Digital GKS

Operating System Information for the GKS$ Binding

January 1996

This document describes Digital GKS™ (formerly DEC GKS™) programming considerations for different operating systems.

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, May 1992
Revised, February 1993
Revised, January 1996

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

© Digital Equipment Corporation 1992, 1993, 1996.

The following are trademarks of Digital Equipment Corporation: DDIF, DEC, DEC C, DEC Fortran, DEC GKS, DEC OSF/1, DECwindows, Digital GKS, LA75, OpenVMS, VAX, and the DIGITAL logo.

Motif and OSF/Motif are registered trademarks of Open Software Foundation, Inc.

PostScript is a registered trademark of Adobe Systems Incorporated.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Ltd.

ZK5676

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface v

1 Digital UNIX Programming Considerations

1.1 Including Definition Files 1–1
1.2 Compiling, Linking, and Executing Your Programs 1–2
1.2.1 Compiling and Linking Your Program 1–2
1.2.2 Executing Your Program 1–2
1.3 Opening a Workstation 1–3
1.3.1 Specifying the Workstation Connection Identifier 1–3
1.3.2 Specifying the Workstation Type 1–3
1.4 Digital GKS Environment Variables 1–3
1.5 Error Handling 1–8
1.5.1 Error Codes 1–8
1.5.2 Error Files 1–9
1.6 Configuration Files 1–9
1.6.1 Customizing the Configuration File at System Level 1–9
1.6.2 Customizing the Configuration File at User Level 1–10

2 OpenVMS Programming Considerations

2.1 Including Definition Files 2–1
2.2 Compiling, Linking, and Executing Your Programs 2–2
2.2.1 Linking Your Program 2–2
2.2.2 Executing Your Program 2–2
2.3 Opening a Workstation 2–3
2.3.1 Specifying the Workstation Connection Identifier 2–3
2.3.2 Specifying the Workstation Type 2–3
2.4 Digital GKS Logical Names 2–3
2.5 Error Handling 2–8
2.5.1 Error Codes 2–8
2.5.2 Error Files 2–8

3 ULTRIX Programming Considerations

3.1 Including Definition Files 3–1
3.2 Compiling, Linking, and Executing Your Programs 3–2
3.2.1 Compiling and Linking Your Program 3–2
3.2.2 Executing Your Program 3–3
3.3 Opening a Workstation 3–3
3.3.1 Specifying the Connection Identifier 3–3
3.3.2 Specifying the Workstation Type 3–3
3.4 Digital GKS Environment Variables 3–4
3.5 Error Handling 3–8

3.5.1 Error Codes 3–8
3.5.2 Error Files 3–9
3.6 Configuration Files 3–9
3.6.1 Customizing the Configuration File at System Level 3–9
3.6.2 Customizing the Configuration File at User Level 3–10

Index

Tables

1–1 General Environment Variables for Digital GKS 1–6
2–1 General Logical Names for Digital GKS 2–4
3–1 General Environment Variables for Digital GKS 3–6

Preface

This manual describes programming considerations for using Digital GKS on Digital UNIX® (formerly DEC OSF/1™), OpenVMS™, and ULTRIX™ operating systems. This manual is to be used in conjunction with the Digital GKS binding manuals.

Intended Audience

This manual is for programmers who have experience developing graphics applications in one of the languages supported by Digital GKS. They also should be familiar with the principles of programming Digital GKS, as described in the *DEC GKS User Manual*.

Conventions

The following conventions are used in this manual:

Convention	Meaning
RETURN	The symbol RETURN represents a single stroke of the Return key on a terminal.
UPPERCASE TEXT	Uppercase text indicates a Digital GKS function or symbol name.
. . .	A vertical ellipsis indicates that not all of the text of a program or program output is illustrated. Only relevant material is shown in the example.
. . .	A horizontal ellipsis indicates that additional arguments, options, or values can be entered. A comma preceding the ellipsis indicates that successive items must be separated by commas. Horizontal ellipses in illustrations indicate that there is information not illustrated that either precedes or follows the information included in the illustration itself.
[ ]	Square brackets, in function synopses and a few other contexts, indicate that a syntactic element is optional.
Digital GKS	References to Digital GKS apply to the Digital GKS (formerly DEC GKS) product.
VMS™ OpenVMS ™	The terms VMS and OpenVMS refer collectively to the OpenVMS Alpha and OpenVMS VAX operating systems, unless otherwise specified.
Digital UNIX	References to Digital UNIX apply to the Digital UNIX (formerly DEC OSF/1) operating system.

1

Digital UNIX Programming Considerations

The specific method for using Digital GKS software depends on the features and conventions of each programming language. This chapter describes general issues that must be considered when using the Digital UNIX™ C (cc) compiler with Digital GKS.

The information contained in this chapter was correct when the manual was printed; however, the information may have been changed. For the most up-to-date information on using Digital GKS on Digital UNIX systems, see the following files:

/usr/lib/GKS/doc/decgks_gbind_op_spec.ps
/usr/lib/GKS/doc/decgks_gbind_op_spec.txt

1.1 Including Definition Files

You use Digital GKS software primarily by placing calls to Digital GKS functions in your program. You also need statements in your program other than the calls to Digital GKS functions. These statements make Digital GKS constants and their values accessible to all programs that call Digital GKS functions, regardless of the programming language you use.

Your installation kit includes several definition files that contain Digital GKS constants, and error message definition files that contain Digital GKS completion status code constants. All high-level programming languages that use Digital GKS enable you to insert these external, definition files into the program source code stream at compile time, thus making the constants available.

In the case of the C language, an `#include` statement is provided for inserting a definition file into a program. For example, any C program that uses the Digital GKS GKS$ binding should contain the following line:

```
#include <GKS/gksdefs.h>
```

The language definition files and error message definition files are:

Language	Language Definition Files	Error Message Definition File
ADA	/usr/include/GKS/gksdefs.ada	/usr/include/GKS/gksmsgs.ada
C	/usr/include/GKS/gksdefs.h /usr/include/GKS/gksdescrip.h	/usr/include/GKS/gksmsgs.h
Fortran	/usr/include/GKS/gksdefs.for	/usr/include/GKS/gksmsgs.for
Pascal	/usr/include/GKS/gksdefs.pas /usr/include/GKS/gksdefs.pen	/usr/include/GKS/gksmsgs.pas /usr/include/GKS/gksmsgs.pen

Each definition file includes comments describing its use.

1.2 Compiling, Linking, and Executing Your Programs

A program that uses Digital GKS function calls is compiled, linked, and executed like any other program. To compile and link a program, you should use the compile command appropriate for the processor you are using. To execute the program image, you type the executable program name. The following sections describe how to compile, link, and execute programs on UNIX systems.

Note

Digital GKS and the Digital GKS language bindings are supplied both as shared libraries (*.so* files) and as archive libraries (*.a* files). Linking with the shared libraries results in a substantial savings in disk space.

1.2.1 Compiling and Linking Your Program

You can compile and link a Digital GKS C program in one of two ways:

- **using shared libraries**

  The command is:

  ```
  # cc -I/usr/include/GKS -o program program.c \ RETURN
  -lGKSbnd -lGKS -lm -lc RETURN
  ```

  If your program makes calls to Motif or the X Window System, you may need to specify additional libraries as shown in the command line for linking with archive libraries.

