            enum Gprflag *enable;   /*O Array of prompt flags - enumerated type*/
            Gchar *title_string;    /*O The title string */
        } choicepet2_datarec;
        struct Gchoicepet0003 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet3_datarec;
        struct Gchoicepet0004 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet4_datarec;
        struct Gchoicepet0005 {     /*O */
            Gint  seg;              /*O Segment name*/
            Gint  number;           /*O Number of alternatives*/
            Gint  *pickids;         /*O Array of pick indentifiers*/
            Gchar *title_string;    /*O The title string */
```

```
        } choicepet5_datarec;
        struct Gchoicepet_0001 {  /*O */
            Gint  number;         /*O Number of choice strings*/
            Gint  *lengths;       /*O Lengths of choice strings*/
            Gchar **strings;      /*O Array of strings*/
            Gchar *title_string;  /*O The title string */
        } choicepet_1_datarec;
    }
} record;
```

NOTE

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

INQUIRE DEFAULT DEFERRAL STATE VALUES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqdefdeferst *(workstation_type, def, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
def	Gdefer	O	By reference	Default deferral state values structure
error_status	Gint	O	By reference	Error status

```
struct Gdefer {                    /*O Deferral state*/
    Gdefmode defmode;              /*O Deferral mode*/
    Girgmode irgmode;              /*O Implicit regeneration mode*/
} def;
```

Data type	Constant	Description
Gdefmode	GASAP	As soon as possible
	GBNIG	Before next interaction globally
	GBNIL	Before next interaction locally
	GASTI	At some time

Data type	Constant	Description
Girgmode	GSUPPRESSED	Suppressed
	GALLOWED	Allowed

INQUIRE DEFAULT LOCATOR DEVICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Locator device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit in buffer pets of parameter data
data_size	Gint	M	By reference	Size of the locator data record (size of the structure record of parameter data)
data	Gdefloc (structure)	O	By reference	Default locator device data structure
error_status	Gint	O	By reference	Error status

```
struct Gdefloc {                    /*O Default locator data*/
    Gpoint position;                /*O Number of prompt and echo types*/
    Gintlist *pets;                 /*O List of prompt and echo types*/
    Glimit e_area;                  /*O Default echo area*/
    Glocrec record;                 /*O Default locator data record*/
} data;

struct Gpoint {                     /*O Coordinate point*/
    Gfloat x;                       /*O X coordinate*/
    Gfloat y;                       /*O Y coordinate*/
} position;

struct Gintlist {                   /*O User supplied prompt and echo type buffer*/
    Gint number;                    /*O Number of prompt and echo types*/
    Gint *integers;                 /*O List of prompt and echo types (in user
                                        supplied integer array)*/
} pets;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Glocrec {                    /*O Locator data record structure*/
    union {
        struct Glocpet0001 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet1_datarec;
        struct Glocpet0002 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet2_datarec;
        struct Glocpet0003 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet3_datarec;
        struct Glocpet0004 {        /*O */
            Gacf  acf;              /*O Attribute control flag*/
            Glnattr ln;             /*O Polyline attributes*/
        } locpet4_datarec;
        struct Glocpet0005 {        /*O */
            enum Gpfcf pfcf;        /*O Polyline/fill area control flag -
                                    /*O Enumerated type*/
            enum Gacf  acf;         /*O Attribute control flag _ Enumerated type*/
            union {
                Glnattr  ln;        /*O Polyline attributes*/
                Gflattr  fl;        /*O Fill area attributes*/
            } attr;
        } locpet5_datarec;
        struct Glocpet0006 {        /*O */
            Gchar *title_string;    /*O The title string */
        } locpet6_datarec;
        struct Glocpet_0001 {       /*O */
```

```
    Gfloat box_x;          /*O Size of the box in x */
    Gfloat box_y;          /*O Size of the box in y */
} locpet_1_datarec;
struct Glocpet_0002 {      /*O */
    enum Gpfcf pfcf;       /*O Polyline/fill area control flag -
                           /*  Enumerated type*/
    enum Gacf  acf;        /*O Attribute control flag _ Enumerated type*/
    union {
        Glnattr  ln;       /*O Polyline attributes*/
        Gflattr  fl;       /*O Fill area attributes*/
    } attr;
} locpet_2_datarec;
struct Glocpet_0003 {      /*O */
    Gacf  acf;             /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*O Point 1 for echo */
          Gpoint  point2; /*O Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;      /*O Polyline attributes */
          Gpoint point1;   /*O Point 1 for echo */
          Gpoint point2;   /*O Point 2 for echo */
       } lnecho;
    } attr;
} locpet_3_datarec;
struct Glocpet_0004 {      /*O */
    Gacf  acf;             /*O Attribute control flag*/
    Glnattr ln;            /*O Polyline attributes*/
} locpet_4_datarec;
struct Glocpet_0005 {      /*O */
    Gacf  acf;             /*O Attribute control flag*/
    Glnattr ln;            /*O Polyline attributes*/
} locpet_5_datarec;
struct Glocpet_0006 {      /*O */
    Gacf  acf;             /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*O Point 1 for echo */
          Gpoint  point2; /*O Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;      /*O Polyline attributes */
          Gpoint point1;   /*O Point 1 for echo */
          Gpoint point2;   /*O Point 2 for echo */
       } lnecho;
    } attr;
} locpet_6_datarec;
struct Glocpet_0007 {      /*O */
    Gacf  acf;             /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1; /*O Point 1 for echo */
```

```
                Gpoint  point2; /*O Point 2 for echo */
            } echo;
             struct {
                Glnattr ln;     /*O Polyline attributes */
                Gpoint point1;  /*O Point 1 for echo */
                Gpoint point2;  /*O Point 2 for echo */
            } lnecho;
        } attr;
    } locpet_7_datarec;
    struct Glocpet_0008 {       /*O */
        Gacf  acf;              /*O Attribute control flag */
        union {
            struct {
                Gpoint  point1; /*O Point 1 for echo */
                Gpoint  point2; /*O Point 2 for echo */
            } echo;
             struct {
                Glnattr ln;     /*O Polyline attributes */
                Gpoint point1;  /*O Point 1 for echo */
                Gpoint point2;  /*O Point 2 for echo */
            } lnecho;
        } attr;
    } locpet_8_datarec;
    struct Glocpet_0009 {       /*O */
        Gacf  acf;              /*O Attribute control flag*/
        Glnattr ln;             /*O Polyline attributes*/
    } locpet_9_datarec;
    struct Glocpet_0010 {       /*O */
        Gacf  acf;              /*O Attribute control flag*/
        Glnattr ln;             /*O Polyline attributes*/
    } locpet_10_datarec;
    struct Glocpet_0011 {       /*O */
        Gchar *data;            /*O Device/implementation dependent data*/
    } locpet_11_datarec;
    struct Glocpet_0012 {       /*O */
        Gacf  acf;              /*O Attribute control flag*/
        Glnattr ln;             /*O Polyline attributes*/
    } locpet_12_datarec;
  }
} record;                       /*O Locator data record*/

struct Glnattr {                /*O Polyline attributes*/
    enum Gasf {                 /*O Aspect control flag -
                                    Enumerated type*/
    } type|width|color;         /*O Linetype ASF|linewidth
                                    ASF|linecolor ASF*/
    Gint line;                  /*O Line index*/
    struct Glnbundl {           /*O Aspect control flag*/
        Gint type;              /*O Line type*/
        Gfloat width;           /*O Linewidth scale factor*/
        Gint colour;            /*O Polyline colour index*/
    } bundl;                    /*O Line bundle*/
} ln;                           /*O Polyline attributes*/
```

```
struct Gflattr {                    /*I Fill area attributes*/
     enum Gasf {                    /*I Aspect control flag - Enumerated type*/
     } inter|style|color;           /*I Fill interior ASF|fill style ASF|linecolor ASF*/
     Gint fill;                     /*I Fill color*/
     struct Gflbundl {              /*I Fill area bundle*/
         Gint inter;                /*I Fill area interior*/
         Gint style;                /*I Fill area style*/
         Gint colour;               /*I Fill area colour index*/
     } bundl;                       /*I Fill area bundle*/
} fl;                               /*I Fill area attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gpfcf	GPF_POLYLINE	Polyline
	GPF_FILLAREA	Fill area

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INQUIRE DEFAULT PICK DEVICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit into buffer pets of data
data_size	Gint	M	By reference	Size of the choice data record (size of the structure record of parameter data)
data	Gdefchoice (structure)	O	By reference	Default choice device data structure
error_status	Gint	O	By reference	Error status

```
struct Gdefpick {                   /*O Default pick data*/
    Gintlist *pets;                 /*O List of prompt and echo types*/
    Glimit e_area;                  /*O Default echo area*/
    Gpickrec record;                /*O Default pick data record*/
} data;

struct Gintlist {                   /*O User supplied prompt and echo type buffer*/
    Gint number;                    /*O Number of prompt and echo types*/
    Gint *integers;                 /*O List of prompt and echo types (in user
                                        supplied integer array)*/
} pets;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gpickrec {                   /*O Pick data record structure*/
    union {
        struct Gpickpet0001  {      /*O */
            Gfloat aperture;        /*O The pick aperture size */
        } pickpet1_datarec;
        struct Gpickpet0002 {       /*O */
            Gfloat aperture;        /*O The pick aperture size */
        } pickpet2_datarec;
        struct Gpickpet0003  {      /*O */
            Gfloat aperture;        /*O The pick aperture size */
        } pickpet3_datarec;
    }
} record;
```

INQUIRE DEFAULT STRING DEVICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit into buffer pets of data
data_size	Gint	M	By reference	Size of the choice data record (size of the structure record of parameter data)
data	Gdefchoice (structure)	O	By reference	Default string device data structure
error_status	Gint	O	By reference	Error status

```
struct Gdefstring {                 /*O Default string data*/
    Gint bufsiz;                    /*O Initial buffer size*/
    Gintlist *pets;                 /*O List of prompt and echo types*/
    Glimit e_area;                  /*O Default echo area*/
    Gstringrec record;              /*O Default string data record*/
} data;

struct Gintlist {                   /*O User supplied prompt and echo type buffer*/
    Gint number;                    /*O Number of prompt and echo types*/
    Gint *integers;                 /*O List of prompt and echo types (in user
                                      supplied integer array)*/
} pets;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gstringrec {                 /*O String data record structure*/
    union {
        struct Gstringpet0001 {     /*O */
            Gint  bufsiz;           /*O Buffer size*/
            Gint  position;         /*O Initial cursor position*/
            Gchar  *title_string;   /*O The title string */
        } stringpet1_datarec;
    }
} record;
```

INQUIRE DEFAULT STROKE DEVICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit into buffer pets of data
data_size	Gint	M	By reference	Size of the choice data record (size of the structure record of parameter data)
data	Gdefchoice (structure)	O	By reference	Default stroke device data structure
error_status	Gint	O	By reference	Error status

```
struct Gdefstroke {                    /*O Default stroke data*/
    Gint bufsiz;                       /*O Initial buffer size*/
    Gintlist *pets;                    /*O List of prompt and echo types*/
    Glimit e_area;                     /*O Default echo area*/
    Gstrokerec record;                 /*O Default string data record*/
} data;

struct Gintlist {                      /*O User supplied prompt and echo type buffer*/
    Gint number;                       /*O Number of prompt and echo types*/
    Gint *integers;                    /*O List of prompt and echo types (in user
                                         supplied integer array)*/
} pets;

Glimit e_area {                        /*O Coordinate limits*/
    Gfloat xmin;                       /*O X minimum limit */
    Gfloat xmax;                       /*O X maximum limit */
    Gfloat ymin;                       /*O Y minimum limit */
    Gfloat ymax;                       /*O Y maximum limit */
} e_area;

struct Gstrokerec {                    /*O Stroke data record structure*/
    union {
        struct Gstrokepet0001 {        /*O */
            Gint  bufsiz;              /*O Input buffer size*/
            Gint  editpos;             /*O Editing position*/
            Gpoint interval;           /*O X, Y interval*/
            Gfloat time;               /*O Time interval*/
        } strokepet1_datarec;
        struct Gstrokepet0002  {       /*O */
            Gint  bufsiz;              /*O Input buffer size*/
            Gint  editpos;             /*O Editing position*/
            Gpoint interval;           /*O X, Y interval*/
            Gfloat time;               /*O Time interval*/
        } strokepet2_datarec;
        struct Gstrokepet0003  {       /*O */
            Gint  bufsiz;              /*O Input buffer size*/
            Gint  editpos;             /*O Editing position*/
            Gpoint interval;           /*O X, Y interval*/
            Gfloat time;               /*O Time interval*/
            enum Gacf acf;             /*O Attribute control flag _ Enumerated type*/
            Gmkattr mk;                /*O Marker attributes*/
        } strokepet3_datarec;
        struct Gstrokepet0004  {       /*O */
            Gint  bufsiz;              /*O Input buffer size*/
            Gint  editpos;             /*O Editing position*/
            Gpoint interval;           /*O X, Y interval*/
            Gfloat time;               /*O Time interval*/
            enum Gacf acf;             /*O Attribute control flag _ Enumerated type*/
            Glnattr ln;                /*O Line attributes*/
        } strokepet4_datarec;
    }
} record;
```

```
struct Gmkattr {                    /*O Polymarker attributes*/
     enum Gasf {                    /*O Aspect control flag - Enumerated type*/
     } type|size|color;             /*O Markertype ASF|markerwidth ASF|markercolor ASF*/
     Gint marker;                   /*O Marker index*/
     struct Gmkbundl {              /*O Polymarker bundle*/
         Gint type;                 /*O Marker type*/
         Gfloat width;              /*O Markerwidth scale factor*/
         Gint colour;               /*O Polymarker colour index*/
     } bundl;                       /*O Marker bundle*/
} ln;                               /*O Polymarker attributes*/

struct Glnattr {                    /*O Polyline attributes*/
     enum Gasf {                    /*O Aspect control flag - Enumerated type*/
     } type|width|color;            /*O Linetype ASF|linewidth ASF|linecolor ASF*/
     Gint line;                     /*O Line index*/
     struct Glnbundl {              /*O Aspect control flag*/
         Gint type;                 /*O Line type*/
         Gfloat width;              /*O Linewidth scale factor*/
         Gint colour;               /*O Polyline colour index*/
     } bundl;                       /*O Line bundle*/
} ln;                               /*O Polyline attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INQUIRE DEFAULT VALUATOR DEVICE DATA

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_type	Gwstype	I	By reference	Workstation type
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Maximum number of prompt and echo types that will fit into buffer pets of data
data_size	Gint	M	By reference	Size of the choice data record (size of the structure record of parameter data)
data	Gdefchoice (structure)	O	By reference	Default valuator device data structure
error_status	Gint	O	By reference	Error status

Workstation Description Table Inquiry Functions
INQUIRE DEFAULT VALUATOR DEVICE DATA

