Copyright Notice

GKS/GKS-3D at CERN

CERN Program Library documentation

© Copyright CERN, Geneva 1992

Copyright and any other appropriate legal protection of these computer programs and associated documentation reserved in all countries of the world.

These programs or documentation may not be reproduced by any method without prior written consent of the Director-General of CERN or his delegate.

Permission for the usage of any programs described herein is granted apriori to those scientific institutes associated with the CERN experimental program or with whom CERN has concluded a scientific collaboration agreement.

CERN welcomes comments concerning this program but undertakes no obligation for its maintenance, nor responsibility for its correctness, and accepts no liability whatsoever resulting from the use of this program.

Requests for information should be addressed to:

CERN Program Library Office
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel.  +41 22 767 4951
Fax.  +41 22 767 7155
Bitnet: CERNLIB@CERNVM
DECnet: VXCERN::CERNLIB (node 22.190)
Internet: CERNLIB@CERNVM.CERN.CH

Trademark notice: All trademarks appearing in this guide are acknowledged as such.

Contact Person: Carlo Vandoni/CN (VANDONI@CERNVM.CERN.CH)
Technical Realization: Michel Goossens/CN (GOOSENS@CERNVM.CERN.CH)

Second edition - July 1992
Preliminary remarks

This manual explains how to use GKS and GKS-3D at CERN.
In this manual examples are in monotype face and strings to be input by the user are underlined. In the index the page where a routine is defined is in bold, page numbers where a routine is referenced are in normal type.
In the description of the routines a * following the name of a parameter indicates that this is an output parameter. If another * precedes a parameter in the calling sequence, the parameter in question is both an input and output parameter.
Certain areas described in this manual are still subject to change. Their description is included in the hope of soliciting feedback. They are flagged with a changebar in the margin.
This document has been produced using \LaTeX\ [1] with the cernman style option, developed at CERN. A PostScript file hbook.ps, containing a complete printable version of this manual, can be obtained from any CERN machine by anonymous ftp as follows (commands to be typed by the user are underlined):

```
ftp asis01.cern.ch
Trying 128.141.8.104...
Connected to asis01.cern.ch.
220 asis01 FTP server (SunOS 4.1) ready.
Name (asis01:username): anonymous
Password: your_mailaddress
ftp> cd doc/cernlib
ftp> get gks.ps
ftp> quit
```
# Table of Contents

## 1 Introduction

1.1 What is GKS (the Graphical Kernel System) ........................................... 1
1.2 Major Differences to Earlier Graphics Interfaces ............................... 2
1.3 Computer Graphics Metafiles (GKSM and CGM) .................................. 3
1.4 The Computer Graphics Virtual Device Interface (CGI) ..................... 3
1.5 Overview of Basic Facilities available at CERN ................................ 4
   1.5.1 GKSGRAL ........................................................................ 4
   1.5.2 Include Files ..................................................................... 4
   1.5.3 GKS Metafile Display and Editing ...................................... 4
   1.5.4 Hardcopy and Conversion Facilities .................................... 4
   1.5.5 Mainframe Graphics using PCs ........................................... 5
   1.5.6 Documentation .................................................................. 5
   1.5.7 The User Consultancy Office (UCO) .................................... 5

## 2 GKS Primer

2.1 Workstations .................................................................................... 6
   2.1.1 Definition ........................................................................... 6
   2.1.2 Opening and Closing GKS .................................................. 6
   2.1.3 Opening and Closing a Workstation ...................................... 6
   2.1.4 Activating and Deactivating a Workstation ......................... 7
   2.1.5 Clearing the Workstation ................................................... 7
   2.1.6 Deferral States .................................................................... 7
   2.1.7 Updating the Workstation .................................................. 8
   2.1.8 Sending Messages/Prompts ............................................... 8