- **using archive libraries**

  The command is:

  ```
  # cc -I/usr/include/GKS -o program program.c [gksconfig.c] -non_shared \ RETURN
  -lGKSbnd -lGKS -lMrm -lXm -lDXm -lXt -lcda -lX11 \ RETURN
  -llmf -lm -lc RETURN
  ```

  A workstation or device handler can be deliberately excluded from the executable image to decrease image size and link time. This is done by customizing the configuration file and specifying the customized version in the `link` command. The installed version of the configuration file is `/usr/lib/GKS/gksconfig.c`.

  The `gksconfig.c` file is optional, and is useful only when linking with the archive libraries. See Section 3.6 for more information on how to use configuration files.

  The options to the `link` command are required if certain device handlers are included in the configuration file:

  - The libraries `[-lMrm, -lXm, -lDXm, -lXt, -lX11]`, and `[-lc]` are required for the Motif® device handler.
  - The library `[-lcda]` is required for the DDIF™ device handler.

1.2.2 Executing Your Program

Once you have compiled and linked your program, you can execute it by typing the program file name at the command line:

```
$ program RETURN
```

1.3 Opening a Workstation

The following sections contain information on opening a workstation, which comprises two steps:

1. Specifying the workstation connection identifier
2. Specifying the workstation type

1.3.1 Specifying the Workstation Connection Identifier

An application can specify the connection by passing the connection identifier (ID) to the OPEN WORKSTATION function in any of the following ways:

- *By passing an environment variable specifying the connection ID.*
- *By passing a file or device name.*

  Pass the connection ID as a string, by descriptor.
- *By passing the value 0 or a null string.*

  Digital GKS attempts to translate the environment variable GKSconid. If no translation exists, Digital GKS uses GKS_default.output, which specifies a file in the current directory as the connected device. You can specify an environment option in one of two ways:

  – Define the environment variable GKSconid to be a file name or device name, depending on the workstation type.

  – Use the user defaults file `~/.GKSdefaults` or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

1.3.2 Specifying the Workstation Type

An application can specify the workstation type to the OPEN WORKSTATION function in either of the following ways:

- *Use the Digital GKS workstation types.*

  Pass any of the workstation types defined in the language file `/usr/include/GKS/gksdefs.*`, where `*` is the extension of the respective programming language.
- *Use the default workstation type by passing a value of 0.*

  Digital GKS attempts to translate the environment variable GKSwstype. If no translation exists, Digital GKS uses the workstation type 61 (PostScript). You can do either of the following:

  – Define the environment variable GKSwstype to be any supported workstation type.

  – Use the user defaults file `~/.GKSdefaults` or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

1.4 Digital GKS Environment Variables

In Digital GKS, a number of environment variables are interpreted at run-time. These environment variables allow the application to tailor Digital GKS to the needs of the application itself or a specific device. Each environment variable controls some aspect of the Digital GKS run-time environment.

Digital UNIX Programming Considerations
1.4 Digital GKS Environment Variables

Defining Environment Variables

You should set all environment variables before starting a Digital GKS session. The variable values remain constant throughout the session; modifying the value of an environment variable during a session may or may not have an effect on Digital GKS operation.

On Digital UNIX systems, environment variables are defined in a file named `.GKSdefaults` in your login directory, or in the `/usr/lib/GKS/.GKSdefaults` system file. The environment variable names are case sensitive; however, their associated values can be either uppercase or lowercase.

The syntax for defining environment variables in the `.GKSdefaults` file is:

```
GKSconid          :  gks_default.output    # connection id, device, or file name
GKSwstype         :  61                    # workstation type (PostScript)
```

Environment variables can also be defined at the shell level. To define an environment variable under the *C shell* (*csh*), use the following syntax:

```
# setenv GKSenvironment_variable value
```

To define an environment variable under the *Bourne shell* (*sh*), use the following syntax:

```
# GKSenvironment_variable=value
```

How Digital Searches for Environment Variables

When Digital GKS processes environment variables, it searches for the environment variable definitions in three locations. Once the environment variables are found, the search stops. The locations are searched in the following order:

1. User-specific Digital UNIX environment variables.
2. User-specific environment variables defined in the file `~/.GKSdefaults`. Digital recommends that you define the environment variables in this file.
3. System-wide environment variables defined in the file `/usr/lib/GKS/.GKSdefaults`.

The Default Environment Variable File

The default environment variable file `.GKSdefaults` is as follows:

```
!
! Digital GKS User-specific Environment Variables
! ================================================
!
GKSconid          :  gks_default.output    # connection id, device, or file name
GKSwstype         :  61                    # workstation type (PostScript)
GKSerror          :  on                    # on or off
GKSasf            :  individual            # bundled or individual
GKSndc_clip       :  on                    # on or off
GKSerrfile        :  stderr                # device or file name
GKSmetafile_type  :  gks3
GKSstroke_font1   :  /usr/lib/GFX/font/gfx_font_neg1 # Stroke font 1
!
! file : /usr/lib/GKS/.GKSdefaults
!
```

```
! Digital GKS System-Wide Environment Variables
! ==========================================
!
! Environment variables allow you to customize the Digital GKS environment
! to suit your needs.
!
! i)  Modify the Digital GKS system wide default settings.
!
!       Edit this file to change Digital GKS system-wide environment variable
!       default settings.
!
! ii) Modify the Digital GKS user-specific environment variable default settings
!     using the ~/.GKSdefaults file.
!
!       Copy this file into your login account (i.e., ~/.GKSdefaults) and
!       modify it to suit your needs.
!
! iii) Modify the Digital GKS user-specific settings using Digital UNIX
!      environment variables.
!
!      For example : setenv GKSwstype 231
!
!
!
! The Digital GKS search order for environment variable translation is:
! --------------------------------------------------------------------
!
! 1) User-specific Digital UNIX environment variables
!
! 2) User-specific environment variables defined in the file ~/.GKSdefaults
!
! 3) System-wide environment variables defined in the file
!    /usr/lib/GKS/.GKSdefaults
!
!
! The allowed syntax in this file is:
! ----------------------------------
!
!       Comments start with  ! or  #  character
!       Space, tabs, and  "  characters are ignored
!       Associations are done by the  =  (equal) or  :  (colon) character
!
```

General Environment Variables

The Digital GKS environment variables can be divided into two groups:

- General Digital GKS environment variables
- Graphics-handler environment variables

Table 1–1 lists the general environment variables available with Digital GKS. For information on the graphics-handler environment variables, see the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.