```
struct Gdefval {                    /*O Default valuator data*/
    Gfloat value;                   /*O Initial value*/
    Gintlist *pets;                 /*O List of prompt and echo types*/
    Glimit e_area;                  /*O Default echo area*/
    Gvalrec record;                 /*O Default valuator data record*/
} data;

struct Gintlist {                   /*O User supplied prompt and echo type buffer*/
    Gint number;                    /*O Number of prompt and echo types*/
    Gint *integers;                 /*O List of prompt and echo types (in user
                                        supplied integer array)*/
} pets;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gvalrec {                    /*O Valuator data record structure*/
    union {
        struct Gvalpet0001 {        /*O */
            Gfloat low;             /*O Low range limit*/
            Gfloat high;            /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet1_datarec;
        struct Gvalpet0002 {        /*O */
            Gfloat low;             /*O Low range limit*/
            Gfloat high;            /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet2_datarec;
        struct Gvalpet0003  {       /*O */
            Gfloat low;             /*O Low range limit*/
            Gfloat high;            /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet3_datarec;
        struct Gvalpet_0001  {      /*O */
            Gfloat low;             /*O Low range limit*/
            Gfloat high;            /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_1_datarec;
```

```
        struct Gvalpet_0002  {   /*O */
            Gfloat low;          /*O Low range limit*/
            Gfloat high;         /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_2_datarec;
        struct Gvalpet_0003  {   /*O */
            Gfloat low;          /*O Low range limit*/
            Gfloat high;         /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_3_datarec;
    }
} record;
```

INQUIRE DISPLAY SPACE SIZE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqdisplaysize *(workstation_type, dspsz, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
dspsz	Gdspsize	O	By reference	Display space size structure
error_status	Gint	O	By reference	Error status

```
struct Gdspsize {                    /*O Display size*/
    enum Gdevunits units;            /*O Device coordinate units - Enumerated
                                         type*/
    Gpoint device;                   /*O Device coordinate unit size*/
    Gipoint raster;                  /*O Raster unit size*/
} dspsz;
```

Data type	Constant	Description
Gdevunits	GDC_METRES	Meters
	GDC_OTHER	Other

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} device;

struct Gipoint {                /*O Integer point*/
    Gint x;                     /*O X coordinate*/
    Gint y;                     /*O Y coordinate*/
} raster;
```

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmodwsattr *(workstation_type, dyn, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
dyn	Gmodws	O	By reference	Dynamic modification indicators structure
error_status	Gint	O	By reference	Error status

```
struct Gmodws {                          /*O Dynamic workstation attribute
                                             modification*/
    enum Gmodtype line;                  /*O Polyline - Enumerated type*/
    enum Gmodtype mark;                  /*O Polymarker - Enumerated type*/
    enum Gmodtype text;                  /*O Text - Enumerated type*/
    enum Gmodtype fill;                  /*O Fill area - Enumerated type*/
    enum Gmodtype pat;                   /*O Pattern - Enumerated type*/
    enum Gmodtype colour;                /*O Colour - Enumerated type*/
    enum Gmodtype wstran;                /*O Workstation transformation -
                                             Enumerated type*/
} dyn;
```

Data type	Constant	Description
Gmodtype	GIRG	Implicit regeneration necessary
	GIMM	Immediate

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmodsegattr *(workstation_type, dyn, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
dyn	Gmodseg	O	By reference	Dynamic modification of segment attributes
error_status	Gint	O	By reference	Error status

```
struct Gmodseg {                    /*O Dynamic segment attribute
                                        modification*/
    enum Gmodtype transform;        /*O Transformation - Enumerated type*/
    enum Gmodtype appear;           /*O Appearing (turning visible) -
                                        Enumerated type*/
    enum Gmodtype disappear;        /*O Disappearing (turning invisible) -
                                        Enumerated type*/
    enum Gmodtype highlight;        /*O Highlighting - Enumerated type*/
    enum Gmodtype priority;         /*O Priority- Enumerated type*/
    enum Gmodtype addition;         /*O Addition of primitives to segment -
                                        Enumerated type*/
    enum Gmodtype deletion;         /*O Deletion of segment - Enumerated type*/
} dyn;
```

Data type	Constant	Description
Gmodtype	GIRG	Implicit regeneration necessary
	GIMM	Immediate

INQUIRE FILL AREA FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Maximum number of hatch styles that will fit into buffer hatches
fac_size	Gint	O	By reference	Total number of hatch styles available
fac	Gflfac (structure)	O	By reference	Fill area facilities structure
error_status	Gint	O	By reference	Error status

```
struct Gflfac {                     /*O Fill area facilities*/
    Gint n_interiors;               /*O Number of interior styles*/
    enum Gflinter *interiors;       /*O List of available interior styles -
                                        enumerated*/
    Gintlist *hatches;              /*O List of available hatch styles*/
    Gint predefined;                /*O Number of predefined bundles*/
} fac;
```

Data type	Constant	Description
Gflinter	GHOLLOW	Hollow
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Hatched

```
struct Gintlist {                   /*O Hatch buffer*/
    Gint number;                    /*O Number of hatch styles*/
    Gint *integers;                 /*O Hatch styles (in user supplied integer
                                        array)*/
} hatches;
```

INQUIRE GENERALIZED DRAWING PRIMITIVE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqgdp *(workstation_type, functions, fac, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
functions	Gint	I	By value	GDP function identifier
fac	Ggdpfac (structure)	O	By reference	List of sets of attributes
error_status	Gint	O	By reference	Error status

```
struct Ggdpfac {                    /*O GDP facilities*/
    Gint n_attrs;                   /*M Number of GDP's*/
    enum Gattrs *attrs;             /*O List of attributes used -
                                        Enumerated type*/
} fac;
```

Data type	Constant	Description
Gattrs	GPOLYLINE	Polyline
	GPOLYMARKER	Polymarker
	GTEXT	Text
	GFILLAREA	Fill area

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By value	Workstation type
max_gdps	Gint	I	By value	Maximum number of gdp's that fit into buffer gdps
start	Gint	I	By value	Inquiry start position
gdps	Gintlist (structure)	O	By reference	List of gdp functions available
actual_gdps	Gint	O	By reference	Total number of available gdp functions
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                    /*O GDP buffer*/
    Gint number;                     /*O Number of GDPS*/
    Gint *integers;                  /*O GDPS (in user supplied integer array)*/
} gdps;
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqnumavailinput *(workstation_type, num, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
num	Gnumdev	O	By reference	Structure containing the number of input devices
error_status	Gint	O	By reference	Error status

```
struct Gnumdev {                    /*O Input devices
    Gint locator;                   /*O Locators*/
    Gint stroke;                    /*O Strokes*/
    Gint valuator;                  /*O Valuators*/
    Gint choice;                    /*O Choices*/
    Gint pick;                      /*O Picks*/
    Gint string;                    /*O Strings*/
} num;
```

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqnumsegpri *(workstation_type, numpri, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
numpri	Gint	O	By reference	Number of segment priorities supported
error_status	Gint	O	By reference	Error status

INQUIRE PATTERN FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Size of buffer facilities
fac_size	Gint	O	By reference	Size of pattern facilities structure (unused)
fac	Gint	O	By reference	Number of predefined pattern indexes
error_status	Gint	O	By reference	Error status

INQUIRE POLYLINE FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Maximum number of line types that will fit in buffer types
fac_size	Gint	O	By reference	Total number of line types available
fac	Glnfac (structure)	O	By reference	Polyline facilities structure
error_status	Gint	O	By reference	Error status

```
struct Glnfac{                  /*O Polyline facilities
    Gintlist *types;            /*O List of available line types*/
    Gint widths;                /*O Number of line widths*/
    Gfloat nom_width;           /*O Nominal width*/
    Gfloat min_width;           /*O Minimum width*/
    Gfloat max_width;           /*O Maximum width*/
    Gint predefined;            /*O Number of predefined bundles*/
} fac;

struct Gintlist {               /*O Line types buffer*/
    Gint number;                /*O Number of line types*/
    Gint *integers;             /*O Line types (in user supplied integer array)*/
} types;
```

INQUIRE POLYMARKER FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Maximum number of marker types that will fit in buffer types
fac_size	Gint	O	By reference	Total number of marker types available
fac	Gmkfac (structure)	O	By reference	Polymarker facilities structure
error_status	Gint	O	By reference	Error status

```
struct Gmkfac {                  /*O Polymarker facilities
    Gintlist *types;             /*O List of available marker types*/
    Gint sizes;                  /*O Number of marker sizes*/
    Gfloat nom_size;             /*O Nominal size*/
    Gfloat min_size;             /*O Minimum size*/
    Gfloat max_size;             /*O Maximum size*/
    Gint predefined;             /*O Number of predefined bundles*/
} fac;

struct Gintlist {                /*O Marker type buffer*/
    Gint number;                 /*O Number of marker types*/
    Gint *integers;              /*O Marker types (in user supplied integer
                                     array)*/
} types;
```

INQUIRE PREDEFINED COLOUR REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Color index
rep	Gcobundl (structure)	O	By reference	Predefined color representation structure
error_status	Gint	O	By reference	Error status

```
struct Gcobundl {                   /*O Color bundle*/
    Gfloat red;                     /*O Red intensity*/
    Gfloat green;                   /*O Green intensity*/
    Gfloat blue;                    /*O Blue intensity*/
} rep;
```

INQUIRE PREDEFINED FILL AREA REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Fill area index
rep	Gflbundl (structure)	O	By reference	Predefined fill area index representation structure
error_status	Gint	O	By reference	Error status

```
struct Gflbundl {                    /*O Fill area bundle*/
    enum Gflinter inter;             /*O Fill area interior style -
                                         enumerated*/
    Gint style;                      /*O Fill area style index*/
    Gint colour;                     /*O Fill area colour index*/
} rep;
```

Data type	Constant	Description
Gflinter	GHOLLOW	Hollow
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Hatched

INQUIRE PREDEFINED PATTERN REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Pattern index
rep	Gptbundl (structure)	O	By reference	Predefined pattern index representation structure
error_status	Gint	O	By reference	Error status

```
struct Gptbundl {                   /*O Pattern bundle*/
    struct Gipoint {                /*O Integer point*/
            Gint x;                 /*M Size of array in X dimension*/
            Gint y;                 /*M Size of array in Y dimension*/
        } size;                     /*O Pattern array size*/
    Gint *array;                    /*O Pattern array (in user supplied array)*/
} rep;
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Polyline index
rep	Glnbundl (structure)	O	By reference	Predefined polyline index representation structure
error_status	Gint	O	By reference	Error status

```
struct Glnbundl {                   /*O Polyline bundle*/
    Gint type;                      /*O Line type*/
    Gfloat width;                   /*O Linewidth scale factor*/
    Gint colour;                    /*O Polyline colour index*/
} rep;
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Polymarker index
rep	Gmkbundl (structure)	O	By reference	Predefined polymarker index representation structure
error_status	Gint	O	By reference	Error status

```
struct Gmkbundl {                   /*O Polymarker bundle*/
    Gint type;                      /*O Marker type*/
    Gfloat size;                    /*O Marker size scale factor*/
    Gint colour;                    /*O Polymarker colour index*/
} rep;
```

INQUIRE PREDEFINED TEXT REPRESENTATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
index	Gint	I	By value	Text index
rep	Gtxbundl (structure)	O	By reference	Predefined text index representation structure
error_status	Gint	O	By reference	Error status

```
struct Gtxbundl {                       /*O Text bundle*/
    Gtxfp fp;                           /*O Font and precision*/
    Gfloat exp;                         /*O Character expansion*/
    Gfloat space;                       /*O Character spacing*/
    Gint colour;                        /*O Text colour index*/
} rep;

struct Gtxfp {                          /*O Text font and precision*/
    Gint font;                          /*O Text font*/
    enum Gtxprec prec;                  /*O Text precision - Enumerated type*/
} fp;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

INQUIRE TEXT FACILITIES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

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

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
bufsize	Gint	I	By value	Maximum number of fonts that will fit in the buffer fp_list of fac
fac_size	Gint	O	By reference	Total number of fonts available
fac	Gtxfac (structure)	O	By reference	Text facilities structure
error_status	Gint	O	By reference	Error status

Workstation Description Table Inquiry Functions
INQUIRE TEXT FACILITIES

```
struct Gtxfac {                    */O Text facilities*/
     Gint fps;                     */O Number of fonts and precisions */
     Gtxfp *fp_list;               */O List of available fonts and precisions */
     Gint heights;                 */O Number of character heights */
     Gfloat min_ht;                */O Minimum height */
     Gfloat max_ht;                */O Maximum height */
     Gint expansions;              */O Number of character expansion factors */
     Gfloat min_exp;               */O Minimum expansion factor */
     Gfloat max_exp;               */O Maximum expansion factor */
     Gint predefined;              */O Number of predefined bundles */
} fac;

struct Gtxfp {                     /*O User supplied text font and precision buffer*/
    Gint font;                     /*O Text font*/
    enum Gtxprec {                 /*O Text precision - Enumerated type*/
    } fp_list;
} fps;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

INQUIRE WORKSTATION CATEGORY

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqwscategory *(workstation_type, cat, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
cat	Gwscat (enumerated)	O	By reference	Workstation category
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Gwscat	GOUTPUT	Output
	GINPUT	Input
	GOUTIN	Ouput & Input
	GWISS	Workstation independent segment storage
	GMO	Metafile output
	GMI	Metafile input

INQUIRE WORKSTATION CLASSIFICATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqwsclass *(workstation_type, class, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
class	Gwsclass (enumerated)	O	By reference	Workstation classification
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Gwsclass	GVECTOR	Vector
	GRASTER	Raster
	GOTHER	Other

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmaxwssttables *(workstation_type, tables, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_ type	Gwstype	I	By reference	Workstation type
tables	Gwstables (structure)	O	By reference	Length of workstation tables
error_status	Gint	O	By reference	Error status

```
struct Gwstables{                    /*O Length of workstation tables*/
     Gint line;                      /*O Polyline tables*/
     Gint mark;                      /*O Polymarker tables*/
     Gint text;                      /*O Text tables*/
     Gint fill;                      /*O Fill area tables*/
     Gint pat;                       /*O Pattern tables*/
     Gint colour;                    /*O Colour tables*/
} tables;
```

INQUIRE CLIPPING

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqclip *(clipping, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
clipping	Gcliprect (structure)	O	By reference	Current clipping indicator and rectangle
error_status	Gint	O	By reference	Error status

```
struct Gcliprect {                  /*O Clipping rectangle*/
     enum Gclip ind;                /*O Clipping indicator -
                                        Enumerated type*/
     struct Glimit {                /*O Coordinate limits*/
             Gfloat xmin;           /*O X minimum limit */
             Gfloat xmax;           /*O X maximum limit */
             Gfloat ymin;           /*O Y minimum limit */
             Gfloat ymax;           /*O Y maximum limit */
     } rec;                         /*O Clipping rectangle*/
} clipping:
```

Data type	Constant	Description
Gclip	GCLIP	Clipping
	GNCLIP	No clipping

INQUIRE LIST OF ASPECT SOURCE FLAGS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqasf *(asflist, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
asflist	Gasfs (structure)	O	By reference	Array of 13 aspect source flags
error_status	Gint	O	By reference	Error status

INQUIRE LIST OF ASPECT SOURCE FLAGS

```
struct Gasfs {                      /*O Aspect source flags structure*/
     enum Gasf ln_type;             /*O Line type - Enumerated type*/
     enum Gasf ln_width;            /*O Line width - Enumerated type*/
     enum Gasf ln_colour;           /*O Line colour - Enumerated type*/
     enum Gasf mk_type;             /*O Marker type - Enumerated type*/
     enum Gasf mk_size;             /*O Marker size - Enumerated type*/
     enum Gasf mk_colour;           /*O Marker colour - Enumerated type*/
     enum Gasf tx_fp;               /*O Text font and precision -
                                        Enumerated type*/
     enum Gasf tx_exp;              /*O Text expansion -
                                        Enumerated type*/
     enum Gasf tx_space;            /*O Text character spacing -
                                        Enumerated type*/
     enum Gasf tx_colour;           /*O Text colour - Enumerated type*/
     enum Gasf fl_inter;            /*O Fill area interior style -
                                        Enumerated type*/
     enum Gasf fl_style;            /*O Fill area style index -
                                        Enumerated type*/
     enum Gasf fl_colour;           /*O Fill area colour - Enumerated type*/
} asflist;
```