2.2 The Drawing Primitives ........................................................................ 9
   2.2.1 Polyline .............................................................................. 9
   2.2.2 Polymarker ......................................................................... 9
   2.2.3 Text .................................................................................... 9
   2.2.4 Fill Area ............................................................................ 10
   2.2.5 Cell Array ......................................................................... 10
   2.2.6 Generalised Drawing Primitive ........................................ 10

2.3 The Output Attributes ...................................................................... 11
   2.3.1 Attribute Bundles .................................................................. 11
   2.3.2 The List of GKS Attributes .................................................. 13
   2.3.3 Specifying Line Styles For Polylines .................................... 14
   2.3.4 Specifying Marker Types for Polymarkers ....................... 14
   2.3.5 Specifying Fill Areas ......................................................... 14
6 GKS-3D Primer

6.1 Introduction to GKS-3D .................................................. 41
6.2 The Drawing Primitives .................................................... 41
6.3 The Output Attributes .................................................... 43
6.4 Viewing in 3D ............................................................... 43
   6.4.1 The Normalization Transformation ............................... 44
   6.4.2 The View Orientation Transformation ............................ 45
   6.4.3 The View Mapping (Projection) Transformation ............... 45
   6.4.4 Setting the 3D View Representation ............................ 46
   6.4.5 Workstation Transformation ....................................... 48
   6.4.6 But I don’t see anything! ........................................... 48
6.5 Segments ................................................................. 49
6.6 Graphical Input .......................................................... 50
6.7 GKS-3D Metafile ........................................................ 50

7 General Hints for GKS Use .................................................. 51

7.1 System and Implementation Dependencies ............................ 51
7.2 Integrating separately written modules of code: .................... 52
   7.2.1 Using GKS-3D libraries for GKS (2D) applications ........... 52
7.3 Plotting numbers as text strings ....................................... 52
7.4 GKSPACK ................................................................. 53
   7.4.1 List of routines .................................................... 55
   7.4.2 GKSPACK Error Messages ......................................... 60
   7.4.3 GKSPACK Example Program ....................................... 61

8 GKSGRAL and GKSGRAL-3D .................................................. 62

8.1 Devices Drivers .......................................................... 62
8.2 Connection Identifiers .................................................... 62
8.3 Implementation-Dependent Features ................................... 63
8.4 System-Dependent Considerations ..................................... 65
   8.4.1 IBM ............................................................. 65
   8.4.2 Use of Segment Storage on IBM .................................. 66
   8.4.3 Debugging on IBM ................................................ 66
   8.4.4 VAX/VMS ........................................................ 66
   8.4.5 VAXstation features .............................................. 67
   8.4.6 UNIX ............................................................. 68
   8.4.7 APOLLO .......................................................... 70
   8.4.8 CRAY under UNICOS ............................................. 73
8.5 DECGKS: Coexistence with GKSGRAL and Implementation ......... 73
# F  GKSGRAL Workstation Description Tables

- F.1   APOLLO .......................................................... 112
- F.2   HP PLOTTERS ..................................................... 114
- F.3   MEGATEK 7xxx/33xx series .................................. 115
  - F.3.1 Control of Viewing Transformations .................. 117
- F.4   PERICOM MONTEREY MG600, GRAPH PAC, and FALCO  118
- F.5   POSTSCRIPT LASER PRINTERS .............................. 120
- F.6   QMS Lasergraphix 800, 1200 ............................... 122
- F.7   TEKTRONIX 4010, 4012, 4014 and Compatible Terminals. 123
- F.8   TEKTRONIX 4100/4200 and PERICOM MX2000/7000/8000 series 125
- F.9   VAXstation GPX ................................................. 127
- F.10  VERSATEC ...................................................... 129
- F.11  VT240 ............................................................. 130
- F.12  VT340 ............................................................. 132
- F.13  XEROX, APA 6670 and 3812 LASER PRINTERS ........... 135

# G  GKSGRAL Software Installation

- G.1   Library Configuration ........................................ 136
- G.2   VAX Installation ............................................... 136
- G.3   IBM Installation ................................................ 137
- G.4   UNIX Installation .............................................. 137
  - G.4.1 Sites with hardware for which object code is not available 137
  - G.4.2 Sites with hardware for which object code is available 141
- G.5   Apollo Installation ........................................... 143