Table 1–1 General Environment Variables for Digital GKS

Variable	Value	Description
GKSasf	INDIVIDUAL or BUNDLED	Specifies the default aspect source flag (ASF) setting to be either BUNDLED or INDIVIDUAL. The predefined value is INDIVIDUAL.
GKSattribute_control	ENVOP or APPL	Specifies how the deferral mode, implicit regeneration mode (IRG), and line cap/join attribute values are used. If GKSattribute_control is set to ENVOP: • The deferral mode and implicit regeneration mode (IRG) values defined by the GKSdef_mode and GKSirg parameters will override the values passed to the SET DEFERRAL STATE function. • The line cap/join attribute values defined by the GKSline_cap and GKSline_join parameters will override the values passed to the SET LINE CAP/JOIN escapes. If GKSattribute_control is set to APPL, the reverse is true.
GKScomplex_fill_method	Nonzero value	Specifies that the MapKernel fill area algorithm is enabled. This algorithm is intended for applications which create extremely complex fill areas with holes, "bridges," and self-intersections.
GKSconid	String containing any valid workstation connection identifier	Specifies the default workstation connection identifier to be used in the call to OPEN WORKSTATION, if the caller passes conid 0. The predefined setting is platform-dependent.
GKSdef_mode	ASAP, BNIG, BNIL or ASTI	Specifies the default deferral mode to be one of the following: • ASAP—as soon as possible • BNIG—before the next interaction globally • BNIL—before the next interaction locally • ASTI—at some time The predefined value is defined in the workstation description table.
GKSerrfile	String containing any valid error file specification	Specifies the default error file to be used in the OPEN GKS function, if you pass a NIL (0) error file descriptor. On Digital UNIX systems, this is set to stderr by default.
GKSerror	ON or OFF	Specifies whether the default standard error checking is ON or OFF. The predefined value is ON. Note that if you turn error checking OFF, you may improve overall Digital GKS throughput, but Digital GKS may terminate in an uncontrolled way.
GKShatch_simulation	Nonzero value	Specifies that fill area hatching is to be simulated.

(continued on next page)

Table 1–1 (Cont.) General Environment Variables for Digital GKS

Variable	Value	Description
GKSirg	SUPPRESSED or ALLOWED	Specifies the default implicit regeneration mode (IRG) to be set to either SUPPRESSED or ALLOWED. The predefined value is defined in the workstation description table.
GKSline_cap	BUTT, ROUND, SQUARE, DEFAULT	Specifies the appearance of endpoints for a thick polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). • BUTT—a line is drawn through the endpoint • ROUND—a semicircle is drawn around the endpoint • SQUARE—a square is drawn around the endpoint • DEFAULT—the BUTT option is used
GKSline_join	MITRE, ROUND, BEVEL, DEFAULT	Specifies the appearance of vertices between the endpoints of a thick, multisegment polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). The options are: • MITRE—the outside boundaries of the line (at this vertex point) are extended until they meet • ROUND—a semicircle is drawn around the vertex • BEVEL—a chamfer is drawn through the vertex • DEFAULT—the MITRE option is used
GKSmax_gdp_points	Positive number	Specifies the maximum number of points to be computed for a simulated GDP.
GKSmetafile_type	GKSM or GKS3	Specifies the dimension of the metafile output. The value GKSM is for two-dimensional metafiles; GKS3 is for three-dimensional metafiles. The default value is GKS3.
GKSmetalog	String containing any valid file specification	Specifies that metafile logging is enabled and that the log information should be output to the specified file.
GKSndc_clip	ON or OFF	Specifies the default NDC clipping to ON or OFF. The predefined value is ON.

(continued on next page)

Table 1–1 (Cont.) General Environment Variables for Digital GKS

Variable	Value	Description
GKSnist_certified	Nonzero value	Specifies that Digital GKS will operate strictly according to the GKS standard as interpreted by NIST as follows: • Workstation types 231 (Motif output/input) and 61 (black and white PostScript) only are valid. • Error codes returned by some functions in certain conditions differ from the codes returned in a non-NIST environment. • The Motif workstation is mapped when you call the OPEN WORKSTATION function. • The number of marker types is limited to 18. • Separate viewport data for current and requested viewports is maintained for inquiries. • Implicit regenerations (IRGs) are not optimized (grouped together for better performance). • Output primitives are not deferred if an IRG is pending.
GKSstroke_font1	String containing the file path for stroke font1	Specifies the font file to be used for stroke font 1.
GKSstroke_font_neg*n*	String containing the file path for stroke font *-n*	Specifies the font file to be used for the indicated implementation-defined font *-n*, where *n* is a value from 1 to 24. If you are using the Japanese version, you can specify that font -10001 be used by defining this environment variable as follows: `GKS$STROKE_FONT_NEG`*`10001`*
GKSwstype	String containing any valid workstation type number	Specifies the default workstation type to be used in the call to OPEN WORKSTATION, if the caller passes wstype 0. The predefined setting is platform-dependent.

1.5 Error Handling

The following sections provide information on error codes and error files.

1.5.1 Error Codes

Each Digital GKS function call returns a Digital GKS error value to the calling routine. The returned error values are defined in the definition file `/usr/include/GKS/gksmsgs.*`, where `*` is the extension of the respective programming language.

The function call can include a check of the returned status. The value GKS$_SUCCESS is returned if the function has executed successfully. If this value is not returned, the function has failed. See Appendix A of the related binding manual for information about the Digital GKS errors.

1.5.2 Error Files

Error messages are normally written to an error logging file. The application can specify the error logging file by passing the file to the OPEN GKS function in any of the following ways:

- *By specifying the file name explicitly.*
- *By passing an environment variable (by descriptor) that specifies the error file.*
- *By passing the value 0 or a null string.*

  This causes Digital GKS to open the default error file. To do this, it first attempts to translate the environment variable GKSerrfile. If no translation exists, Digital GKS translates stderr to find the error logging file. You can do either of the following:

  - Define the environment variable GKSerrfile to be any valid file specification.
  - Use the user defaults file `~/.GKSdefaults` or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

If no messages have been written to the error file, a call to CLOSE GKS deletes the file.

1.6 Configuration Files

A configuration file contains the list of workstations to be linked with a Digital GKS application. On Digital UNIX systems, configuration files are only useful when an application is to be linked with the archive libraries (`libGKS.a`, and so forth).

If you use the default configuration file `gksconfig.c` included in the Digital GKS libraries, all device handlers supplied by Digital will be linked to the program. This approach allows you to use any of the Digital-supplied device handlers without relinking your program, but usually results in longer link times and larger executable images than necessary.

To reduce link time and image size, you can customize the configuration file at the system or user level, as described in the following sections. In both cases, you customize the configuration file by changing `INCLUDE` macro invocations to `EXCLUDE`, or vice versa, for each device handler specified in the file. For a list of the workstation handlers and more information on the `INCLUDE` and `EXCLUDE` macros, see the `gksconfig.c` configuration file.

1.6.1 Customizing the Configuration File at System Level

To customize the file at the system level, perform the following steps:

1. Edit the configuration file to exclude those handlers you do not wish to have included automatically in any program.
2. Compile the file to create the new configuration module.
3. Use the command `ar(1)` to replace the file in the library `/usr/lib/libGKS.a`.

When you replace the configuration module, other users must create their own copies of the configuration file (and link to it) to include handlers not contained in the system version of the file.

1.6.2 Customizing the Configuration File at User Level

To customize the configuration file at the user level, perform the following steps:

1. Make a private copy of the configuration file.
2. Edit the file to include only the desired handlers.
3. Compile the private copy of the file to create the new configuration module.
4. Link this private module before linking to the Digital GKS libraries.

2
OpenVMS Programming Considerations

The specific method for using Digital GKS software depends on the features and conventions of each programming language. This chapter describes general issues that must be considered when using any programming language with Digital GKS on an OpenVMS system.