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INQUIRE CHARACTER BASE VECTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharbase *(base, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
base	Gpoint (structure)	O	By reference	X and Y value of character base vector
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} base;
```

INQUIRE CHARACTER EXPANSION FACTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharexpan *(chexp, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
chexp	Gfloat	O	By reference	Character expansion factor
error_status	Gint	O	By reference	Error status

INQUIRE CHARACTER HEIGHT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharheight *(height, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
height	Gfloat	O	By reference	Character height in world coordinates
error_status	Gint	O	By reference	Error status

INQUIRE CHARACTER SPACING

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharspace *(chspc, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
chspc	Gfloat	O	By reference	Character spacing
error_status	Gint	O	By reference	Error status

INQUIRE CHARACTER UP VECTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharup *(up, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
up	Gpoint (structure)	O	By reference	X and Y value of character up vector
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} up;
```

INQUIRE CHARACTER WIDTH

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcharwidth *(width, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
width	Gfloat	O	By reference	Character width
error_status	Gint	O	By reference	Error status

INQUIRE FILL AREA COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqfillcolourind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Fill area color index
error_status	Gint	O	By reference	Error status

INQUIRE FILL AREA INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqfillind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Fill area index
error_status	Gint	O	By reference	Error status

INQUIRE INPUT QUEUE OVERFLOW

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqinputoverflow *(overflow, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
overflow	Gqueue	O	By reference	Input queue overflow information
error_status	Gint	O	By reference	Error indicator pointer

```
struct Gqueue {                 /*O Queue information*/
    Gint ws;                    /*O Workstation identifier*/
    enum Giclass {              /*O Input device class - Enumerated type*/
    Gint devno;                 /*O Logical input device number*/
} overflow;
```

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

INQUIRE MORE SIMULTANEOUS EVENTS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmoreevents *(events, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
events	Gsimultev (enumerated)	O	By reference	More simultaneous events
error_status	Gint	O	By reference	Error indicator pointer

Data type	Constant	Description
Gsimultev	GNOMORE	No more simultaneous events
	GMORE	More simultaneous events

INQUIRE MARKER SIZE SCALE FACTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmarkersize *(mksize, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
mksize	Gfloat	O	By reference	Polymarker size scale factor
error_status	Gint	O	By reference	Error status

INQUIRE MARKER TYPE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmarkertype *(mktype, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
mktype	Gmktype	O	By reference	Polymarker type
error_status	Gint	O	By reference	Error status

Data type	Constant	Value	Description
Gmktype	GMK_POINT	1	Point
	GMK_PLUS	2	Plus sign
	GMK_STAR	3	Star
	GMK_O	4	Circle
	GMK_X	5	X (Cross)
	GMK_SOLID_DIAMOND	–13	Solid Diamond
	GMK_DIAMOND	–12	Diamond
	GMK_SOLID_HGLASS	–11	Solid Hourglass
	GMK_HOURGLASS	–10	Hourglass
	GMK_SOLID_BOWTIE	–9	Solid Bowtie
	GMK_BOWTIE	–8	Bowtie
	GMK_SOLID_SQUARE	–7	Solid Square
	GMK_SQUARE	–6	Square

Data type	Constant	Value	Description
	GMK_SOLID_TRI_DOWN	–5	Solid Triangle Down
	GMK_TRIANGLE_DOWN	–4	Triangle Down
	GMK_SOLID_TRI_UP	–3	Solid Triangle Up
	GMK_TRIANGLE_UP	–2	Triangle Up
	GMK_SOLID_CIRCLE	–1	Solid Circle

NOTE

Beginning at GMK_SOLID_DIAMOND, the enumerated type changes value to –13. Do not use the value. Use instead the proper name for the enumerated type. Your program may not compile correctly if you use the enumerated type value.

INQUIRE NORMALIZATION TRANSFORMATION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqntran *(num, tran, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
num	Gint	I	By value	Normalization transformation number
tran	Gtran (structure)	O	By reference	Transformation asociated with the specified normalization number
error_status	Gint	O	By reference	Error status

```
struct Gtran {                      /*O Transformation*/
    struct Glimit {                 /*O Coordinate limits*/
        Gfloat xmin;                /*O X minimum limit */
        Gfloat xmax;                /*O X maximum limit */
        Gfloat ymin;                /*O Y minimum limit */
        Gfloat ymax;                /*O Y maximum limit */
    } w;                            /*O window */
    struct Glimit {                 /*O Coordinate limits*/
        Gfloat xmin;                /*O X minimum limit */
        Gfloat xmax;                /*O X maximum limit */
        Gfloat ymin;                /*O Y minimum limit */
        Gfloat ymax;                /*O Y maximum limit */
    } v;                            /*O viewport */
} tran;
```

INQUIRE PATTERN REFERENCE POINT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqpatrefpt *(prp, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
prp	Gpoint (structure)	O	By reference	X and Y value of pattern reference point in world coordinates
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} prp;
```

INQUIRE PATTERN HEIGHT VECTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqpatheight *(heightvec, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
heightvec	Gpoint (structure)	O	By reference	X and Y value of pattern height vector
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} heightvec;
```

INQUIRE PATTERN WIDTH VECTOR

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqpatwidth *(widthvec, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
widthvec	Gpoint (structure)	O	By reference	X and Y value of pattern width vector
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} widthvec;
```

INQUIRE CURRENT PICK IDENTIFIER VALUE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcurpickid *(pickid, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
pickid	Gint	O	By reference	Current pick identifier
error_status	Gint	O	By reference	Error status

INQUIRE POLYLINE COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqlinecolourind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Polyline color index
error_status	Gint	O	By reference	Error status

INQUIRE POLYLINE INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqlineind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Polyline index
error_status	Gint	O	By reference	Error status

INQUIRE POLYMARKER COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmarkercolourind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Polymarker color index
error_status	Gint	O	By reference	Error status

INQUIRE POLYMARKER INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqmarkerind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Polymarker index
error_status	Gint	O	By reference	Error status

INQUIRE TEXT ALIGNMENT

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtextalign *(align, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
align	Gtxalign (structure)	O	By reference	Text alignment
error_status	Gint	O	By reference	Error status

```
struct Gtxalign {                /*I Text alignment*/
    enum Gtxhor {                /*I Horizontal alignment -
                                     Enumerated type*/
    } hor;
    enum Gtxver {                /*I Vertical alignment -
                                     Enumerated type*/
    } ver;
} align;
```

Data type	Constant	Description
Gtxhor	GAH_NORMAL	Normal
	GAH_LEFT	Left
	GAH_CENTRE	Center
	GAH_RIGHT	Right

Data type	Constant	Description
Gtxver	GAV_NORMAL	Normal
	GAV_TOP	Top
	GAV_CAP	Cap
	GAV_HALF	Half
	GAV_BASE	Base
	GAV_BOTTOM	Bottom

INQUIRE TEXT COLOUR INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtextcolourind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Text color index
error_status	Gint	O	By reference	Error status

INQUIRE TEXT FONT AND PRECISION

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtextfontprec *(txfp , error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
txfp	Gtxfp (structure)	O	By reference	Text and font precision pointer
error_status	Gint	O	By reference	Error status

```
struct Gtxfp {                          /*O Text facilities*/
    Gint font;                          /*O Text font*/
    enum Gtxprec {                      /*O Text precision -
                                            Enumerated type*/
    } prec;
} txfp;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

INQUIRE TEXT INDEX

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtextind *(index, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
index	Gint	O	By reference	Text index
error_status	Gint	O	By reference	Error status

INQUIRE TEXT PATH

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtexpath *(text_path, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
text_path	Gtxpath (enumerated)	O	By reference	Text path
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Gtxpath	GTP_RIGHT	Right
	GTP_LEFT	Left
	GTP_UP	Up
	GTP_DOWN	Down

INQUIRE NAME OF OPEN SEGMENT

Operating States: GSGOP

Syntax

ginqnameopenseg *(segment_name, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	O	By reference	Segment name
error_status	Gint	O	By reference	Error status

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqcurntrannum *(tran, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
tran	Gint	O	By reference	Current normalization transformation number
error_status	Gint	O	By reference	Error status

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqntrannum *(max_ntrans, start, ntrans, actual_ntrans, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
max_ntrans	Gint	I	By value	Maximum number of transformations that fit into buffer ntrans
start	Gint	I	By value	Inquiry start position
ntrans	Gintlist (structure)	O	By reference	List of normalization transformation numbers in order of decreasing priority
actual_ntrans	Gint	O	By reference	Total number of available transformations
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Transformation buffer*/
    Gint number;                    /*O Number of transformations*/
    Gint *integers;                 /*O Transformations (in user supplied
                                        integer array)*/
} ntrans;
```

INQUIRE SET OF ACTIVE WORKSTATIONS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqactivews *(max_ids, start, wsids, actual_ids, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
max_ids	Gint	I	By value	Maximum number of workstation identifiers that fit in wsids buffer
start	Gint	I	By value	Start position of ids
wsids	Gintlist (structure)	O	By reference	List of workstation identifiers
actual_ids	Gint	O	By reference	Total number of workstation identifiers in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Workstation ID list*/
    Gint number;                    /*O Number of workstation identifiers*/
    Gint *integers;                 /*O Workstation identifiers (in user
                                        supplied integer array)*/
} wsids;
```

INQUIRE SET OF OPEN WORKSTATIONS

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqopenws *(max_ids, start, wsids, actual_ids, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
max_ids	Gint	I	By value	Maximum number of workstation identifiers that fit in wsids buffer
start	Gint	I	By value	Start position of ids
wsids	Gintlist (structure)	O	By reference	List of workstation identifiers
actual_ids	Gint	O	By reference	Total number of workstation identifiers in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Workstation ID buffer*/
    Gint number;                    /*O Number of workstation identifiers*/
    Gint *integers;                 /*O Workstation IDS (in user supplied
                                        integer array)*/
} wsids;
```

INQUIRE SET OF SEGMENT NAMES IN USE

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqsegnames *(max_segnames, start, segnames, actual_segnames, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
max_ segnames	Gint	I	By value	Maximum number of segment names that fit in segnames buffer
start	Gint	I	By value	Start position of inquiry
segnames	Gintlist (structure)	O	By reference	Set of segment names in use
actual_ segnames	Gint	O	By reference	Total number of segment names in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                    /*O Segment names buffer*/
    Gint number;                     /*O Number of segment names*/
    Gint *integers;                  /*O Segment names (in user supplied integer
                                         array)*/
} segnames;
```

INQUIRE CHOICE DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqchoicest *(workstation_id, device_name, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	Choice device name
bufsize	Gint	I	By value	Size of choice data record
state_size	Gint	O	By reference	Size of the choice data record
state	Gchoicest (structure)	O	By reference	Current choice state
error_status	Gint	O	By reference	Error status

NOTE

The choice data record is the structure record of parameter state.

```
struct Gchoicest {                  /*O Choice state*/
    enum Gimode mode;               /*O Mode - Enumerated type*/
    enum Gesw esw;                  /*O Echo switch - Enumerated type*/
    Gchoice choice;                 /*O Choice data*/
    Gint pet;                       /*O Prompt and echo type*/
    Glimit e_area;                  /*O Echo area*/
    Gchoicerec record;              /*O Choice data record*/
} state;
```

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
struct Gchoice {                    /*O Choice data*/
    Gcstat status;                  /*O Choice status*/
    Gint;                           /*O Choice number*/
} choice;

struct Glimit {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gchoicerec {                 /*O Choice data record structure*/
    union {
        struct Gchoicepet0001 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gint  *lengths;         /*O Lengths of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet1_datarec;
        struct Gchoicepet0002 {     /*O */
            Gint  number;           /*O Number of alternatives*/
            enum Gprflag *enable;   /*O Array of prompt flags - enumerated type*/
            Gchar *title_string;    /*O The title string */
        } choicepet2_datarec;
        struct Gchoicepet0003 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet3_datarec;
        struct Gchoicepet0004 {     /*O */
            Gint  number;           /*O Number of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet4_datarec;
        struct Gchoicepet0005 {     /*O */
            Gint  seg;              /*O Segment name*/
            Gint  number;           /*O Number of alternatives*/
            Gint  *pickids;         /*O Array of pick indentifiers*/
            Gchar *title_string;    /*O The title string */
        } choicepet5_datarec;
        struct Gchoicepet_0001 {    /*O */
            Gint  number;           /*O Number of choice strings*/
            Gint  *lengths;         /*O Lengths of choice strings*/
            Gchar **strings;        /*O Array of strings*/
            Gchar *title_string;    /*O The title string */
        } choicepet_1_datarec;
    }
} record;
```

Data type	Constant	Description
Gprflag	GPROFF	Prompt off
	GPRON	Prompt on

INQUIRE COLOUR REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqcolourrep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Color index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Gcobundl (structure)	O	By reference	Color representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gcobundl {                   /*O Color bundle*/
    Gfloat red;                     /*O Red intensity*/
    Gfloat green;                   /*O Green intensity*/
    Gfloat blue;                    /*O Blue intensity*/
} rep;
```

INQUIRE FILL AREA REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqfillrep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Fill area index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Gflbundl (structure)	O	By reference	Fill area representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gflbundl {                    /*O Fill area bundle*/
    Gflinter inter;                  /*O Fill area interior style
                                         - enumerated*/
    Gint style;                      /*O Fill area style index*/
    Gint colour;                     /*O Fill area colour index*/
} rep;
```

Data type	Constant	Description
Gflinter	GHOLLOW	Hollow
	GSOLID	Solid
	GPATTERN	Pattern
	GHATCH	Hatched

INQUIRE LIST OF COLOUR INDICES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqcolourindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of color indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Indices buffer*/
    Gint number;                    /*O Number of color indices*/
    Gint *integers;                 /*O List of color indices (in user supplied
                                        integer array)*/
} indices;
```

INQUIRE LIST OF FILL AREA INDICES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqfillindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of fill area indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Indices buffer*/
    Gint number;                    /*O Number of fill indices*/
    Gint *integers;                 /*O List of fill indices (in user supplied
                                        integer array)*/
} indices;
```

INQUIRE LIST OF PATTERN INDICES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqpatindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of pattern indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Indices buffer*/
    Gint number;                    /*O Number of pattern indices*/
    Gint *integers;                 /*O List of pattern indices (in user
                                        supplied integer array)*/
} indices;
```

INQUIRE LIST OF POLYLINE INDICES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqlineindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of defined polyline indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                    /*O Indices buffer*/
    Gint number;                     /*O Number of polyline indices*/
    Gint *integers;                  /*O List of polyline indices (in user
                                         supplied integer array)*/
} indices;
```

INQUIRE LIST OF POLYMARKER INDICES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqmarkerindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of polymarker indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Indices buffer*/
    Gint number;                    /*O Number of polymarker indices*/
    Gint *integers;                 /*O List of polymarker indices (in user
                                        supplied integer array)*/
} indices;
```

INQUIRE LOCATOR DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqlocst *(workstation_id, device_name, workstation_type, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	Locator device name
type	Ginqtype (enumerated)	I	By value	Type of returned values
bufsize	Gint	I	By value	Size of the locator data record
state_size	Gint	O	By reference	Size of the locator data record
state	Glocst (structure)	O	By reference	Current locator state
error_status	Gint	O	By reference	Error status

NOTE

The locator data record is the structure record of parameter state.