# H  Distribution of GKSGRAL to Affiliated Institutes

- H.1   Summary of Offer for an Embedded Licence ................ 144
- H.2   Pricing ......................................................... 145
- H.3   Procedure ..................................................... 146

# Bibliography

# Index
List of Figures

2.1 Examples of some GKS Primitives .................................................. 9
2.2 GKS line styles ................................................................................. 14
2.3 GKS marker types ............................................................................ 14
2.4 GKS fill area styles ........................................................................... 15
2.5 Text fonts, precisions and up vectors .................................................. 16
2.6 Text alignment ................................................................................... 17
2.7 Windows and Viewports ..................................................................... 19
2.8 Normalization and Workstation Transformations .................................. 20
2.9 Data Flow for GKS ............................................................................. 24
6.1 The GKS-3D Viewing Pipeline ............................................................ 44
6.2 The GKS-3D View Reference System .................................................. 45
6.3 The GKS-3D Projection System .......................................................... 46
8.1 GTS-GRAL italic, proportionally-spaced fonts ...................................... 64
8.2 CERN-defined hatch patterns ............................................................. 64
A.1 Output produced by example "Ducks" .................................................. 78

List of Tables
Chapter 1: Introduction

As a matter of policy, a recommendation was made by HEPCCC to base future HEP computer graphics applications on the ISO standard Graphical Kernel System, GKS [2] and GKS-3D [3]. GKS had been in use already at CERN for 2D applications, but the decision to use GKS also for 3D work led to a new tender operation being carried out in 1986. This resulted in the firm GTS-GRAL, of Darmstadt, being selected to provide new implementations of both GKS and GKS-3D (called GKSGRAL and GKSGRAL-3D). These have been installed on all the major CERN operating systems (VM/CMS, UNICOS, VMS, AEGIS and UNIX) and the contract with GTS-GRAL includes provision for institutes affiliated to CERN to obtain a licence for the use of the software at very favourable rates.

GKSGRAL drivers are available for a large range of graphics terminals. Users should be aware of the implications in testing and maintenance work which has to be carried out on the very large number of operating-system/driver combinations.

1.1 What is GKS (the Graphical Kernel System)

The Graphical Kernel System (GKS) [2] is a document produced by the International Standards Organization (ISO) which defines a common interface to interactive computer graphics for application programs. GKS has been designed by a group of experts representing the national standards institutions of most major industrialized countries. The full standard provides functional specifications for some 200 subroutines which perform graphics input and output in a device independent way. Application programs can thus move freely between different graphics devices and different host computers. For the first time graphics programs have become genuinely portable.

However, one should point out that GKS itself is not portable. Individual GKS implementations will vary substantially as they have to support different graphics devices on different computers. Moreover, GKS is a kernel system, and thus does not include an arbitrary collection of functions to produce histograms or contour plots, etc. Such facilities are regarded as applications which sit on top of the basic graphics package and, at CERN, they are provided by the Graphical Extensions to the NAG Library [4], or the HPLOT package [5].

In order to allow particular applications to choose a graphics package with the appropriate capability, GKS has been defined to have different levels. The level structure has two dimensions, one for output (0, 1, or 2) and one for input (a, b, or c). Higher levels include the capabilities of lower levels. In the United States, ANSI has defined also a level ‘m’, for very simple applications, which sits below output level ’0’. Most implementations provide all output (level ’2’) and intermediate input (level ’b’). The reason input level ’c’ is not usually supported is that it requires asynchronous input facilities not found in all operating systems.

The GKS functions have been defined independently from a specific programming language, and bindings to individual languages are subject to separate standards efforts which have been undertaken for all the major languages. The FORTRAN binding is defined by [6].

The Graphical Kernel System for two dimensional graphics was adopted as an ISO standard in 1985, and since that date work has been in progress to define a three dimensional super-set [3] which was accepted as an International Standard during 1988. The FORTRAN binding to GKS-3D has also been published as a Draft International Standard [7].