The information contained in this chapter was correct when the manual was printed; however, the information may have been changed. For the most up-to-date information on using Digital GKS on OpenVMS systems, refer to the following files:

SYS$HELP:DECGKS_GBIND_OP_SPEC.PS
SYS$HELP:DECGKS_GBIND_OP_SPEC.TXT

2.1 Including Definition Files

You use Digital GKS software primarily by placing calls to Digital GKS functions in your program. You also need statements in your program other than the calls to Digital GKS functions. These statements make Digital GKS constants and their values accessible to all programs that call Digital GKS functions, regardless of the programming language you use.

Your installation kit includes several definition files that contain Digital GKS constants, and error message definition files that contain Digital GKS completion status code constants. All high-level programming languages that use Digital GKS enable you to insert these external, definition files into the program source code stream at compile time, thus making the constants available.

In the case of the C language, an `#include` statement is provided for inserting a definition file into a program. For example, any C program that uses the Digital GKS GKS$ binding should contain the following line:

```
#include <gksdefs.h>
```

The language definition files and error message files located in SYS$LIBRARY are:

Language	Language Definition Files	Error Message Definition File
ADA	gksdefs.ada	gksmsgs.ada
BASIC	gksdefs.bas	gksmsgs.bas
C	gksdefs.h gksdescrip.h	gksmsgs.h
COBOL	gksdefs.lib	gksmsgs.lib
FORTRAN	gksdefs.for	gksmsgs.for

Language	Language Definition Files	Error Message Definition File
MACRO	gksdefs.r32	gksmsgs.r32
Pascal	gksdefs.pas gksdefs.pen	gksmsgs.pas gksmsgs.pen
PL/1	gksdefs.pli gksdefs.pl2	gksmsgs.pli

Each definition file includes comments describing its use.

2.2 Compiling, Linking, and Executing Your Programs

A program that uses Digital GKS function calls is compiled, linked, and executed like any other program. Digital GKS functions are supplied as an installed shareable image library. The advantages of using a shareable image library are:

- Linking is fast and easy.
- You can upgrade applications with new versions of Digital GKS without having to rebuild the applications.

To compile a program, you should use the compile command appropriate for the programming language you are using. To link the object file and execute the program image, you use the LINK and RUN commands, as described in the following sections.

2.2.1 Linking Your Program

A Digital GKS program compiled with **VAX C** can be linked with the following DCL command:

```
$ LINK program RETURN
```

If the program is written in VAX C and you make calls to C library functions, you must link the program with the VAXCRTL library by using either of the following commands:

```
$ LINK program, SYS$LIBRARY:VAXCRTL/LIB RETURN

$LINK program, SYS$INPUT:/OPTIONS RETURN
SYS$LIBRARY:VAXCRTL/SHARE CTRL/Z RETURN
```

A Digital GKS program compiled with **DEC C** can be linked with the following DCL command:

```
$ LINK program RETURN
```

On *OpenVMS Alpha* systems, you can use IEEE floating point format by linking with IEEE libraries, as follows:

```
$ LINK program, SYS$LIBRARY:GKS$BND/OPT RETURN
```

2.2.2 Executing Your Program

Once you have compiled and linked your program, you can execute it by typing the following command:

```
$ RUN program RETURN
```

2.3 Opening a Workstation

The following sections contain information on opening a workstation, which comprises two steps:

1. Specifying the workstation connection identifier
2. Specifying the workstation type

2.3.1 Specifying the Workstation Connection Identifier

An application can specify the connection to a device by passing the connection identifier (ID) to the OPEN WORKSTATION function in any of the following ways:

- *By passing a logical name specifying the connection ID.*
- *By passing a file or device name.*

  Pass the connection ID as a string, by descriptor.
- *By passing the value 0 or a null string.*

  Digital GKS attempts to translate the logical name GKS$CONID. If no translation exists, GKS uses GKS_DEFAULT.OUTPUT, which specifies a file in the current directory as the connected device. You can define GKS$CONID to be a file name, a device name, or an X11 display name, depending on the workstation type.

2.3.2 Specifying the Workstation Type

The application can specify the workstation type to the OPEN WORKSTATION function in either of the following ways:

- *Use the Digital GKS workstation types.*

  Pass any of the workstation types defined in the appropriate language file `SYS$LIBRARY:GKSDEFS.*`, where `*` is the extension of the respective programming language.
- *Use the default workstation type by passing a value of 0.*

  Digital GKS attempts to translate the OpenVMS logical name GKS$WSTYPE. If no translation exists, Digital GKS uses the workstation type 61 (PostScript®). You can define the logical name GKS$WSTYPE to be any supported workstation type.

2.4 Digital GKS Logical Names

In Digital GKS, a number of logical names are interpreted at run-time. These logical names allow the application to tailor Digital GKS to the needs of the application itself or a specific device. Each logical name controls some aspect of the Digital GKS run-time environment.

Defining Logical Names

You should set all logical names before starting a Digital GKS session. The logical name values remain constant throughout the session; modifying the value of a logical name during a session may or may not have an effect on Digital GKS operation.

On OpenVMS systems, logical names and values can be either uppercase or lowercase, and are defined at the DCL level. The syntax for defining logical names is:

```
$ define GKS$logical value
```

How Digital Searches for Logical Names

When Digital GKS processes logical names, it searches for the logical name definitions in three locations. Once the logical names are found, the search stops. The locations are searched in the following order:

1. The PROCESS logical table.
2. The GROUP logical table.
3. The SYSTEM logical table. This is the default location for defining logical names.

Types of Logical Names

The Digital GKS logical names can be divided into two groups:

- General Digital GKS logical names
- Graphics-handler logical names

Table 2–1 describes the general logical names available with Digital GKS. For information on the graphics-handler logical names, see the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.

Table 2–1 General Logical Names for Digital GKS

Logical Name	Value	Description
GKS$ASF	INDIVIDUAL or BUNDLED	Specifies the default aspect source flag (ASF) setting to be either BUNDLED or INDIVIDUAL. The predefined value is INDIVIDUAL.
GKS$ATTRIBUTE_ CONTROL	ENVOP or APPL	Specifies how the deferral mode, implicit regeneration mode (IRG), and line cap/join attribute values are used. If GKS$ATTRIBUTE_CONTROL is set to ENVOP: • The deferral mode and implicit regeneration mode (IRG) values defined by the GKS$DEF_ MODE and GKS$IRG parameters will override the values passed to the SET DEFERRAL STATE function. • The line cap/join attribute values defined by the GKS$LINE_CAP and GKS$LINE_JOIN parameters will override the values passed to the SET LINE CAP/JOIN escapes. If GKS$ATTRIBUTE_CONTROL is set to APPL, the reverse is true.
GKS$COMPLEX_FILL_ METHOD	Any nonzero value	Specifies that the MapKernel fill area algorithm is enabled. This algorithm is intended for applications which create extremely complex fill areas with holes, "bridges," and self-intersections.