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Glocst {                 /*O Locator state*/
    Gimode mode;                /*O Mode*/
    Gesw esw;                   /*O Echo switch*/
    Gloc loc;                   /*O Locator data*/
    Gint pet;                   /*O Prompt and echo type*/
    Glimit e_area;              /*O Echo area*/
    Glocrec record;             /*O Locator data record*/
} state;
```

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
struct Gloc {                       /*O Locator data*/
    Gint transform;                 /*O Normalization transformation
                                        number*/
    Gpoint position;                /*O Locator position*/
} loc;

struct Gpoint {                     /*O Coordinate point*/
    Gfloat x;                       /*O X coordinate*/
    Gfloat y;                       /*O Y coordinate*/
} position;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Glocrec {                    /*O Locator data record structure*/
    union {
        struct Glocpet0001 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet1_datarec;
        struct Glocpet0002 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet2_datarec;
        struct Glocpet0003 {        /*O */
            Gchar *data;            /*O Device/implementation dependent data*/
        } locpet3_datarec;
        struct Glocpet0004 {        /*O */
            Gacf  acf;              /*O Attribute control flag*/
            Glnattr ln;             /*O Polyline attributes*/
        } locpet4_datarec;
        struct Glocpet0005 {        /*O */
            enum Gpfcf pfcf;        /*O Polyline/fill area control flag -
                                    /*O Enumerated type*/
            enum Gacf  acf;         /*O Attribute control flag _ Enumerated type*/
            union {
                Glnattr  ln;        /*O Polyline attributes*/
                Gflattr  fl;        /*O Fill area attributes*/
            } attr;
        } locpet5_datarec;
        struct Glocpet0006 {        /*O */
            Gchar *title_string;    /*O The title string */
        } locpet6_datarec;
        struct Glocpet_0001 {       /*O */
            Gfloat box_x;           /*O Size of the box in x */
            Gfloat box_y;           /*O Size of the box in y */
        } locpet_1_datarec;
        struct Glocpet_0002 {       /*O */
            enum Gpfcf pfcf;        /*O Polyline/fill area control flag -
                                    /*  Enumerated type*/
            enum Gacf  acf;         /*O Attribute control flag _ Enumerated type*/
```

```
    union {
        Glnattr  ln;      /*O Polyline attributes*/
        Gflattr  fl;      /*O Fill area attributes*/
    } attr;
} locpet_2_datarec;
struct Glocpet_0003 {     /*O */
    Gacf  acf;            /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1;/*O Point 1 for echo */
          Gpoint  point2;/*O Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /*O Polyline attributes */
          Gpoint point1;  /*O Point 1 for echo */
          Gpoint point2;  /*O Point 2 for echo */
       } lnecho;
    } attr;
} locpet_3_datarec;
struct Glocpet_0004 {     /*O */
    Gacf  acf;            /*O Attribute control flag*/
    Glnattr ln;           /*O Polyline attributes*/
} locpet_4_datarec;
struct Glocpet_0005 {     /*O */
    Gacf  acf;            /*O Attribute control flag*/
    Glnattr ln;           /*O Polyline attributes*/
} locpet_5_datarec;
struct Glocpet_0006 {     /*O */
    Gacf  acf;            /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1;/*O Point 1 for echo */
          Gpoint  point2;/*O Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /*O Polyline attributes */
          Gpoint point1;  /*O Point 1 for echo */
          Gpoint point2;  /*O Point 2 for echo */
       } lnecho;
    } attr;
} locpet_6_datarec;
struct Glocpet_0007 {     /*O */
    Gacf  acf;            /*O Attribute control flag */
    union {
       struct {
          Gpoint  point1;/*O Point 1 for echo */
          Gpoint  point2;/*O Point 2 for echo */
       } echo;
        struct {
          Glnattr ln;     /*O Polyline attributes */
          Gpoint point1;  /*O Point 1 for echo */
          Gpoint point2;  /*O Point 2 for echo */
       } lnecho;
```

```
            } attr;
        } locpet_7_datarec;
        struct Glocpet_0008 {     /*O */
            Gacf  acf;            /*O Attribute control flag */
            union {
               struct {
                  Gpoint  point1;/*O Point 1 for echo */
                  Gpoint  point2;/*O Point 2 for echo */
               } echo;
                struct {
                  Glnattr ln;     /*O Polyline attributes */
                  Gpoint point1; /*O Point 1 for echo */
                  Gpoint point2; /*O Point 2 for echo */
               } lnecho;
            } attr;
        } locpet_8_datarec;
        struct Glocpet_0009 {     /*O */
            Gacf  acf;            /*O Attribute control flag*/
            Glnattr ln;           /*O Polyline attributes*/
        } locpet_9_datarec;
        struct Glocpet_0010 {     /*O */
            Gacf  acf;            /*O Attribute control flag*/
            Glnattr ln;           /*O Polyline attributes*/
        } locpet_10_datarec;
        struct Glocpet_0011 {     /*O */
            Gchar *data;          /*O Device/implementation dependent data*/
        } locpet_11_datarec;
        struct Glocpet_0012 {     /*O */
            Gacf  acf;            /*O Attribute control flag*/
            Glnattr ln;           /*O Polyline attributes*/
        } locpet_12_datarec;
    }
} record;                         /*O Locator data record*/

struct Glnattr {                  /*O Polyline attributes*/
     enum Gasf {                  /*O Aspect control flag - Enumerated
                                      type*/
     } type|width|color;          /*O Linetype ASF|linewidth ASF|linecolor
                                      ASF*/
     Gint line;                   /*O Line index*/
     struct Glnbundl {            /*O Aspect control flag*/
         Gint type;               /*O Line type*/
         Gfloat width;            /*O Linewidth scale factor*/
         Gint colour;             /*O Polyline colour index*/
     } bundl;                     /*O Line bundle*/
} ln;                             /*O Polyline attributes*/
```

```
struct Gflattr {                    /*I Fill area attributes*/
     enum Gasf {                    /*I Aspect control flag - Enumerated type*/
     } inter|style|color;           /*I Fill interior ASF|fill style ASF|linecolor ASF*/
     Gint fill;                     /*I Fill color*/
     struct Gflbundl {              /*I Fill area bundle*/
         Gint inter;                /*I Fill area interior*/
         Gint style;                /*I Fill area style*/
         Gint colour;               /*I Fill area colour index*/
     } bundl;                       /*I Fill area bundle*/
} fl;                               /*I Fill area attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gpfcf	GPF_POLYLINE	Polyline
	GPF_FILLAREA	Fill area

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INQUIRE PATTERN REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqpatrep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Pattern index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Gptbundl (structure)	O	By reference	Pattern representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gptbundl {                   /*O Pattern bundle*/
    struct Gipoint {                /*O Integer point*/
        Gint x;                     /*M Size of array in X dimension*/
        Gint y;                     /*M Size of array in Y dimension*/
    } size;                         /*O Pattern array size*/
    Gint *array;                    /*O Pattern array (in user supplied array)*/
} rep;
```

INQUIRE PICK DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqpickst *(workstation_id, device_name, workstation_type, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	Pick device name
type	Ginqtype (enumerated)	I	By value	Type of returned values
bufsize	Gint	I	By value	Size of pick data record
state_size	Gint	O	By reference	Size of the pick data record
state	Gpickst (structure)	O	By reference	Current pick state
error_status	Gint	O	By reference	Error status

NOTE

The pick data record is the structure record of parameter state.

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gpickst {                    /*O Pick state*/
    enum Gimode mode;               /*O Mode - Enumerated type*/
    enum Gesw esw;                  /*O Echo switch - Enumerated type*/
    Gpick pick;                     /*O Pick data*/
    Gint pet;                       /*O Prompt and echo type*/
    Glimit e_area;                  /*O Echo area*/
    Gpickrec record;                /*O Pick data record*/
} state;
```

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
struct Gpick {                          /*O Pick data*/
     enum Gpstat {                      /*O Pick data - Enumerated type*/
     } status;                          /*O Pick status*/
     Gint seg;                          /*O Pick segment*/
     Gint pickid;                       /*O Pick identifier*/
} pick;

Glimit e_area {                         /*O Coordinate limits*/
    Gfloat xmin;                        /*O X minimum limit */
    Gfloat xmax;                        /*O X maximum limit */
    Gfloat ymin;                        /*O Y minimum limit */
    Gfloat ymax;                        /*O Y maximum limit */
} e_area;

struct Gpickrec {                       /*O Pick data record structure*/
    union {
        struct Gpickpet0001  {          /*O */
            Gfloat aperture;            /*O The pick aperture size */
        } pickpet1_datarec;
        struct Gpickpet0002 {           /*O */
            Gfloat aperture;            /*O The pick aperture size */
        } pickpet2_datarec;
        struct Gpickpet0003  {          /*O */
            Gfloat aperture;            /*O The pick aperture size */
        } pickpet3_datarec;
    }
} record;
```

Data type	Constant	Description
Gpstat	GP_OK	Input obtained
	GP_NOPICK	Input is NOPICK
	GP_NONE	No input obtained

INQUIRE POLYLINE REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqlinerep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Polyline index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Glnbundl (structure)	O	By reference	Polyline representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Glnbundl {                   /*O Aspect control flag*/
    Gint type;                      /*O Line type*/
    Gfloat width;                   /*O Linewidth scale factor*/
    Gint colour;                    /*O Polyline colour index*/
} rep;
```

INQUIRE POLYMARKER REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqmarkerrep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Polymarker index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Gmkbundl (structure)	O	By reference	Polymarker representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gmkbundl {                   /*O Polymarker bundle*/
    Gint type;                      /*O Marker type*/
    Gfloat size;                    /*O Marker size scale factor*/
    Gint colour;                    /*O Polymarker colour index*/
} rep;
```

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqsegnamesws *(workstation_id, max_segnames, start, segnames, actual_segnames, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_ segnames	Gint	I	By value	Maximum number of segment names that fit into buffer segnames
start	Gint	I	By value	Inquiry start position
segnames	Gintlist (structure)	O	By reference	Set of stored segments for this workstation
actual_ segnames	Gint	O	By reference	Total number of available segnames in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Segment names buffer*/
    Gint number;                    /*O Number of segment names*/
    Gint *integers;                 /*O List of segment names (in user
                                        supplied integer array)*/
} segnames;
```

INQUIRE STRING DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqstringst *(workstation_id, device_name, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	String device name
bufsize	Gint	I	By value	Size of string data record
state_size	Gint	O	By reference	Size of the string data record
state	Gstringst (structure)	O	By reference	Current string state
error_status	Gint	O	By reference	Error status

NOTE

The string data record is the structure record of parameter state.

```
struct Gstringst {                  /*O String state*/
    Gimode mode;                    /*O Mode*/
    Gesw esw;                       /*O Echo switch*/
    Gchar *string;                  /*M String data*/
    Gint pet;                       /*O Prompt and echo type*/
    Glimit e_area;                  /*O Echo area*/
    Gstringrec record;              /*O String data record*/
} state;
```

NOTE

The string parameter should be initialized with a string, before the call. GKS will do a strlen to get the available size and will use this size as the maximum size of any returned string.

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
Glimit e_area {                          /*O Coordinate limits*/
    Gfloat xmin;                         /*O X minimum limit */
    Gfloat xmax;                         /*O X maximum limit */
    Gfloat ymin;                         /*O Y minimum limit */
    Gfloat ymax;                         /*O Y maximum limit */
} e_area;

struct Gstringrec {                      /*O String data record
                                             structure*/
    union {
        struct Gstringpet0001 {          /*O */
            Gint  bufsiz;                /*O Buffer size*/
            Gint  position;              /*O Initial cursor position*/
            Gchar  *title_string;        /*O The title string */
        } stringpet1_datarec;
    }
} record;
```

INQUIRE STROKE DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqstrokest *(workstation_id, device_name, workstation_type, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	Stroke device name
type	Ginqtype (enumerated)	I	By value	Type of returned values
bufsize	Gint	I	By value	Size of stroke data record
state_size	Gint	O	By reference	Size of the stroke data record
state	Gstrokest (structure)	O	By reference	Current stroke state
error_status	Gint	O	By reference	Error status

NOTE

The stroke data record is the structure record of parameter state.

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gstrokest {                  /*O Stroke state*/
    Gimode mode;                    /*O Mode*/
    Gesw esw;                       /*O Echo switch*/
    Gstroke *stroke;                /*O Stroke data*/
    Gint pet;                       /*O Prompt and echo type*/
    Glimit e_area;                  /*O Echo area*/
    Gstrokerec record;              /*O Stroke data record*/
} state;
```

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
struct Gstroke {                    /*O Stroke data*/
     Gint transform;                /*O Normalization transformation
                                        number*/
     Gint n_points;                 /*O Number of points in stroke*/
     struct Gpoint {                /*O Points in stroke*/
         Gfloat x;                  /*O X coordinate*/
         Gfloat y;                  /*O Y coordinate*/
     } *points;
} stroke;

struct Gpoint {                     /*O Coordinate point*/
    Gfloat x;                       /*O X coordinate*/
    Gfloat y;                       /*O Y coordinate*/
} points;

Glimit e_area {                     /*O Coordinate limits*/
    Gfloat xmin;                    /*O X minimum limit */
    Gfloat xmax;                    /*O X maximum limit */
    Gfloat ymin;                    /*O Y minimum limit */
    Gfloat ymax;                    /*O Y maximum limit */
} e_area;

struct Gstrokerec {                 /*O Stroke data record structure*/
    union {
        struct Gstrokepet0001 {     /*O */
            Gint  bufsiz;           /*O Input buffer size*/
            Gint  editpos;          /*O Editing position*/
            Gpoint interval;        /*O X, Y interval*/
            Gfloat time;            /*O Time interval*/
        } strokepet1_datarec;
        struct Gstrokepet0002  {    /*O */
            Gint  bufsiz;           /*O Input buffer size*/
            Gint  editpos;          /*O Editing position*/
            Gpoint interval;        /*O X, Y interval*/
            Gfloat time;            /*O Time interval*/
        } strokepet2_datarec;
        struct Gstrokepet0003  {    /*O */
            Gint  bufsiz;           /*O Input buffer size*/
            Gint  editpos;          /*O Editing position*/
            Gpoint interval;        /*O X, Y interval*/
            Gfloat time;            /*O Time interval*/
            enum Gacf acf;          /*O Attribute control flag _ Enumerated type*/
            Gmkattr mk;             /*O Marker attributes*/
        } strokepet3_datarec;
        struct Gstrokepet0004  {    /*O */
            Gint  bufsiz;           /*O Input buffer size*/
            Gint  editpos;          /*O Editing position*/
            Gpoint interval;        /*O X, Y interval*/
            Gfloat time;            /*O Time interval*/
            enum Gacf acf;          /*O Attribute control flag _ Enumerated type*/
            Glnattr ln;             /*O Line attributes*/
        } strokepet4_datarec;
    }
```

```
} record;

struct Gmkattr {                    /*O Polymarker attributes*/
     enum Gasf {                    /*O Aspect control flag -
                                        Enumerated type*/
     } type|size|color;             /*O Markertype ASF|markerwidth
                                        ASF|markercolor ASF*/
     Gint marker;                   /*O Marker index*/
     struct Gmkbundl {              /*O Polymarker bundle*/
         Gint type;                 /*O Marker type*/
         Gfloat width               /*O Markerwidth scale factor*/
         Gint colour;               /*O Polymarker colour index*/
     } bundl;                       /*O Marker bundle*/
} mk;                               /*O Polymarker attributes*/

struct Glnattr {                    /*O Polyline attributes*/
     enum Gasf {                    /*O Aspect control flag - Enumerated type*/
     } type|width|color;            /*O Linetype ASF|linewidth ASF|linecolor ASF*/
     Gint line;                     /*O Line index*/
     struct Glnbundl {              /*O Aspect control flag*/
         Gint type;                 /*O Line type*/
         Gfloat width               /*O Linewidth scale factor*/
         Gint colour;               /*O Polyline colour index*/
     } bundl;                       /*O Line bundle*/
} ln;                               /*O Polyline attributes*/
```