The GKS functions are separated into those which pass values to GKS for control, setting or output, and those which inquire about status information. There are 8 distinct classes:
Chapter 1. Introduction

1. Control functions  5. Transformations
2. Output Attributes  6. Input functions
3. Output Primitives  7. Metafile functions
4. Segment functions  8. Inquiry functions

Note

Throughout this document many values are specified using GKS Enumerated Types. The symbolic names for these all begin with 'G' and are enclosed in single quotes within the text. (But do not include the quotes in FORTRAN calls!) They have been coded into an Include File, called ENUM.INC, which is listed in Appendix on Page 100.

1.2 Major Differences to Earlier Graphics Interfaces

In some respects GKS is a major departure from earlier software interfaces to graphics display systems, and several new concepts have been introduced. The following is a brief list of the new features, which will be more fully described within the body of the document.

1 GKS has no concept of a current point. This idea dates from when most graphics devices were pen plotters, in which case the current point corresponded clearly to the position of the pen. However, it would be less obvious where one would define the current point when drawing a filled polygon on a raster display. Thus, all GKS primitives which require a starting point specify that point in the function call.

2 In most cases earlier software interfaces could only control a single device at a time, and if some means was provided to ‘capture’ the graphics output for storage, then this was handled in a special way. GKS simplifies this situation considerably by defining Workstations, which correspond to ‘virtual devices’. GKS maps each workstation onto a real piece of hardware, but to the application programmer they are all handled in the same way, and several workstations can be active simultaneously. There is a workstation which writes the output it receives onto a Metafile for storage and/or transmission over a network and this, also, is handled just like any other workstation.

3 GKS incorporates the concept of a Logical Device to control input. There are six classes of Logical Input Devices, and these are mapped onto the actual physical devices in use, without the application programmer having to write special code if, for example, (s)he is using a mouse instead of a tablet. This helps to make application code much more portable. The following logical devices are of particular interest:

   1 The Logical Locator Device, which returns the locator position in the user’s coordinate system, not that of the physical device.
   2 The Logical Pick Device, which tells the application at which object the user is pointing, without having to provide the code to work it out.
   3 The Logical Choice Device, which allows the user to choose one option out of a set.

4 GKS provides a mechanism to group the graphics output commands into Segments, where they may be stored for later use. Thus, an image stored in a segment could be made invisible and then visible again without the application having to re-draw it. Information sent to a workstation to produce an image which is not stored is called non-retained data.
1.3. **Computer Graphics Metafiles (GKSM and CGM)**

Finally, GKS provides a whole host of functions allowing the application to inquire at run-time the state and/or capabilities of a workstation or the implementation of GKS itself. This is a very important feature for allowing code to be written in a portable way, and still to make the best possible use of the environment in which it is running.

### 1.3 Computer Graphics Metafiles (GKSM and CGM)

A graphics metafile is a computer file containing a set of data records describing a graphical image. It may be used for:

1. Transmitting graphical information between different types of computers.
2. Transmitting graphical information from one site to another. (by magnetic tape for example)
3. Transmitting graphical information from one application to another.
4. Device spooling, e.g. for a plotter.

As yet there is no official ISO standard for writing a GKS Metafile (GKSM). However, the ISO GKS Functional Description assumes the existence of one and, in Appendix E of the document, a metafile format is described and its use is recommended. A GKS metafile created using this format is known as an Appendix E metafile. Unfortunately, not all implementations follow the Appendix E format, and so GKSM metafiles created by different GKS packages may be incompatible. In fact, even different examples of Appendix E metafiles may be incompatible due to variations in the file record structures, etc.

The Computer Graphics Metafile (CGM) has been produced by a separate standards group within ISO and, when implementations become available, it will be an alternative to the Appendix E metafile for the storage and transmission of complete pictures. One should note that because CGM is an independent standard compatible with GKS, it is expected to become adopted as the picture interchange format used between all graphics standards.