(continued on next page)

Table 2–1 (Cont.) General Logical Names for Digital GKS

Logical Name	Value	Description
GKS$CONID	String containing any valid workstation connection identifier	Specifies the default workstation connection identifier to be used in the call to OPEN WORKSTATION, if the caller passes CONID 0. The predefined setting is platform-dependent.
GKS$DEF_MODE	ASAP, BNIG, BNIL or ASTI	Specifies the default deferral mode to be one of the following: • ASAP—as soon as possible • BNIG—before the next interaction globally • BNIL—before the next interaction locally • ASTI—at some time The predefined value is defined in the workstation description table.
GKS$ERRFILE	String containing any valid error file specification	Specifies the default error file to be used in the OPEN GKS function, if you pass a NIL (0) error file descriptor. On OpenVMS systems, this is set to SYS$ERROR: by default.
GKS$ERROR	ON or OFF	Specifies whether the default standard error checking is ON or OFF. The default value is ON. If you turn error checking OFF, you may improve overall Digital GKS throughput, but Digital GKS may terminate in an uncontrolled way.
GKS$HATCH_ SIMULATION	Nonzero value	Specifies that fill area hatching is to be simulated.
GKS$IRG	SUPPRESSED or ALLOWED	Specifies the default implicit regeneration mode (IRG) to be set to either SUPPRESSED or ALLOWED. The predefined value is defined in the workstation description table.
GKS$LINE_CAP	BUTT, ROUND, SQUARE, DEFAULT	Specifies the appearance of endpoints for a thick polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). The options are: • BUTT—a line is drawn through the endpoint • ROUND—a semicircle is drawn around the endpoint • SQUARE—a square is drawn around the endpoint • DEFAULT—the BUTT option is used

(continued on next page)

Table 2–1 (Cont.) General Logical Names for Digital GKS

Logical Name	Value	Description
GKS$LINE_JOIN	MITRE, ROUND, BEVEL, DEFAULT	Specifies the appearance of vertices between the endpoints of a thick, multisegment polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). The options are: • MITRE—the outside boundaries of the line (at this vertex point) are extended until they meet • ROUND—a semicircle is drawn around the vertex • BEVEL—a chamfer is drawn through the vertex • DEFAULT—the MITRE option is used
GKS$LIST_TYPES	List of workstation types, separated by commas	Specifies the list of workstation types supported by Digital GKS.
GKS$MAX_GDP_POINTS	Positive number	Specifies the maximum number of points to be computed for a simulated GDP.
GKS$METAFILE_TYPE	GKSM or GKS3	Specifies the dimension of the metafile output. The value GKSM is for two-dimensional metafile output. GKS3, the default value, is for three-dimensional metafile output.
GKS$METALOG	String containing any valid file specification	Specifies that metafile logging is enabled and that the log information should be output to the specified file.
GKS$NDC_CLIP	ON or OFF	Specifies the default normalized device coordinate (NDC) clipping to ON or OFF. The predefined value is ON.
GKS$NIST_CERTIFIED	Nonzero value	Specifies that Digital GKS will operate strictly according to the GKS standard as interpreted by NIST as follows: • Workstation types 231 (Motif output/input) and 61 (black and white PostScript) only are valid. • Error codes returned by some functions in certain conditions differ from the codes returned in a non-NIST environment. • The Motif workstation is mapped when you call the OPEN WORKSTATION function. • The number of marker types is limited to 18. • Separate viewport data for current and requested viewports is maintained for inquiries. • Implicit regenerations are not optimized (grouped together for better performance). • Output primitives are not deferred if an implicit regeneration is pending.

(continued on next page)

Table 2–1 (Cont.) General Logical Names for Digital GKS

Logical Name	Value	Description
GKS$STROKE_FONT1	String containing the file path for stroke font1	Specifies the font file to be used for stroke font 1.
GKS$STROKE_FONT_NEG*n*	String containing the file path for stroke font *-n*	Specifies the font file to be used for the indicated implementation-defined font *-n*, where *n* is a value from 1 to 24. If you are using the Japanese version, you can specify that font -10001 be used by defining this option as follows: `GKS$STROKE_FONT_NEG10001.`
GKS$WSTYPE	String containing any valid workstation type number	Specifies the default workstation type to be used in the call to OPEN WORKSTATION, if the caller passes wstype 0. The predefined setting is platform-dependent.

2.5 Error Handling

The following sections contain information on error codes and error files.

2.5.1 Error Codes

Each Digital GKS function call returns a Digital GKS error value to the calling routine. The returned error values are defined in the definition file `gksmsgs.*`, where `*` is the extension of the respective programming language.

The function call can include a check of the returned status. The value GKS$_SUCCESS is returned if the function has executed successfully. If this value is not returned, the function has failed. See Appendix A of the related binding manual for information about the Digital GKS errors.

2.5.2 Error Files

Error messages are normally written to an error logging file. The application can specify the error logging file by passing the file name to the OPEN GKS function in either of the following ways:

- *By specifying the file name explicitly.*
- *By passing a logical name (by descriptor) that specifies the error file.*
- *By passing the value 0 or a null string.*

  This causes Digital GKS to open the default error file. To do this, Digital GKS first attempts to translate the logical name GKS$ERRFILE. If no translation exists, Digital GKS uses SYS$ERROR as the file pointer. You can define GKS$ERRFILE to be any valid file specification.

If no messages have been written to the error file, a call to CLOSE GKS deletes the file.

3
ULTRIX Programming Considerations

The specific method for using Digital GKS software depends on the features and conventions of each programming language. This section describes general issues that must be considered when using the ULTRIX™ C (cc) compiler with Digital GKS on RISC processors.

The information contained in this chapter was correct when the manual was printed; however, the information may have been changed. For the most up-to-date information on using Digital GKS on ULTRIX systems, see the following files:

/usr/lib/GKS/doc/decgks_gbind_op_spec.ps
/usr/lib/GKS/doc/decgks_gbind_op_spec.txt

3.1 Including Definition Files

You use Digital GKS software primarily by placing calls to Digital GKS functions in your program. You also need statements in your program other than the calls to Digital GKS functions. These statements make Digital GKS constants and their values accessible to all programs that call Digital GKS functions, regardless of the programming language you use.

Your installation kit includes several definition files that contain Digital GKS constants, and error message definition files that contain Digital GKS completion status code constants. All high-level programming languages that use Digital GKS enable you to insert these external, definition files into the program source code stream at compile time, thus making the constants available.

In the case of the C language, an `include` statement is provided for inserting a definition file into a program. For example, any C program that uses the Digital GKS GKS$ binding should contain the following line:

```
# include <GKS/gksdefs.h>
```

The language definition files and error message definition files are:

Language	Language Definition Files	Error Message Definition File
ADA	/usr/include/GKS/gksdefs.ada	/usr/include/GKS/gksmsgs.ada
C	/usr/include/GKS/gksdefs.h /usr/include/GKS/gksdescrip.h	/usr/include/GKS/gksmsgs.h
Fortran	/usr/include/GKS/gksdefs.for	/usr/include/GKS/gksmsgs.for

Each definition file includes comments describing its use.

3.2 Compiling, Linking, and Executing Your Programs

A program that uses Digital GKS function calls is compiled, linked, and executed like any other program. To compile and link a program, you should use the compile command appropriate for the processor you are using. To execute the program image, you type the executable program name. The following sections describe how to compile, link, and execute programs on ULTRIX systems on RISC processors.

Note

Only the `ld` linker is supported by Version 5.0 or higher of Digital GKS for ULTRIX on RISC processors.