Data type	Constant	Description
Gacf	GCURRENT	Current
	GSPECIFIED	Specified

Data type	Constant	Description
Gasf	GBUNDLED	Bundled
	GINDIVIDUAL	Individual

INQUIRE TEXT EXTENT

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqtextextent *(workstation_id, position, string, extent, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
position	Gpoint (structure)	I	By reference	Text position
string	Gchar	I	By reference	Character string
extent	Gextent (structure)	O	By reference	Concatenation point and text extent parallelogram
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} position;

struct Gextent {                /*O Text extent*/
    Gpoint concat;              /*O Concatenation point*/
    Gpoint corner_1;            /*O Corner 1*/
    Gpoint corner_2;            /*O Corner 2*/
    Gpoint corner_3;            /*O Corner 3*/
    Gpoint corner_4;            /*O Corner 4*/
} extent;
```

INQUIRE LIST OF TEXT INDICES

Operating States: GGKOP, GWSOP, GWSAC, GSGOP

Syntax

ginqtextindices *(workstation_id, max_indices, start, indices, actual_indices, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
max_indices	Gint	I	By value	Maximum number of indices that fit into buffer indices
start	Gint	I	By value	Inquiry start position
indices	Gintlist (structure)	O	By reference	List of defined text indices
actual_indices	Gint	O	By reference	Total number of available indices in GKS
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                    /*O Text indices buffer*/
    Gint number;                     /*O Number of text indices*/
    Gint *integers;                  /*O List of text indices (in user
                                         supplied integer array)*/
} indices;
```

INQUIRE TEXT REPRESENTATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqtextrep *(workstation_id, index, type, rep, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
index	Gint	I	By value	Text index
type	Ginqtype (enumerated)	I	By value	Type of returned values
rep	Gtxbundl (structure)	O	By reference	Text representation
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Ginqtype	GSET	Set
	GREALIZED	Realized

```
struct Gtxbundl {                   /*I Text bundle*/
    Gtxfp fp;                       /*I Font and precision*/
    Gfloat exp;                     /*I Character expansion*/
    Gfloat space;                   /*I Character spacing*/
    Gint colour;                    /*I Polymarker colour index*/
} rep;

struct Gtxfp {                      /*I Text facilities*/
    Gint font;                      /*I Text font*/
    enum Gtxprec {                  /*I Text precision - Enumerated type*/
    } prec;
} fp;
```

Data type	Constant	Description
Gtxprec	GP_STRING	Lowest precision
	GP_CHAR	Moderate precision
	GP_STROKE	Highest precision

INQUIRE VALUATOR DEVICE STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqvalst *(workstation_id, device_name, bufsize, state_size, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
device_name	Gint	I	By value	Valuator device name
bufsize	Gint	I	By value	Size of valuator data record
state_size	Gint	O	By reference	Size of the valuator data record
state	Gvalst (structure)	O	By reference	Current valuator state
error_status	Gint	O	By reference	Error status

NOTE

The valuator data record is the structure record of parameter state.

```
struct Gvalst {                    /*O Valuator state*/
    enum Gimode mode;              /*O Mode - Enumerated type*/
    enum Gesw esw;                 /*O Echo switch - Enumerated type*/
    Gfloat val;                    /*O Valuator data*/
    Gint pet;                      /*O Prompt and echo type*/
    Glimit e_area;                 /*O Echo area*/
    Gvalrec record;                /*O Valuator data record*/
} state;
```

Data type	Constant	Description
Gimode	GREQUEST	Request mode
	GSAMPLE	Sample mode
	GEVENT	Event mode

Data type	Constant	Description
Gesw	GECHO	Echo
	GNOECHO	No echo

```
Glimit e_area {                    /*O Coordinate limits*/
    Gfloat xmin;                   /*O X minimum limit */
    Gfloat xmax;                   /*O X maximum limit */
    Gfloat ymin;                   /*O Y minimum limit */
    Gfloat ymax;                   /*O Y maximum limit */
} e_area;

struct Gvalrec {                   /*O Valuator data record structure*/
    union {
        struct Gvalpet0001 {       /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet1_datarec;
        struct Gvalpet0002 {       /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet2_datarec;
        struct Gvalpet0003  {      /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet3_datarec;
        struct Gvalpet_0001  {     /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_1_datarec;
        struct Gvalpet_0002  {     /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_2_datarec;
        struct Gvalpet_0003  {     /*O */
            Gfloat low;            /*O Low range limit*/
            Gfloat high;           /*O High range limit*/
            Gchar  *title_string;/*O The title string */
        } valpet_3_datarec;
    }
} record;
```

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqwsdeferupdatest *(workstation_id, du, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
du	Gwsdus (structure)	O	By reference	Current workstation deferral and update states
error_status	Gint	O	By reference	Error status

```
struct Gwsdus {                     /*O WS Deferral and update state*/
    enum Gdefmode defmode;          /*O Deferral mode - Enumerated type*/
    enum Gdspsurf dspsurf;          /*O Display surface -
                                        Enumerated type*/
    enum Girgmode irgmode;          /*O Implicit regeneration mode -
                                        Enumerated type*/
    enum Gnframe nframe;            /*O New frame action at update -
                                        Enumerated type*/
} du;
```

Data type	Constant	Description
Gdefmode	GASAP	As soon as possible
	GBNIG	Before next interaction globally
	GBNIL	Before next interaction locally
	GASTI	At some time

Data type	Constant	Description
Gdspsurf	GEMPTY	Empty
	GNOEMPTY	Not empty

Data type	Constant	Description
Girgmode	GSUPPRESSED	Supressed
	GALLOWED	Allowed

Data type	Constant	Description
Gnframe	GNO	No
	GYES	Yes

INQUIRE WORKSTATION CONNECTION AND TYPE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqwsconntype *(workstation_id, bufsize, ct_size, ct, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
bufsize	Gint	I	By value	Buffer size of buffer conn of ct
ct_size	Gint	O	By reference	Return size of buffer conn of ct
ct	Gwsct (structure)	O	By reference	Connection identifier and workstation type
error_status	Gint	O	By reference	Error status

```
struct Gwsct {                  /*O Workstation connection and type*/
    Gconn Gchar {               /*O Connection identifier*/
        } *conn;                /*O Workstation connection*/
    Gwstype int {               /*O Workstation type*/
        } *type;                /*O Type identifier*/
} ct;
```

INQUIRE WORKSTATION STATE

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqwsst *(workstation_id, state, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
state	Gwsstate (enumerated)	O	By reference	Current workstation state
error_status	Gint	O	By reference	Error status

Data type	Constant	Description
Gwsstate	GINACTIVE	Inactive
	GACTIVE	Active

INQUIRE WORKSTATION TRANSFORMATION

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqwstran *(workstation_id, wstran, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
wstran	Gwsti (structure)	O	By reference	Requested and current transformations
error_status	Gint	O	By reference	Error status

```
struct Gwsti {                          /*O Workstation transformation
                                            information*/
    enum Gwstus wstus;                  /*O Workstation transformation update
                                            state - Enumerated type*/
    Gtran request;                      /*O Requested transformation*/
    Gtran current;                      /*O Current transformation*/
} wstran;
```

Data type	Constant	Description
Gwstus	GNOTPENDING	Not pending
	GPENDING	Pending

Workstation State List Inquiry Functions
INQUIRE WORKSTATION TRANSFORMATION

```
struct Gtran {                      /*O Transformation*/
    Glimit w;                       /*O Window*/
    Glimit v;                       /*O Viewport*/
} request, current;

struct Glimit {                       /*O Coordinate limits*/
    Gfloat xmin;                      /*O X minimum limit */
    Gfloat xmax;                      /*O X maximum limit */
    Gfloat ymin;                      /*O Y minimum limit */
    Gfloat ymax;                      /*O Y maximum limit */
} w, v;
```

INQUIRE SET OF ASSOCIATED WORKSTATIONS

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqassocws *(segment_name, max, start, actual, assocws, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By value	Segment name
max	Gint	I	By value	Maximum number of workstation IDS that fit into buffer assocws
start	Gint	I	By value	Inquiry start position
actual	Gint	O	By reference	Total number of associated workstations
assocws	Gintlist (structure)	O	By reference	Set of associated workstation identifiers
error_status	Gint	O	By reference	Error status

```
struct Gintlist {                   /*O Workstation ID buffer*/
    Gint number;                    /*O Number of workstation identifiers*/
    Gint *integers;                 /*O List of workstation identifiers (in user
                                        supplied integer array)*/
} assocws;
```

INQUIRE SEGMENT ATTRIBUTES

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqsegattr *(segment_name, segattr, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
segment_ name	Gint	I	By reference	Segment name
segattr	Gsegattr (structure)	O	By reference	Segment attribute structure
error_status	Gint	O	By reference	Error status

```
struct Gsegattr {                    /*O Segment attribute values*/
     Gfloat segtran[2][3];           /*O Segment transformation*/
     enum Gsegvis segvis;            /*O Segment visibility -
                                         Enumerated type*/
     enum Gseghi seghi;              /*O Segment highlighting -
                                         Enumerated type*/
     Gfloat segpri;                  /*O Segment priority*/
     enum Gsegdet segdet;            /*O Segment detectability -
                                         Enumerated type*/
} segattr;
```

Data type	Constant	Description
Gsegvis	GVISIBLE	Visible
	GINVISIBLE	Invisible

Data type	Constant	Description
Gseghi	GNORMAL	Normal
	GHIGHLIGHTED	Highlighted

Data type	Constant	Description
Gsegdet	GUNDETECTABLE	Undetected
	GDETECTABLE	Detected

INQUIRE PIXEL

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqpixel *(workstation_id, ppoint, pix, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
ppoint	Gpoint (structure)	I	By reference	Pixel world coordinate location
pix	Gint	O	By reference	Pixel color
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                 /*O Coordinate point*/
    Gfloat x;                   /*O X coordinate*/
    Gfloat y;                   /*O Y coordinate*/
} base;
```

INQUIRE PIXEL ARRAY

Operating States: GGKOP, GGWSOP, GWSAC, GSGOP

Syntax

ginqpixelarray *(workstation_id, point, dimen, bufsize, covalid, pxarray, actual_size, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
point	Gpoint (structure)	I	By reference	Pixel array location pointer
dimen	Gidim (structure)	I	By reference	Pixel array dimensions pointer
bufsize	Gint	I	By value	Size of buffer pxarray
covalid	Gcovalid (enumerated)	O	By reference	Color validity indicator
pxarray	Gintlist (structure)	O	By reference	Returned pixel array
actual_size	Gint	O	By reference	Actual size of the available pixel array (not used)
error_status	Gint	O	By reference	Error status

```
struct Gpoint {                    /*I Coordinate point*/
    Gfloat x;                      /*I X coordinate*/
    Gfloat y;                      /*I Y coordinate*/
} point;

struct Gidim {                     /*I Dimension in integer values*/
    Gint x_dim;                    /*I X dimension*/
    Gint y_dim;                    /*I Y dimension*/
} dimen;
```

Data type	Constant	Description
Gcovalid	GABSENT	
	GPRESENT	

```
struct Gintlist {                  /*O Integer list*/
    Gint number;                   /*O Number of integers in list*/
    Gint *integers;                /*O List of integers*/
} pxarray;
```

INQUIRE PIXEL ARRAY DIMENSIONS

Operating States: GWSOP, GWSAC, GSGOP

Syntax

ginqpixelarraydim *(workstation_id, rect, dim, error_status)*

Arguments

Argument	Data Type	I/O	Mechanism	Description
workstation_id	Gint	I	By value	Workstation identifier
rect	Grect (structure)	I	By reference	Rectangle pointer
dim	Gidim (structure)	O	By reference	Pixel array dimensions pointer
error_status	Gint	O	By reference	Error status

```
struct Grect {                          /*O Coordinate rectangle*/
    struct Gpoint {                     /*O Coordinate point*/
        Gfloat x;                       /*O Upper left X coordinate*/
        Gfloat y;                       /*O Upper left Y coordinate*/
    } ul;
    struct Gpoint {                     /*O Coordinate point*/
        Gfloat x;                       /*O Lower right X coordinate*/
        Gfloat y;                       /*O Lower right Y coordinate*/
    } lr;
} rect;