3.2.1 Compiling and Linking Your Program

The command you use to compile and link a Digital GKS C program depends on which device handler you are using:

- **any device handler except the DECwindows XUI device handler**

  The command is:

  ```
  # cc -I/usr/include/GKS -o program program.c \ RETURN
  [gksconfig.c] -lGKSbnd -lGKS -lMrm -lXm -lXt \ RETURN
  -lddif -li -lcursesX -lc -lX11 -llmf -lm RETURN
  ```

- **the DECwindows XUI device handler**

  The command is:

  ```
  # cc -I/usr/include/GKS -o program program.c \ RETURN
  gks_decw_config.c -lGKSbnd -lGKS -lddif -li -ldwt -lcursesX \ RETURN
  -lc -lX11 -llmf -lm RETURN
  ```

A workstation or device handler can be deliberately excluded from the executable image to decrease image size and link time. This can be done by customizing the configuration file and specifying the customized version in the `link` command. The installed version of the configuration file, `/usr/lib/GKS/gksconfig.c`, is optional.

There is also an installed version of the configuration file that must be used when linking with the DECwindows XUI device handler. This file is called `/usr/lib/GKS/gks_decw_config.c`. See Section 3.6 for more information on how to use configuration files.

The options to the `link` command are required if certain device handlers are included in the configuration file, as follows:

- The libraries `[-lc]` and `[-lX11]` are required for the DECwindows XUI and Motif® device handlers.
- The library `[-ldwt]` is required for the DECwindows XUI device handler.
- The library `[-lddif]` is required for the DDIF™ device handler.
- The library `[-lcursesX]` is required for any device handler for workstations capable of input, output, or both.
- The libraries `[-lMrm -lXm -lXt]` are required for the Motif device handler.

3.2.2 Executing Your Program

Once you have compiled and linked your program, you can execute it by typing the program file name at the command line:

```
$ program RETURN
```

3.3 Opening a Workstation

The following sections contain information on opening a workstation, which comprises two steps:

1. Specifying the workstation connection identifier
2. Specifying the workstation type

3.3.1 Specifying the Connection Identifier

An application can specify the connection by passing the connection identifier (ID) to the OPEN WORKSTATION function in any of the following ways:

- *By passing an environment variable specifying the connection ID.*
- *By file or device name.*

  Pass the connection ID as a string, by descriptor.

- *By passing the value 0 or a null string.*

  Digital GKS attempts to translate the environment variable GKSconid. If no translation exists, Digital GKS uses GKS_default.output, which specifies a file in the current directory as the connected device. You can specify an environment option in one of two ways:

  – Define the environment variable GKSconid to be a file name or device name, depending on the workstation type.

  – Use the user defaults file `~/.GKSdefaults` or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

3.3.2 Specifying the Workstation Type

An application can specify the workstation type to the OPEN WORKSTATION function in either of the following ways:

- *Use the Digital GKS workstation types.*

  Pass any of the workstation types defined in the language file `/usr/include/GKS/gksdefs.*`.

- *Use the default workstation type by passing a value of 0.*

  Digital GKS attempts to translate the environment variable GKSwstype. If no translation exists, Digital GKS uses the workstation type 35 (LA75 printer). You can do either of the following:

  – Define the environment variable GKSwstype to be any supported workstation type.

  – Use the user defaults file ~/.GKSdefaults, or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

3.4 Digital GKS Environment Variables

In Digital GKS, a number of environment variables are interpreted at run-time. These environment variables allow the application to tailor Digital GKS to the needs of the application itself or a specific device. Each environment variable controls some aspect of the Digital GKS run-time environment.

Defining Environment Variables

You must set all environment variables before starting a Digital GKS session. The variable values remain constant throughout the session; modifying the value of a environment variable during a session may or may not have an effect on Digital GKS operation.

On ULTRIX systems, environment variables are defined in a file named `.GKSdefaults` in your login directory, or in the system file `/usr/lib/GKS/.GKSdefaults`. The environment variable names are case sensitive; however, their associated values can be either uppercase or lowercase.

The syntax for defining environment variables is:

```
GKSconid          :  gks_default.output    # connection id, device, or file name
GKSwstype         :  35                    # workstation type (LA 75)
```

The environment variables can also be defined at the shell level. To define an environment variable under the *C shell* (*csh*), use the following syntax:

```
# setenv GKSenvironment_variable value
```

To define an environment variable under the *Bourne shell* (*sh*), use the following syntax:

```
# GKSenvironment_variable=value
```

How Digital Searches for Environment Variables

When Digital GKS processes environment variables, it searches for the environment variable definitions in three locations. Once the environment variables are found, the search stops. The locations are searched in the following order:

1. User-specific ULTRIX environment variables.
2. User-specific environment variables defined in the file `~/.GKSdefaults`. Digital recommends that you define the environment variables in this file.
3. System-wide environment variables defined in the file `/usr/lib/GKS/.GKSdefaults`.

The Default Environment Variable File

The default environment variable file `.GKSdefaults` is as follows:

```
!
! Digital GKS User-specific Environment Variables
! ==============================================
!
GKSconid          :  gks_default.output    # connection id, device, or file name
GKSwstype         :  61                    # workstation type (PostScript)
GKSerror          :  on                    # on or off
GKSasf            :  individual            # bundled or individual
GKSndc_clip       :  on                    # on or off
GKSerrfile        :  stderr                # device or file name
GKSmetafile_type  :  gks3
GKSstroke_font1   :  /usr/lib/GFX/font/gfx_font_neg1 # Stroke font 1
!
! file : /usr/lib/GKS/.GKSdefaults
!
!
! Digital GKS System-Wide Environment Variables
! ============================================
!
! Environment variables allow you to customize the Digital GKS environment
! to suit your needs.
!
!   i) Modify the Digital GKS system wide default settings.
!
!      Edit this file to change Digital GKS system-wide environment variable
!      default settings.
!
!  ii) Modify the Digital GKS user-specific environment variable default settings
!      using the ~/.GKSdefaults file.
!
!      Copy this file into your login account (for example, ~/.GKSdefaults) and
!      modify it to suit your needs.
!
! iii) Modify the Digital GKS user-specific settings using Digital UNIX
!      environment variables.
!
!      For example : setenv GKSwstype 231
!
!
!
! The Digital GKS search order for environment variable translation is:
! --------------------------------------------------------------------
!
! 1) User-specific Digital UNIX environment variables
!
! 2) User-specific environment variables defined in the file ~/.GKSdefaults
!
! 3) System-wide environment variables defined in the file
!    /usr/lib/GKS/.GKSdefaults
!
!
! The allowed syntax in this file is:
! ----------------------------------
!
!       Comments start with  ! or  #  character
!       Space, tabs, and  "  characters are ignored
!       Associations are done by the  =  (equal) or  :  (colon) character
!
```

General Environment Variables

The Digital GKS environment variables can be divided into two groups:

- General Digital GKS environment variables
- Graphics-handler environment variables

Table 3–1 lists the general environment variables available with Digital GKS. For information on the graphics-handler environment variables, see the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*.

Table 3–1 General Environment Variables for Digital GKS

Variable	Value	Description
GKSasf	INDIVIDUAL or BUNDLED	Specifies the default aspect source flag (ASF) setting to be either BUNDLED or INDIVIDUAL. The predefined value is INDIVIDUAL.
GKSattribute_control	ENVOP or APPL	Specifies the deferral mode, implicit regeneration mode (IRG), and line cap/join attribute values used. If GKSattribute_control is set to ENVOP, the deferral mode and implicit regeneration mode (IRG) values defined by the GKSdef_mode and GKSirg parameters will override the values passed to the SET DEFERRAL STATE function, and the line cap /join attribute values defined by GKSline_cap and GKSline_join will override the values passed to the SET LINE CAP/JOIN escapes. If GKSattribute_ control is set to APPL, the reverse is true.
GKScomplex_fill_method	Nonzero value	Specifies that the MapKernel fill area algorithm is enabled. This algorithm is intended for applications which create extremely complex fill areas with holes, "bridges," and self-intersections.
GKSconid	String containing any valid workstation connection identifier	Specifies the default workstation connection identifier to be used in the call to OPEN WORKSTATION, if the caller passes conid 0. The predefined setting is platform-dependent.
GKSdef_mode	ASAP, BNIG, BNIL or ASTI	Specifies the default deferral mode to be ASAP (as soon as possible), BNIG (before the next interaction globally), BNIL (before the next interaction locally), or ASTI (at some time). The predefined value is defined in the workstation description table.
GKSerrfile	String containing any valid error file specification	Specifies the default error file to be used in the OPEN GKS function, if you pass a NIL (0) error file descriptor. On ULTRIX systems, this is set to stderr by default.
GKSerror	ON or OFF	Specifies whether the default standard error checking is ON or OFF. The predefined value is ON. Note that if you turn error checking OFF, you may improve overall Digital GKS throughput, but Digital GKS may terminate in an uncontrolled way.
GKShatch_simulation	Nonzero value	Specifies that fill area hatching is to be simulated.
GKSirg	SUPPRESSED or ALLOWED	Specifies the default implicit regeneration mode (IRG) to be set to either SUPPRESSED or ALLOWED. The predefined value is defined in the workstation description table.

(continued on next page)

Table 3–1 (Cont.) General Environment Variables for Digital GKS

Variable	Value	Description
GKSline_cap	BUTT, ROUND, SQUARE, DEFAULT	Specifies the appearance of endpoints for a thick polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). BUTT—a line is drawn through the endpoint ROUND—a semicircle is drawn around the endpoint SQUARE—a square is drawn around the endpoint DEFAULT—the BUTT option is used
GKSline_join	MITRE, ROUND, BEVEL, DEFAULT	Specifies the appearance of vertices between the endpoints of a thick, multisegment polyline (a line that is more than one pixel wide or is wider than the nominal width of the device). The options are: MITRE—the outside boundaries of the line (at this vertex point) are extended until they meet ROUND—a semicircle is drawn around the vertex BEVEL—a chamfer is drawn through the vertex DEFAULT—the MITRE option is used
GKSmax_gdp_points	Positive number	Specifies the maximum number of points to be computed for a simulated GDP.
GKSmetafile_type	GKSM or GKS3	Specifies the dimension of the metafile output. The value GKSM is for two-dimensional metafiles; GKS3 is for three-dimensional metafiles. The default value is GKS3.
GKSmetalog	String containing any valid file specification	Specifies that metafile logging is enabled and that the log information should be output to the specified file.
GKSndc_clip	ON or OFF	Specifies the default NDC clipping to ON or OFF. The predefined value is ON.

(continued on next page)

Table 3–1 (Cont.) General Environment Variables for Digital GKS

Variable	Value	Description
GKSnist_certified	Nonzero value	Specifies that Digital GKS will operate strictly according to the GKS standard as interpreted by NIST as follows: • Workstation types 231 (Motif output/input) and 61 (black and white PostScript) only are valid. • Error codes returned by some functions in certain conditions differ from the codes returned in a non-NIST environment. • The Motif workstation is mapped when you call the OPEN WORKSTATION function. • The number of marker types is limited to 18. • Separate viewport data for current and requested viewports is maintained for inquiries. • Implicit regenerations (IRGs) are not optimized (grouped together for better performance). • Output primitives are not deferred if an IRG is pending.
GKSstroke_font1	String containing the file path for stroke font1	Specifies the default stroke font 1 to be used.
GKSstroke_font_neg*n*	String containing the file path for stroke font *-n*	Specifies the font file to be used for the indicated implementation-defined font *-n*, where *n* is a value from 1 to 24. If you are using the Japanese version, you can specify that the font -10001 be used by defining this environment variable as follows: `GKS$STROKE_FONT_NEG`*`10001`*`.`
GKSwstype	String containing any valid workstation type number	Specifies the default workstation type to be used in the call to OPEN WORKSTATION, if the caller passes wstype 0. The predefined setting is platform-dependent.

3.5 Error Handling

The following sections contain information on error codes and error files.

3.5.1 Error Codes

Each Digital GKS function call returns a Digital GKS error value to the calling routine. The returned error values are defined in the definition file `/usr/include/GKS/gksmsgs.*`, where `*` is the extension of the respective programming language.

The function call can include a check of the returned status. The value GKS_SUCCESS (0) is returned if the function has executed successfully. If this value is not returned, the function has failed. See Appendix A of the related binding manual for information about the Digital GKS errors.

3.5.2 Error Files

Error messages are normally written to an error logging file. The application can specify the error logging file by passing the file name to the OPEN GKS function in any of the following ways:

- *By specifying the file name explicitly.*
- *By passing an environment variable (by descriptor) that specifies the error file.*
- *By passing the value 0 or a null string.*

  This causes Digital GKS to open the default error file. To do this, it first attempts to translate the environment variable GKSerrfile. If no translation exists, Digital GKS translates stderr to find the error logging file. You can do either of the following:

  - Define the environment variable GKSerrfile to be any valid file specification.
  - Use the user defaults file `~/.GKSdefaults` or the system defaults file `/usr/lib/GKS/.GKSdefaults`.

If no messages have been written to the error file, a call to CLOSE GKS deletes the file.

3.6 Configuration Files

A configuration file contains the list of workstations to be linked with a Digital GKS application. There are two configuration files:

- `gksconfig.c`—for all device handlers except the DECwindows XUI handler
- `gks_decw_config.c`—for all device handlers except the Motif handler

You can use either of the two files, but not both. If you use the default configuration file `gksconfig.c` included in the Digital GKS libraries, all device handlers supplied by Digital (except DECwindows XUI) will be linked to the program. If you use the `gks_decw_config.c` configuration file, all device handlers supplied by Digital (except Motif) will be linked to the program. These files allow you to use numerous device handlers without relinking your program, but usually result in longer link times and larger executable images than necessary.

To reduce link time and image size, you can customize these files at the system or user level, as described in the following sections. In both cases, you customize the configuration file by changing the INCLUDE macro to EXCLUDE, or vice versa, for each device handler specified in the file. For a list of the workstation handlers and more information on the INCLUDE and EXCLUDE macros, see the configuration file `gksconfig.c` or `gks_decw_config.c`.

3.6.1 Customizing the Configuration File at System Level

To customize the configuration file at the system level, perform the following steps:

1. Edit the configuration file to exclude those handlers you do not wish to have included automatically in any program.
2. Compile the file to create the new configuration module.
3. Use the command `ar(1)` to replace the file in the library `/usr/lib/libGKS.a`.

When you replace the configuration module, other users must create their own copies of the configuration file (and link to it) to include handlers not contained in the system version of the file.