struct Gidim {                          /*O Dimension in integer values*/
    Gint x_dim;                         /*O X dimension*/
    Gint y_dim;                         /*O Y dimension*/
} dim;
```

Appendix A

DEC GKS Function Names and C Binding Function Names

GKS$ interface, and the corresponding C binding name or names, if applicable. Function names are organized according to functional category.

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

DEC GKS Function	C Binding Function(s)
Control Functions	
GKS$ACTIVATE_WS	gactivatews
GKS$CLEAR_WS	gclearws
GKS$CLOSE_GKS	gclosegks
GKS$CLOSE_WS	gclosews
GKS$DEACTIVATE_WS	gdeactivatews
GKS$ESCAPE	gescape
GKS$MESSAGE	gmessage
GKS$OPEN_GKS	gopengks
GKS$OPEN_WS	gopenws
GKS$REDRAW_SEG_ON_WS	gredrawsegws
GKS$SET_DEFER_STATE	gsetdeferst
GKS$UPDATE_WS	gupdatews

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
Output Functions	
GKS$CELL_ARRAY	gcellarray
GKS$FILL_AREA	gfillarea
GKS$GDP	ggdp
GKS$POLYLINE	gpolyline
GKS$POLYMARKER	gpolymarker
GKS$TEXT	gtext
Attribute Functions	
GKS$SET_ASF	gsetasf
GKS$SET_COLOR_REP	gsetcolourrep
GKS$SET_FILL_COLOR_INDEX	gsetfillcolourind
GKS$SET_FILL_INDEX	gsetfillind
GKS$SET_FILL_INT_STYLE	gsetfillintstyle
GKS$SET_FILL_REP	gsetfillrep
GKS$SET_FILL_STYLE_INDEX	gsetfillstyleind
GKS$SET_PAT_REF_PT	gsetpatrefpt
GKS$SET_PAT_REP	gsetpatrep
GKS$SET_PAT_SIZE	gsetpatsize
GKS$SET_PLINE_COLOR_INDEX	gsetlinecolourind
GKS$SET_PLINE_INDEX	gsetlineind
GKS$SET_PLINE_LINETYPE	gsetlinetype
GKS$SET_PLINE_LINEWIDTH	gsetlinewidth
GKS$SET_PLINE_REP	gsetlinerep
GKS$SET_PMARK_COLOR_INDEX	gsetmarkercolourind
GKS$SET_PMARK_INDEX	gsetmarkerind
GKS$SET_PMARK_REP	gsetmarkerrep
GKS$SET_PMARK_SIZE	gsetmarkersize

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$SET_PMARK_TYPE	gsetmarkertype
GKS$SET_TEXT_ALIGN	gsettextalign
GKS$SET_TEXT_COLOR_INDEX	gsettextcolourind
GKS$SET_TEXT_EXPFAC	gsetcharexpan
GKS$SET_TEXT_FONTPREC	gsettextfontprec
GKS$SET_TEXT_HEIGHT	gsetcharheight
GKS$SET_TEXT_INDEX	gsettextind
GKS$SET_TEXT_PATH	gsettextpath
GKS$SET_TEXT_REP	gsettextrep
GKS$SET_TEXT_SPACING	gsetcharspace
GKS$SET_TEXT_UPVEC	gsetcharup
Transformation Functions	
GKS$SELECT_XFORM	gselntran
GKS$SET_CLIPPING	gsetclip
GKS$SET_WINDOW	gsetwindow
GKS$SET_VIEWPORT	gsetviewport
GKS$SET_VIEWPORT_PRIORITY	gsetviewportinputpri
GKS$SET_WS_WINDOW	gsetwswindow
GKS$SET_WS_VIEWPORT	gsetwsviewport
Segment Functions	
GKS$ACCUM_XFORM_MATRIX	gaccumtran (C binding Utility function)
GKS$ASSOC_SEG_WITH_WS	gassocsegws
GKS$CLOSE_SEG	gcloseseg
GKS$COPY_SEG_TO_WS	gcopysegws
GKS$CREATE_SEG	gcreateseg

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$DELETE_SEG	gdelseg
GKS$DELETE_SEG_FROM_WS	gdelsegws
GKS$EVAL_XFORM_MATRIX	gevaltran (C binding Utility function)
GKS$INSERT_SEG	ginsertseg
GKS$RENAME_SEG	grenameseg
GKS$SET_PICK_ID	gsetpickid
GKS$SET_SEG_DETECTABILITY	gsetdet
GKS$SET_SEG_HIGHLIGHTING	gsethighlight
GKS$SET_SEG_PRIORITY	gsetsegpri
GKS$SET_SEG_VISIBILITY	gsetvis
GKS$SET_SEG_XFORM	gsetsegtran
Input Functions	
GKS$AWAIT_EVENT	gawaitevent
GKS$FLUSH_DEVICE_EVENTS	gflushevents
GKS$GET_CHOICE	ggetchoice
GKS$GET_LOCATOR	ggetloc
GKS$GET_PICK	ggetpick
GKS$GET_STRING	ggetstring
GKS$GET_STROKE	ggetstroke
GKS$GET_VALUATOR	ggetval
GKS$INIT_CHOICE	ginitchoice
GKS$INIT_LOCATOR	ginitloc
GKS$INIT_PICK	ginitpick
GKS$INIT_STRING	ginitstring
GKS$INIT_STROKE	ginitstroke
GKS$INIT_VALUATOR	ginitval
GKS$REQUEST_CHOICE	greqchoice

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$REQUEST_LOCATOR	greqloc
GKS$REQUEST_PICK	greqpick
GKS$REQUEST_STRING	greqstring
GKS$REQUEST_STROKE	greqstroke
GKS$REQUEST_VALUATOR	greqval
GKS$SAMPLE_CHOICE	gsamplechoice
GKS$SAMPLE_LOCATOR	gsampleloc
GKS$SAMPLE_PICK	gsamplepick
GKS$SAMPLE_STRING	gsamplestring
GKS$SAMPLE_STROKE	gsamplestroke
GKS$SAMPLE_VALUATOR	gsampleval
GKS$SET_CHOICE_MODE	gsetchoicemode
GKS$SET_LOCATOR_MODE	gsetlocmode
GKS$SET_PICK_MODE	gsetpickmode
GKS$SET_STRING_MODE	gsetstringmode
GKS$SET_STROKE_MODE	gsetstrokemode
GKS$SET_VALUATOR_MODE	gsetvalmode
Inquiry Functions	
GKS Description Table	
GKS$INQ_LEVEL	ginqlevelgks
GKS$INQ_MAX_XFORM	ginqmaxntrannum
GKS$INQ_WS_MAX_NUM	ginqwsmaxnum
GKS$INQ_WSTYPE_LIST	ginqavailwstypes
GKS State List	
	ginqprimattr
GKS$INQ_ACTIVE_WS	ginqactivews

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$INQ_CLIP	ginqclip
GKS$INQ_CURRENT_XFORMNO	ginqcurntrannum
GKS$INQ_INDIV_ATTB	ginqindivattr
	ginqlinetype
	ginqlinewidth
	ginqlinecolourind
	ginqmarkertype
	ginqmarkersize
	ginqmarkercolourind
	ginqtextfontprec
	ginqcharexpan
	ginqcharspace
	ginqtextcolourind
	ginqfillintstyle
	ginqfillstyleind
	ginqfillcolourind
	ginqasf
GKS$INQ_INPUT_QUEUE_OVERFLOW	ginqinputoverflow
GKS$INQ_MORE_SIMUL_EVENTS	ginqmoreevents
GKS$INQ_NAME_OPEN_SEG	ginqnameopenseg
GKS$INQ_OPERATING_STATE	ginqopst
GKS$INQ_OPEN_WS	ginqopenws
GKS$INQ_PICK_ID	ginqcurpickid
GKS$INQ_PRIM_ATTB	ginqprimattr
	ginqmarkerind
	ginqtextind
	ginqcharheight
	ginqcharup

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
	ginqcharwidth
	ginqcharbase
	ginqtextpath
	ginqtextalign
	ginqfillind
	ginqpatheight
	ginqpatwidth
	ginqpatrefpt
GKS$INQ_SEG_NAMES	ginqsegnames
GKS$INQ_XFORM	ginqntran
GKS$INQ_XFORM_LIST	ginqntrannum
Workstation State List	
GKS$INQ_CHOICE_STATE	ginqchoicest
GKS$INQ_COLOR_INDEXES	ginqcolourindices
GKS$INQ_COLOR_REP	ginqcolourrep
GKS$INQ_FILL_INDEXES	ginqfillindices
GKS$INQ_FILL_REP	ginqfillrep
GKS$INQ_LOCATOR_STATE	ginqlocst
GKS$INQ_PAT_INDEXES	ginqpatindices
GKS$INQ_PAT_REP	ginqpatrep
GKS$INQ_PICK_STATE	ginqpickst
GKS$INQ_PLINE_INDEXES	ginqlineindices
GKS$INQ_PLINE_REP	ginqlinerep
GKS$INQ_PMARK_INDEXES	ginqmarkerindices
GKS$INQ_PMARK_REP	ginqmarkerrep
GKS$INQ_SEG_NAMES_ON_WS	ginqsegnamesws
GKS$INQ_STRING_STATE	ginqstringst

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$INQ_STROKE_STATE	ginqstrokest
GKS$INQ_TEXT_EXTENT	ginqtextextent
GKS$INQ_TEXT_INDEXES	ginqtextindices
GKS$INQ_TEXT_REP	ginqtextrep
GKS$INQ_VALUATOR_STATE	ginqvalst
GKS$INQ_WS_DEFER_AND_UPDATE	ginqwsdeferupdatest
GKS$INQ_WS_STATE	ginqwsst
GKS$INQ_WS_TYPE	ginqwsconntype
GKS$INQ_WS_XFORM	ginqwstran
Workstation Description Table	
GKS$INQ_AVAIL_GDP	ginqavailgdp
GKS$INQ_COLOR_FAC	ginqcolourfacil
GKS$INQ_DEF_CHOICE_DATA	ginqdefchoice
GKS$INQ_DEF_DEFER_STATE	ginqdefdeferst
GKS$INQ_DEF_LOCATOR_DATA	ginqdefloc
GKS$INQ_DEF_PICK_DATA	ginqdefpick
GKS$INQ_DEF_STRING_DATA	ginqdefstring
GKS$INQ_DEF_STROKE_DATA	ginqdefstroke
GKS$INQ_DEF_VALUATOR_DATA	ginqdefval
GKS$INQ_DYN_MOD_SEG_ATTB	ginqmodsegattr
GKS$INQ_DYN_MOD_WS_ATTB	ginqmodwsattr
GKS$INQ_FILL_FAC	ginqfillfacil
GKS$INQ_GDP	ginqgdp
GKS$INQ_INPUT_DEV	ginqnumavailinput
GKS$INQ_MAX_DS_SIZE	ginqdisplaysize
GKS$INQ_MAX_WS_STATE_TABLE	ginqmaxwssttables
GKS$INQ_PAT_FAC	ginqpatfacil

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
GKS$INQ_PLINE_FAC	ginqlinefacil
GKS$INQ_PMARK_FAC	ginqmarkerfacil
GKS$INQ_PREDEF_COLOR_REP	ginqpredcolourrep
GKS$INQ_PREDEF_FILL_REP	ginqpredfillrep
GKS$INQ_PREDEF_PAT_REP	ginqpredpatrep
GKS$INQ_PREDEF_PLINE_REP	ginqpredlinerep
GKS$INQ_PREDEF_PMARK_REP	ginqpredmarkerrep
GKS$INQ_PREDEF_TEXT_REP	ginqpredtextrep
GKS$INQ_SEG_PRIORITY	ginqnumsegpri
GKS$INQ_TEXT_FAC	ginqtextfacil
GKS$INQ_WS_CATEGORY	ginqwscategory
GKS$INQ_WS_CLASSIFICATION	ginqwsclass
Segment	
GKS$INQ_SEG_ASSOC_WS	ginqassocws
GKS$INQ_SEG_ATTB	ginqsegattr
Pixel	
GKS$INQ_PIXEL	ginqpixel
GKS$INQ_PIXEL_ARRAY	ginqpixelarray
GKS$INQ_PIXEL_ARRAY_DIM	ginqpixelarraydim
Metafile Functions	
GKS$GET_ITEM	ggettypegksm
GKS$INTERPRET_ITEM	ginterpret
GKS$READ_ITEM	greadgksm
GKS$WRITE_ITEM	gwritegksm

(continued on next page)

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

DEC GKS Function	C Binding Function(s)
Error Functions	
GKS$EMERGENCY_CLOSE	gemergencyclosegks
GKS$ERROR_HANDLER	gerrorhand
GKS$LOG_ERROR	gerrorlog

Appendix B

C Binding Type Definitions

This appendix alphabetically lists the type definitions for the VAX C Binding.

Table B–1: C Binding Type Definitions

Type Definition	Description
	A
enum { GCURRENT, GSPECIFIED } Gacf;	Attribute control flag
enum { GBUNDLED, GINDIVIDUAL } Gasf;	Aspect control flag

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gasf ln_type; Gasf ln_width; Gasf ln_colour; Gasf mk_type; Gasf mk_size; Gasf mk_colour; Gasf tx_fp; Gasf tx_exp; Gasf tx_space; Gasf tx_colour; Gasf fl_inter; Gasf fl_style; Gasf fl_colour } Gasfs;	Aspect source flags
enum { GPOLYLINE, GPOLYMARKER, GTEXT, GFILLAREA } Gattrs;	Attributes used
	C
char Gchar;	Character
struct { Gcstat status; Gint choice; } Gchoice;	Choice data

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint number; Gint *lengths; Gchar **strings; Gchar *title_string; } Gchoicepet0001;	Choice prompt and echo type 1 data record
struct { Gint number; enum Gprflag *enable; Gchar *title_string; } Gchoicepet0002;	Choice prompt and echo type 2 data record
struct { Gint number; Gchar **strings; Gchar *title_string; } Gchoicepet0003;	Choice prompt and echo type 3 data record
struct { Gint number; Gchar **strings; Gchar *title_string; } Gchoicepet0004;	Choice prompt and echo type 4 data record
struct { Gint seg; Gint number; Gint *pickids; Gchar *title_string; } Gchoicepet0005;	Choice prompt and echo type 5 data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint number; Gint *lengths; Gchar **strings; Gchar *title_string; } Gchoicepet_0001;	Choice prompt and echo type data record
union { Gchoicepet0001 choicepet1_datarec; Gchoicepet0002 choicepet2_datarec; Gchoicepet0003 choicepet3_datarec; Gchoicepet0004 choicepet4_datarec; Gchoicepet0005 choicepet5_datarec; Gchoicepet_0001 choicepet_1_datarec; } Gchoicerec;	Choice data record
struct { Gimode mode; Gesw esw; Gchoice choice; Gint pet; Glimit e_area; Gchoicerec record; } Gchoicest;	Choice state
enum { GCLIP, GNOCLIP } Gclip;	Clipping indicator
struct { Gclip ind; Glimit rec; } Gcliprect;	Clipping rectangle

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GCONDITIONALLY, GALWAYS } Gclrflag;	Clear control flag
enum { GCOLOUR, GMONOCHROME } Gcoavail;	Color availability
struct { Gfloat red; Gfloat green; Gfloat blue; } Gcobundl;	Color bundle
struct { Gint colours; Gcoavail coavail; Gint predefined; } Gcofac;	Color facilities
enum { GABSENT, GPRESENT } Gcovalid;	Color values
enum { GC_OD, GC_NOCHOICE, GC_NONE } Gcstat;	Choice status

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GWC, GNDC } Gcsw;	Coordinate switch
	D
struct { Gint choices; Gintlist *pets; Glimit e_area; Gchoicerec record; } Gdefchoice;	Default choice data
struct { Gdefmode defmode; Girgmode irgmode; Gdefer;	Deferral state
struct { Gpoint position; Gintlist *pets; Glimit e_area; Glocrec record; } Gdefloc;	Default locator data
enum { GASAP, GBNIG, GBNIL, GASTI } Gdefmode;	Deferral mode

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gintlist *pets; Glimit e_area; Gpickrec record; } Gdefpick;	Default pick data
struct { Gint bufsiz; Gintlist *pets; Glimit e_area; Gstringrec record; } Gdefstring;	Default string data
struct { Gint bufsiz; Gintlist *pets; Glimit e_area; Gstrokerec record; } Gdefstroke;	Default stroke data
struct { Gfloat value; Gintlist *pets; Glimit e_area; Gvalrec record; } Gdefval;	Default valuator data
enum { GDC_METRES, GDC_OTHER } Gdevunits;	Device coordinate units

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description

struct {
  Gfloat x_dim;
  Gfloat y_dim;
} Gdim;
	Dimension in real values

struct {
  Gdevunits units;
  Gpoint device;
  Gipoint raster;
} Gdspsize;
	Display size

enum {
  GEMPTY,
  GNOEMPTY
} Gdspsurf;
	Display surface

E	
union {	Escape input structure
Guesc_idatarec esc_idatarec; } Gescin;	
union {	Escape output structure
Guesc_odatarec esc_odatarec; } Gescout;	

enum {
  GECHO,
  GNOECHO
} Gesw;
	Echo switch

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint ws; Gint dev; Giclass class; } Gevent;	Event
struct { Gpoint concat; Gpoint corner_1; Gpoint corner_2; Gpoint corner_3; Gpoint corner_4; } Gextent;	Text extent
	F
FILE Gfile	File
struct { Gasf inter; Gasf style; Gasf colour; Gint fill; Gflbundl bundl; } Gflattr;	Fill area attributes
struct { Gflinter inter; Gint style; Gint colour; } Gflbundl;	Fill area bundle

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint n_interiors; Gflinter *interiors; Gintlist *hatches; Gint predefined; } Gflfac;	Fill area facilities
struct { GHOLLOW, GSOLID, GPATTERN, GHATCH } Gflinter;	Fill area interior style
Float Gfloat	Float
	G
struct { Gint n_attrs; Gattrs *attrs; } Ggdpfac;	GDP facilities
union { Gugdp_datarec gdp_datarec; } Ggdprec;	GDP data record
struct { Gchar *gksmrec; } Ggksmrec;	GKS metafile data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint type; Gint length; } Ggksmit;	GKS metafile item
	I
enum { GNCLASS, GLOCATOR, GSTROKE, GVALUATOR, GCHOICE, GPICK, GSTRING } Giclass;	Input device class
struct { Guint x_dim; Guint y_dim; } Gidim;	Dimension in integer values
enum { GREQUEST, GSAMPLE, GEVENT } Gimode;	Input device mode

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint ln_type; Gfloat ln_width; Gint ln_colour; Gint mk_type; Gfloat mk_size; Gint mk_colour; Gtxfp tx_fontprec; Gfloat ch_exp; Gfloat ch_space; Gint tx_colour; Gflinter fl_interior; Gint fl_style; Gint fl_colour; Gasfs asfs; } Gindivattr;	Individual attributes
enum { GSET, GREALIZED } Ginqtype;	Inquiry type