3.6.2 Customizing the Configuration File at User Level

To customize the configuration file at the user level, perform the following steps:

1. Make a private copy of the configuration file.
2. Edit the file to include only the desired handlers.
3. Compile the private copy of the file to create the new configuration module.
4. Link this private module before linking to the Digital GKS libraries.

Index

C

Compiling
  Digital UNIX programs, 1–2
  OpenVMS programs, 2–2
  ULTRIX programs, 3–2
Configuration files, 1–9 to 1–10, 3–9 to 3–10
  customizing
    system level, 1–9, 3–9
    user level, 1–10, 3–10
Connection identifiers
  specifying on Digital UNIX, 1–3
  specifying on OpenVMS, 2–3
  specifying on ULTRIX, 3–3
Constants
  requirements, 1–1, 2–1, 3–1

D

Definition files
  including, 1–1, 2–1, 3–1
  list of, 2–1
Digital UNIX operating system, 1–1 to 1–10

E

Environment variables, 1–3, 3–4
  default file, 1–4, 3–4
  defining
    in file, 1–4, 3–4
      under *csh*, 1–4, 3–4
      under *sh*, 1–4, 3–4
  GKSasf, 1–6, 3–6
  GKSattribute_control, 1–6, 3–6
  GKScomplex_fill_method, 1–6, 3–6
  GKSconid, 1–3, 1–6, 3–3, 3–6
  .GKSdefaults, 1–4, 3–4
  GKSdef_mode, 1–6, 3–6
  GKSerrfile, 1–6, 1–9, 3–6, 3–9
  GKSerror, 1–6, 3–6
  GKShatch_simulation, 1–6, 3–6
  GKSirg, 1–6, 3–6
  GKSline_cap, 1–7, 3–6
  GKSline_join, 1–7, 3–7
  GKSmax_gdp_points, 1–7, 3–7
  GKSmetafile_type, 1–7, 3–7
  GKSmetalog, 1–7, 3–7
Environment variables (cont'd)
  GKSndc_clip, 1–7, 3–7
  GKSnist_certified, 1–7, 3–7
  GKSstroke_font1, 1–8, 3–8
  GKSstroke_font_neg*n*, 1–8, 3–8
  GKSwstype, 1–8, 3–8
  search order, 1–4, 3–4
  stderr, 1–9, 3–9
  system defaults file, 1–3, 3–3
  types, 1–5, 3–6
    general, 1–5, 3–6
  user defaults file, 1–3, 3–3
Error codes
  defined, 1–8, 2–8, 3–8
  Digital UNIX, 1–8
  ULTRIX, 3–8
  VMS, 2–8
Error files
  default, 2–8
  defined, 1–9, 3–9
  Digital UNIX, 1–9
  OpenVMS, 2–8
  ULTRIX, 3–9
Error handling, 1–8 to 1–9, 2–8, 3–8 to 3–9
Error status files
  list of, 1–1, 2–2, 3–1
Executing
  Digital UNIX programs, 1–2
  OpenVMS programs, 2–2
  ULTRIX programs, 3–2, 3–3

F

Files
  definition
    list of, 2–1
  error status
    list of, 1–1, 2–2, 3–1

G

GKS$ASF, 2–4
GKS$ATTRIBUTE, 2–4
GKS$ binding files
  Digital UNIX, 1–1, 3–1
  OpenVMS, 2–1

GKS$COMPLEX_FILL_METHOD, 2–4
GKS$CONID, 2–4
GKS$DEF_MODE, 2–5
GKS$ERRFILE, 2–5
GKS$ERROR, 2–5
GKS$HATCH_SIMULATION, 2–5
GKS$IRG, 2–5
GKS$LINE_CAP, 2–5
GKS$LINE_JOIN, 2–5
GKS$LIST_TYPES, 2–6
GKS$MAX_GDP_POINTS, 2–6
GKS$METAFILE_TYPE, 2–6
GKS$METALOG, 2–6
GKS$NDC_CLIP, 2–6
GKS$NIST_CERTIFIED, 2–6
GKS$STROKE_FONT1, 2–6
GKS$STROKE_FONT_NEGn, 2–7
GKS$WSTYPE, 2–7
GKSasf, 1–6, 3–6
GKSattribute_control, 1–6, 3–6
GKScomplex_fill_method, 1–6, 3–6
gksconfig.c, 1–9, 3–9
GKSconid, 1–6, 3–6
GKSdef_mode, 1–6, 3–6
GKSerrfile, 1–6, 3–6
GKSerror, 1–6, 3–6
GKShatch_simulation, 1–6, 3–6
GKSirg, 1–6, 3–6
GKSline_cap, 1–7, 3–6
GKSline_join, 1–7, 3–7
GKSmax_gdp_points, 1–7, 3–7
GKSmetafile_type, 1–7, 3–7
GKSmetalog, 1–7, 3–7
GKSndc_clip, 1–7, 3–7
GKSnist_certified, 1–7, 3–7
GKSstroke_font1, 1–8, 3–8
GKSstroke_font_neg*n*, 1–8, 3–8
GKSwstype, 1–8, 3–8
gks_decw_config.c, 3–9

I

Include
  definition files, 1–1, 2–1, 3–1
INCLUDE statement
  all languages, 1–1, 2–1, 3–1

L

Linking
  Digital UNIX programs, 1–2
  OpenVMS programs, 2–2
  reducing time, 1–9, 3–9
  ULTRIX programs, 3–2
Logical names, 2–3
  default, 2–3
  defining
    at DCL level, 2–3
Logical names (cont'd)
  GKS$ASF, 2–4
  GKS$ATTRIBUTE_CONTROL, 2–4
  GKS$COMPLEX_FILL_METHOD, 2–4
  GKS$CONID, 2–3, 2–4
  GKS$DEF_MODE, 2–5
  GKS$ERRFILE, 2–5, 2–8
  GKS$ERROR, 2–5
  GKS$HATCH_SIMULATION, 2–5
  GKS$IRG, 2–5
  GKS$LINE_CAP, 2–5
  GKS$LINE_JOIN, 2–5
  GKS$LIST_TYPES, 2–6
  GKS$MAX_GDP_POINTS, 2–6
  GKS$METAFILE_TYPE, 2–6
  GKS$METALOG, 2–6
  GKS$NDC_CLIP, 2–6
  GKS$NIST_CERTIFIED, 2–6
  GKS$STROKE_FONT1, 2–6
  GKS$STROKE_FONT_NEGn, 2–7
  GKS$WSTYPE, 2–3, 2–7
  search order, 2–4
  types, 2–4

O

Opening a workstation, 1–3, 2–3, 3–3
OpenVMS logical names
  GKS$CONID, 2–3
  GKS$ERRFILE, 2–8
  GKS$WSTYPE, 2–3
Operating system
  Digital UNIX, 1–1
  ULTRIX, 3–1

P

Programs
  execution of, 1–2, 2–2, 3–2, 3–3

R

RUN DCL command, 2–2

S

Statements
  include, 1–1, 2–1, 3–1
System defaults file, 1–9, 3–9

U

ULTRIX linking
  RISC, 3–2
ULTRIX operating system, 3–1 to 3–10
User defaults file, 1–9, 3–9

W

Workstation type
default, 1–3, 2–3, 3–3
defined, 1–3, 2–3, 3–3
specifying on Digital UNIX, 1–3
specifying on OpenVMS, 2–3
specifying on ULTRIX, 3–3