Gint int	Integer
struct { Gint number; Gint *integers; } Gintlist;	Integer list
struct { Gint x; Gint y; } Gipoint;	Integer point

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GSUPRESSED, GALLOWED } Girgmode;	Implicit regeneration mode
enum { GOK, GNONE } Gistat;	Request status
	L
enum { GLAST, GNOTLAST } Glastev;	Last event
struct { Gfloat xmin; Gfloat xmax; Gfloat ymin; Gfloat ymax; } Glimit;	Coordinate limits
struct { Gasf type; Gasf width; Gasf colour; Gint line; Glnbundl bundl; } Glnattr;	Polyline attributes

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint type; Gfloat width; Gint colour; } Glnbundl;	Polyline bundle
struct { Gintlist *types; Gint widths; Gfloat nom_width; Gfloat min_width; Gfloat max_width; Gint predefined; } Glnfac;	Polyline facilities
enum { GLN_SOLID, GLN_DASHED, GLN_DOTTED, GLN_DASHDOT GLN_TRIPLE_DOT, GLN_DOUBLE_DOT, GLN_SPACED_DOT, GLN_SPACED_DASH, GLN_LONG_SHORT_DASH, GLN_LONG_DASH, GLN_DASH_3_DOT, GLN_DASH_2_DOT, } Glntype;	Line type - nonstandard
struct { Gint transform; Gpoint position; } Gloc;	Request status

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gchar *data; } Glocpet0001;	Locator prompt and echo type 1 data record
struct { Gchar *data; } Glocpet0002;	Locator prompt and echo type 2 data record
struct { Gchar *data; } Glocpet0003;	Locator prompt and echo type 3 data record
struct { Gacf acf; Glnattr ln; } Glocpet0004;	Locator prompt and echo type 4 data record

struct {
enum Gpfcf pfcf;
Gacf acf;
union {
glnattr ln;
gflattr fl;
} attr;
Gchar *data;
} Glocpet0005;
	Locator prompt and echo type 5 data record

struct { Gchar *title_string; } Glocpet0006;	Locator prompt and echo type 6 data record
struct { Gfloat box_x; Gfloat box_y; } Glocpet_0001;	Locator prompt and echo type data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description

struct {
enum Gpfcf acf;
enum Gacf acf;
 	
union {
glnattr ln;
gflattr fl;
} attr;
} Glocpet_0002;
	Locator prompt and echo type data record

struct {
Gacf acf;
 	
union {
struct {
gpoint point1;
gpoint point2;
} echo;
struct {
glnattr ln;
gpoint point1;
gpoint point2;
} lnecho;
} attr;
} Glocpet_0003;
	Locator prompt and echo type data record

struct { Gacf acf; Glnattr ln; } Glocpet_0004;	Locator prompt and echo type data record
struct { Gacf acf; Glnattr ln; } Glocpet_0005;	Locator prompt and echo type data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description

struct {
Gacf acf;
 	
union {
struct {
gpoint point1;
gpoint point2;
} echo;
struct {
glnattr ln;
gpoint point1;
gpoint point2;
} lnecho;
} attr;
} Glocpet_0006;
	Locator prompt and echo type data record

struct {
Gacf acf;
 	
union {
struct {
gpoint point1;
gpoint point2;
} echo;
struct {
glnattr ln;
gpoint point1;
gpoint point2;
} lnecho;
} attr;
} Glocpet_0007;
	Locator prompt and echo type data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description

struct {
Gacf acf;
 	
union {
struct {
gpoint point1;
gpoint point2;
} echo;
struct {
glnattr ln;
gpoint point1;
gpoint point2;
} lnecho;
} attr;
} Glocpet_0008;
	Locator prompt and echo type data record

struct { Gacf acf; Glnattr ln; } Glocpet_0009;	Locator prompt and echo type data record
struct { Gacf acf; Glnattr ln; } Glocpet_00010;	Locator prompt and echo type data record
struct { Gchar *data; } Glocpet_00011;	Locator prompt and echo type data record
struct { Gacf acf; Glnattr ln; } Glocpet_00012;	Locator prompt and echo type data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
union {	Locator data record
Glocpet0001 locpet1_datarec; Glocpet0002 locpet2_datarec; Glocpet0003 locpet3_datarec; Glocpet0004 locpet4_datarec; Glocpet0005 locpet5_datarec; Glocpet0006 locpet6_datarec; }	
Glocpet_0001 locpet_1_datarec; Glocpet_0002 locpet_2_datarec; Glocpet_0003 locpet_3_datarec; Glocpet_0004 locpet_4_datarec; Glocpet_0005 locpet_5_datarec; Glocpet_0006 locpet_6_datarec; Glocpet_0007 locpet_7_datarec; Glocpet_0008 locpet_8_datarec; Glocpet_0009 locpet_9_datarec; Glocpet_00010 locpet_10_datarec; Glocpet_00011 locpet_11_datarec; Glocpet_00012 locpet_12_datarec; } Glocrec;	
struct {	Locator state
Gimode mode; Gesw esw; Gloc loc; Gint pet; Glimit e_area; Glocrec record; } Glocst;	
Glong long	Long

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
	M
struct { Gasf type; Gasf size; Gasf colour; Gint mark; Gmkbundl bundl; } Gmkattr;	Polymarker attributes
struct { Gint type; Gfloar size; Gint colour; } Gmkbundl;	Polymarker bundle
struct { Gintlist *types; Gint sizes; Gfloat nom_size; Gfloat min_size; Gfloat max_size; Gint predefined; } Gmkfac;	Polymarker facilities

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GMK_POINT, GMK_PLUS, GMK_STAR, GMK_O, GMK_STAR_X, GMK_SOLID_DIAMOND, GMK_DIAMOND, GMK_SOLID_HGLASS, GMK_HOURGLASS, GMK_SOLID_BOWTIE, GMK_BOWTIE, GMK_SOLID_SQUARE, GMK_SQUARE, GMK_SOLID_TRI_DOWN, GMK_TRIANGLE_DOWN, GMK_SOLID_TRI_UP, GMK_TRIANGLE_UP, GMK_SOLID_CIRCLE, GMK_X } Gmktype;	Marker type
struct { Gmodtype transform; Gmodtype appear; Gmodtype disappear; Gmodtype highlight; Gmodtype priority; Gmodtype addition; Gmodtype deletion; } Gmodseg;	Dynamic segment attribute modification
enum { GIRG, GIMM } Gmodtype;	Dynamic modification type

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gmodtype line; Gmodtype mark; Gmodtype text; Gmodtype fill; Gmodtype pat; Gmodtype colour; Gmodtype wstran; } Gmodws;	Dynamic workstation attribute modification
	N
enum { GNO, GYES } Gnframe;	Frame action at update
struct { Gint locator; Gint stroke; Gint valuator; Gint choice; Gint pick; Gint string; } Gnumdev;	Input devices
	O
enum { GGKCL, GGKOP, GWSOP, GWSAC, GSGOP } Gopst;	GKS operating state

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
	P
enum { GPF_POLYLINE, GPF_FILLAREA } Gpfcf;	Fill area control flag
struct { Gpstat status; Gint seg; Gint pickid; } Gpick;	Pick data
struct { Gfloat aperture; } Gpickpet0001;	Pick prompt and echo type 1 data record
struct { Gfloat aperture; } Gpickpet0002;	Pick prompt and echo type 2 data record
struct { Gfloat aperture; } Gpickpet0003;	Pick prompt and echo type 3 data record
union { Gpickpet0001 pickpet1_datarec; Gpickpet0002 pickpet2_datarec; Gpickpet0003 pickpet3_datarec; } Gpickrec;	Pick data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gimode mode; Gesw esw; Gpick pick; Gint pet; Glimit e_area; Gpickrec record; } Gpickst;	Pick state
struct { Gfloat x; Gfloat y; } Gpoint;	Coordinate point
enum { GRPOFF, GRPON } Gprflag;	Prompt flag
struct { Gint ln_index; Gint mk_index; Gint tx_index; Gfloat ch_height; Gpoint ch_up; Gfloat ch_width; Gpoint ch_base; Gtxpath tx_path; Gtxalign tx_align; Gint fl_index; Gpoint pa_width; Gpoint pa_height; Gpoint pa_refpt; } Gprimattr;	Primitive attributes

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GP_OK, GP_NOPICK, GP_NONE } Gpstat;	Pick status
struct { Gipoint size; Gint *array; } Gptbundl;	Pattern bundle
	Q
struct { Gcstat status; Gint choice; } Gqchoice;	Request choice
struct { Gistat status; Gloc loc; } Gqloc;	Request locator
struct { Gpstat status; Gint seg; Gint pickid; } Gqpick;	Request pick
struct { Gistat status; Gchar *string; } Gqstring;	Request string

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gistat status; Gstroke stroke; } Gqstroke;	Request stroke
struct { Gint ws; Giclass class; Gint devno; } Gqueue;	Queue information
struct { Gistat status; Gfloat val; } Gqval;	Request valuator
R	
struct { Gpoint ul; Gpoint lr; } Grect;	Coordinate rectangle
enum { GPERFORM, GPOSTPONE } Gregen;	Regeneration flag
S	
struct { Gfloat x_scale; Gfloat y_scaleesw; Gscale;	Scale vector

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gfloat segtran[2][3]; Gsegvis segvis; Gseghi seghi; Gfloat segpri; Gsegdet segdet; } Gsegattr;	Segment attribute values
enum { GUNDETECTABLE, GDETECTABLE } Gsegdet;	Segment detectability
enum { GNORMAL, GHIGHLIGHTED } Gseghi;	Segment highlighting
enum { GVISIBLE, GINVISIBLE } Gsegvis;	Segment visibility
enum { GNOMORE; GMORE; } Gsimultev;	Simultaneous events
struct { Gint bufsiz; Gint position; Gchar *title_string; } Gstringpet0001;	String prompt and echo type 1 data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gstringpet0001 stringpet1_datarec; } Gstringrec;	
struct { Gimode mode; Gesw esw; Gchar *string; Gint pet; Glimit e_area; Gstringrec record; } Gstringst;	String state
struct { Gint n_points; Gchar **strings } Gstrlist;	String list
struct { Gint transform; Gint n_points; Gpoint *points; } Gstroke;	Stroke data
struct { Gint bufsiz; Gint editpos; Gpoint interval; Gfloat time; Gchar *data; } Gstrokepet0001;	Stroke prompt and echo type 1 data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint bufsiz; Gint editpos; Gpoint interval; Gfloat time; } Gstrokepet0002;	Stroke prompt and echo type 2 data record
struct { Gint bufsiz; Gint editpos; Gpoint interval; Gfloat time; enum Gacf acf; Gmkattr mk; } Gstrokepet0003;	Stroke prompt and echo type 3 data record
struct { Gint bufsiz; Gint editpos; Gpoint interval; Gfloat time; Gacf acf; Gmkattr mk; enum Gacf acf; Glnattr ln; } Gstrokepet0004;	Stroke prompt and echo type 4 data record
struct { Gstrokepet0001 strokepet1_datarec; Gstrokepet0002 strokepet2_datarec; Gstrokepet0003 strokepet3_datarec; Gstrokepet0004 strokepet4_datarec; } Gstrokerec;	Stroke data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gimode mode; Gesc esw; Gstroke stroke; Gint pet; Glimit e_area; Gstrokerec record; } Gstrokest;	Stroke state
T	
struct { Glimit w; Glimit v; } Gtran;	Transformation
struct { Gtxhor hor; Gtxver ver; } Gtxalign;	Text alignment
struct { Gasf fp; Gasf exp; Gasf space; Gasf colour; Gint text; Gtxbundl bundl; } Gtxattr;	Text attributes

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gtxfp fp; Gfloat exp; Gfloat space; Gint colour; } Gtxbundl;	Text bundle
struct { Gint fps; Gtxfp *fp_list; Gint heights; Gfloat min_ht; Gfloat max_ht; Gint expansions; Gfloat min_exp; Gfloat max_exp; Gint predefined; } Gtxfac;	Text facilities
struct { Gint font; Gtxprec prec; } Gtxfp;	Text facilities
enum { GAH_NORMAL, GAH_LEFT, GAH_CENTRE, GAH_RIGHT } Gtxhor;	Horizontal text alignment component

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GTP_RIGHT, GTP_LEFT, GTP_UP, GTP_DOWN } Gtxpath;	Text path
enum { GP_STRING, GP_CHAR, GP_STROKE } Gtxprec;	Text precision
enum { GAV_NORMAL; GAV_TOP; GAV_CAP; GAV_HALF; GAV_BASE; GAV_BOTTOM; } Gtxver;	Vertical text alignment component
	U
struct { Gint number_integer; Gint number_float; Gint number_strings; Gint *list_integers Gfloat *list_floats; Gint *list_string_lengths; Gchar **list_strings } Guesc_idatarec;	Escape dependent input data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint number_integer; Gint number_float; Gint number_strings; Gint *list_integers Gfloat *list_floats; Gint *list_string_lengths; Gchar **list_strings } Guesc_odatarec;	Escape dependent output data record
struct { Gint number_integer; Gint number_float; Gint number_strings; Gint *list_integers Gfloat *list_floats; Gint *list_string_lengths; Gchar **list_strings } Gugdp_datarec;	GDP dependent data record
Guint int	Unsigned integer
V	
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet0001;	Valuator prompt and echo type 1 data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet0002;	Valuator prompt and echo type 2 data record
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet0003;	Valuator prompt and echo type 3 data record
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet_1_datarec;	Valuator prompt and echo type 3 data record
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet_2_datarec;	Valuator prompt and echo type 3 data record
struct { Gfloat low; Gfloat high; Gchar *title_string; } Gvalpet_3_datarec;	Valuator prompt and echo type 3 data record

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
union { Gvalpet0001 valpet1_datarec; Gvalpet0002 valpet2_datarec; Gvalpet0003 valpet3_datarec; Gvalpet_0001 valpet_1_datarec; Gvalpet_0002 valpet_2_datarec; Gvalpet_0003 valpet_3_datarec; } Gvalrec;	Valuator data record
struct { Gimode mode; Gesw esw; Gfloat val; Gint pet; Glimit e_area; Gvalrec record; } Gvalst;	Valuator state
enum { GHIGHER, GLOWER } Gvpri;	Viewport Priority
	W
enum { GOUTPUT, GINPUT, GOUTIN, GWISS, GMO, GMI } Gwscat;	Workstation category

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
enum { GVECTOR, GRASTER, GOTHER } Gwsclass;	Workstation class
struct { Gconn *conn; Gwstype *type; } Gwsct;	Workstation connection and type
struct { Gdefmode defmode; Gdspsurf dspsurf; Girgmode irgmode; Gnframe nframe; } Gwsdus;	Workstation deferral and update state
struct { Gint open; Gint active; Gint assoc; } Gwsmax;	Workstation maximum numbers
enum { GINACTIVE, GACTIVE } Gwsstate;	Workstation state

(continued on next page)

Table B–1 (Cont.): C Binding Type Definitions

Type Definition	Description
struct { Gint line; Gint mark; Gloc text; Gint fill; Gint pat; Gint colour; } Gwstables;	Length of workstation tables
struct { Gwstus wstus; Gtran request; Gtran current; } Gwsti;	Workstation transformation information
enum { GNOTPENDING, GPENDING } Gwstus;	Workstation transformation update state

Appendix C

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:

```
        .
        .
        .
  /*  Include the error symbol definitions file. */
#include <gksmsgs.h>

  /*  Check for success. */
      if ( gopenws (wsid, con_id, ws_type ) == GKS$_SUCCESS )
      {
         .
         .
         .
      }
      .
      .
      .
  /*  Check for an invalid workstation identifier */
      if ( gactivatews (ws_id ) == GKS$_ERROR_20 )
      {
         .
         .
         .
      }
      .
      .
      .
```

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

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

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

Some of the DEC GKS specific error messages substitute program information in the message text. In this appendix, the portion of the text to be substituted is shown as ****.

The following sections describe the DEC GKS error messages by category.

C.1 Implementation-Specific Errors

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

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

DECGKS$_ERROR_NEG_2:

Error: Specified color map is too large.

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

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

DECGKS$_ERROR_NEG_3:

Error: Workstation description file contains invalid data.

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

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

DECGKS$_ERROR_NEG_4:

Error: The high word 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 of your choice string array. For more information, refer to the program example for GKS$INQ_DEF_CHOICE_DATA in Chapter 12, Inquiry Functions, in the *DEC GKS Reference Manual.*

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

DECGKS$_ERROR_NEG_6:

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

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

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

DECGKS$_ERROR_NEG_7:

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

User Action: You can use ginqdefchoice 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 GBUNDLED (0) or GINDIVIDUAL (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 GHOLLOW (0), GSOLID (1), GPATTERN (2), or GHATCH (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 GAH_NORMAL (0), GAH_LEFT (1), GAH_CENTER (2), or GAH_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 GAV_NORMAL (0), GAV_TOP (1), GAV_CAP (2), GAV_HALF (3), GAV_BASE (4), or GAV_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 GP_STRING (0), GP_CHAR (1), or GP_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 GTP_RIGHT (0), GTP_LEFT (1), GTP_UP (2), or GTP_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 GNOECHO (0) or GKSECHO (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 GSET(0) or GREALIZED(1) when calling an inquiry function.

User Action: Check the value type argument to make sure that it is either GSET(0) or GREALIZED(1).

–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 GERRORLOG.

User Action: Make sure that the error code passed to GERRORLOG 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 gevaltran and to gaccumtran to make sure that you passed valid transformation components. Also, make sure that you specified a transformation matrix to gsetsegtran or to ginsertseg.

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

DECGKS$_ERROR_NEG_23:

User Action: Make sure that you passed either the value GNOCLIP (0) or GCLIP (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 GHIGHER (0) or GLOWER (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 GPOSTPONE (0) or GPERFORM (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 GASAP (0), GBNIG (1), GBNIL (2), or GASTI (3).

–27 Invalid value specified for regeneration mode in routine ****

DECGKS$_ERROR_NEG_27:

User Action: Make sure that you passed either the value GSUPPRESSED (0) or GALLOWED (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 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 chapter in the *DEC GKS Device Specifics Reference Manual* to determine if the specified font is supported on your device. If you are not using a DEC GKS supported graphics handler, make sure that your handler defines the proper logical names, and that the logicals reference a valid file.

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

DECGKS$_ERROR_NEG_33:

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

User Action: Make sure that you have passed the array by descriptor and that you fill the descriptor with valid values. If you have, and you use an inquiry function to initialize the array variable, make sure that all of the arguments are specified to the inquiry function in the correct order. Also, if the array is passed to the CELL ARRAY function, make sure that you have declared a two-dimensional array.

–34 String length less 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)™ .

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

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

DECGKS$_ERROR_NEG_37:

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

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

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

DECGKS$_ERROR_NEG_38:

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

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

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

DECGKS$_ERROR_NEG_39:

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

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

DECGKS$_ERROR_NEG_40:

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

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

DECGKS$_ERROR_NEG_41:

Error: You illegally defined a workstation description table entry.

User Action: Check your workstation description table definitions for your graphics handler.

–42 Logical name for the list of workstation types, GKS$LIST_TYPES, could not be translated in routine ****

DECGKS$_ERROR_NEG_42:

Error: You improperly defined the logical name.

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

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

DECGKS$_ERROR_NEG_43:

Explanation: Check to make sure either that you specify the correct workstation type when opening a non-VAXstation workstation, or that you passed a correct workstation type value to one of the workstation description table or state list inquiry functions. If you are working on a MicroVAX, make sure that you install the VAXstation Windowing Software.

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

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

DECGKS$_ERROR_NEG_90:

Fatal: DEC GKS memory data structures were corrupted.

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_91:

Fatal: A DEC GKS internal data structure was corrupted.

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_92:

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

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

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

DECGKS$_ERROR_NEG_93:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_94:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_95:

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

User Action: If you generate this error using a user-written graphics handler, make sure that the value of the local storage area is a valid value.

–96 Internal GKS error: Illegal number of points to device handler for rectangular polygon in routine ****

DECGKS$_ERROR_NEG_96:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_97:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_98:

Fatal: You may have recursively defined a logical name.

User Action: Check the currently defined logical names to see if all are properly defined. If you cannot locate an error, submit an SPR.

–99 Internal GKS error: Unable to reduce number of points in fill area to required limit in routine ****

DECGKS$_ERROR_NEG_99:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_100:

Fatal:

User Action: Submit an SPR.

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

DECGKS$_ERROR_NEG_150:

User Action:

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

DECGKS$_ERROR_NEG_151:

User Action:

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

DECGKS$_ERROR_NEG_152:

User Action:

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

DECGKS$_ERROR_NEG_153:

User Action:

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

DECGKS$_ERROR_NEG_154:

User Action:

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

DECGKS$_ERROR_NEG_155:

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

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

DECGKS$_ERROR_NEG_156:

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

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

DECGKS$_ERROR_NEG_157:

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

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

DECGKS$_ERROR_NEG_158:

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

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

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

DECGKS$_ERROR_NEG_159:

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

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

DECGKS$_ERROR_NEG_160:

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

User Action: Pass addresses of larger array buffers in the last four components of the escape data record.

–161 Specified bounding box is too small in routine ****

DECGKS$_ERROR_NEG_161:

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

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

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

DECGKS$_ERROR_NEG_300:

User Action: Make sure that you specify either GNORMAL (0) or GHIGHLIGHTED (1).

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

DECGKS$_ERROR_NEG_301:

User Action: Make sure that you specify either GINVISIBLE (0) or GVISIBLE (1).

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

DECGKS$_ERROR_NEG_302:

User Action:

User Action: Make sure that you specify either GUNDETECTABLE (0) or GDETECTABLE (1).

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

DECGKS$_ERROR_NEG_303:

User Action:

–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:

User Action:

C.2 Operating State Errors

This section lists the errors that result when you call a function that is not permitted in the current operating state. For a list of the functions that you can or cannot call in a given DEC GKS operating state, refer to Chapter 4, Control Functions, in the *DEC GKS Reference Manual.*

–306 The definition of non-numeric values in routine is invalid ****

GKS$_ERROR_NEG_306:

Error:

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

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 gclosews before the current DEC GKS state changes to GGKCL.)

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 gopengks or gclosegks before the DEC GKS state changes to GGKOP.)

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 gactivatews or gcloseseg before the DEC GKS state changes to GWSAC.)

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 gcreateseg before the DEC GKS state changes to GSGOP.)

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 gactivatews before the DEC GKS state changes to GWSAC.)

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 gopenws before the DEC GKS state changes to GWSOP.)

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 gopenws before the DEC GKS state changes to GWSOP.)

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 gopenws before the DEC GKS state changes to GWSOP.)

C.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 ginqwsconntype 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 gopenws, 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 gksopenws 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 gksopenws, 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 gopenws, 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 gactivatews, 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 gactivatews 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 gopenws.

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 gopenws.

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 gopenws.

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 gopenws.

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 gopenws.

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 gopenws.

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 gopenws.

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 ggdp, or make sure that you passed the correct ggdpidentifier.

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

GKS$_ERROR_42:

User Action: You must remove the function call to gopenws. You can use ginqwsmaxnum 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 gactivatews. You can use ginqwsmaxnum to determine the maximum number of active workstations that DEC GKS supports.

C.4 Transformation Errors

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

50 Transformation number is invalid in routine ****

GKS$_ERROR_50:

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

51 Rectangle definition is invalid in routine ****

GKS$_ERROR_51:

Error: Either the normalization window or viewport is invalid.

User Action: Either make sure that you have not reversed the order of the X and Y argument values, that your coordinate values form a valid rectangle, and that your coordinate values are real numbers.

52 Viewport is not within the Normalized Device Coordinate unit square in routine ****

GKS$_ERROR_52:

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

User Action: Redefine the function normalization viewport.

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

GKS$_ERROR_53:

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

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

GKS$_ERROR_54:

User Action: Either make sure that you have not reversed the order of the X and Y argument values, that your coordinate values form a valid rectangle, and that your coordinate values are real numbers. You can use the function ginqdisplaysize to determine the maximum X and Y values of the device coordinate plane.

C.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 gsetlinerep 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 gsetlinerep to define a representation for the index, or use another, predefined index value.

63 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 ginqlinefacil 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 gsetmarkerrep 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 gsetmarkerrep to define a representation for a given index, or use another, predefined index value.

69 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 ginqmarkerfacil 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 gsettextrep 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 gsettextrep 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 gsetfillrep 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 gsetfillrep 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 ginqfillfacil 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 ginqfillfacil 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 gsetpatrep 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 gsetpatrep 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 gsetfillintstyle.

91 Dimensions of color array are invalid in routine ****

GKS$_ERROR_91:

Error: One or more of the arguments passed to gcellarray 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 gsetcolourrep 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 gsetcolourrep 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 gsetpickid 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.

C.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 (GDP) 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 (GDP) 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 (GDP) in routine ****

GKS$_ERROR_104:

User Action: Deactivate the workstations that do not generate the GDPs, or redefine the GDP data record so that all of the workstations can generate the primitive.

105 At least one active workstation is not able to generate the specified Generalized Drawing Primitive (GDP) under the current transformations 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.

C.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 gcreateseg 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:

Error: You attempted to copy, associate, or insert a segment that is not stored in WISS.

User Action: 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 gcreateseg 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.

C.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 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 ginqinputoverflow when the queue was not full, and had not been filled since the beginning of your application.

User Action: Allow the user to 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 ginqinputoverflow when the queue was full, but since the workstation is closed, information about the overflow is not available.

User Action: You can set the devices to request mode (removing their prompts from the workstation surface), and then you can either process reports from the queue until empty or you can flush the queue of all reports.

150 No input value of the correct class is in the current event report in routine ****

GKS$_ERROR_150:

User Action: Make sure that you check the input class argument passed to gawaitevent 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 gawaitevent is a real value between 0.0 and 356,400, specified in the format described in the gawaitevent function description in Chapter 8, Input Functions, in the *DEC GKS Reference Manual.*

152 Initial value is invalid in routine ****

GKS$_ERROR_152:

User Action: Either check to make sure that you specified the correct value, or check the capabilities of the device to see if you requested a value unsupported by the device. If you obtained the value from an inquiry function, check the order and number of arguments specified to the inquiry function.

153 Number of points in the initial stroke is greater than the buffer size in routine ****

GKS$_ERROR_153:

User Action: Either increase the size of the buffer or reduce the number of points in the initial stroke.

154 Length of initial string is greater than the buffer size in routine ****

GKS$_ERROR_154:

User Action: Either increase the size of the buffer or decrease the size of the initial string.

C.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.

C.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.

C.11 Miscellaneous Errors

This section lists the DEC GKS miscellaneous errors.

200 Specified error file in 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.

C.12 System Errors

This section lists implementation-dependent errors.

300 Storage overflow has occured 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 occured in segment storage ****

GKS$_ERROR_301:

User Action: Either remove the index, or check the order and the number of the arguments. If you used an inquiry function to obtain the index value, check the order and the number of the arguments passed to the inquiry function.

302 Input/Output error has occurred while reading in routine ****

GKS$_ERROR_302:

Error: You specified an illegal metafile for a metafile input workstation.

User Action: Make sure that you work with a valid GKSM metafile, and that you correctly specify the connection identifier.

303 Input/Output error has occurred while writing in routine ****

GKS$_ERROR_303:

Error: You specified an illegal metafile for a metafile output workstation.

User Action: Make sure that you work with a valid GKSM metafile, and that you correctly specify the connection identifier.

304 Input/Output error has occurred while sending data to a workstation ****

GKS$_ERROR_304:

User Action:

305 Input/Output error has occurred while receiving data from a workstation ****

GKS$_ERROR_305:

User Action:

306 Input/Output error has occurred during program library management ****

GKS$_ERROR_307:

User Action:

307 Input/Output error has occurred while reading workstation description table ****

GKS$_ERROR_307:

User Action:

308 Arithmetic error has occurred in routine ****

GKS$_ERROR_308:

Error: You either divided by zero or caused data overflow.

User Action: Check the arguments passed in the function call.

C.13 C Binding Errors

This section lists those error messages that are specific to the C binding functions.

2200 Buffer overflow on input or inquiry functions ****

GKS$_ERROR_2000 (EBUFSPAC):

Error: Buffer is not large enough to accept returned data.

User Action: Increase buffer size and call again.

2201 Error log is empty ****

GKS$_ERROR_2001:

Error:

User Action:

2202 Out of Range ****

GKS$_ERROR_2002 (EOUTRANG):

Error: The requested data range does not exist.

User Action: Modify the request to within the range.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Reader's Comments

DEC GKS C Binding Reference Manual
AA–MJ30A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

Reader's Comments

DEC GKS C Binding Reference Manual
AA–MJ30A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Cut Along Dotted Line