### 1.4 The Computer Graphics Virtual Device Interface (CGI)

One of the new features introduced by GKS was the concept of a **Logical Workstation**. This provides an interface layer within the graphics package below which are situated a set of workstation drivers. The workstations simulate in software any features which are not provided by a particular display, and so simplify the problem of driving new devices. To the applications programmer all workstations have (more-or-less) similar capabilities, although their response time will clearly be faster if these capabilities are built into the hardware, rather than emulated. However, GKS defines only the interface as seen by the application, and not the internal workstation interface. This means that it is not possible to move workstation drivers from one implementation of GKS to another.

There are difficulties in defining this interface because, if the level is too high, then most workstation drivers must incorporate a great deal of code. If on the other hand the level is too low, then the software will not be able to make use of advanced features in high-performance display systems. There is thus a trade-off to be made, which is currently decided upon by the designers of each graphics package. The goal of the CGI is to standardize this interface. However, it is proving to be a lengthy business, because there are difficulties to persuade all the parties involved to compromise.

It should be mentioned that one of the goals of CGI is to allow the functionality of the graphics system to be split between multiple CPUs. For example, the features defined by the CGI could either be built
Chapter 1. Introduction

into a graphics terminal, or programmed into a Personal Workstation. This type of functionality has become loosely termed Networked Graphics. In this realm CGI may be overtaken by events, because a networked graphics system called X-Window is fast becoming a ‘de facto’ standard. X-Window comes from MIT, and is an outcome of the Athena Project financed by DEC and IBM.

1.5 Overview of Basic Facilities available at CERN

The following graphics services and facilities are supported at CERN and will be described in more detail within this manual. Note that PLOT10/GKS and MGKS are no longer supported.

1.5.1 GKS/GRAL

Both GKS (2D) and GKS-3D libraries from GTS-GRAL (GKS/GRAL and GKS/GRAL-3D) are available for general use on IBM VM/CMS, VAX VMS, CRAY UNICOS, APOLLO AEGIS, and UNIX. However, in the case of UNIX, compiled libraries are only available from CERN for those machines on which they may be produced.

As of March, 1990, the PROduction GKS/GRAL version is 7.4/3.2, and the GKS/GRAL-3D version is 2.0.

1.5.2 Include Files

To aid program writing two INCLUDE FILES are available:

GTSDEV This file contains a set of parameters defining the GKS Workstation Types available on GKS/GRAL.

ENUM This file contains a set of parameters defining the GKS Enumeration Types. It is highly recommended that these are used in preference to directly coding in collections of integers, the meaning of which is immediately forgotten.

Although the use of include files is not standard FORTRAN-77, this facility is supported by the compilers on all the major CERN systems. To produce standard FORTRAN code the text of the include file must be inserted into the application program.

1.5.3 GKS Metafile Display and Editing

GKS Appendix E metafiles may be interpreted on display screens using the interactive command GRVIEW, which is available on IBM, VAX and APOLLO systems. GRVIEW also allows metafiles to be edited, and combines features from the old commands GKSTV and GKSED, which it replaces.

1.5.4 Hardcopy and Conversion Facilities

GKS Appendix E metafiles may be interpreted onto a range of hardcopy devices using the command GRPLOT, or converted to another format using GRCONV. These two commands have replaced the old utilities GKSVT, GKSCP, GKSX87, GKS3812, and GKS/SGML.

GRCONV runs locally to convert a GKS Appendix E metafile into a PostScript file or a file of Tektronix 4014 escape sequences. The command also may be used to convert a metafile into IBM 3812 format for inclusion in documents produced by SGML. GRPLOT plots a metafile on a device specified by a
1.5. Overview of Basic Facilities available at CERN

parameter. Supported devices include the computer centre monochrome and colour Versatec plotters, the Xerox 4050s, and the IBM 3812s.

Apart from Appendix E metafiles, it is possible to produce PostScript or Tektronix 4014 output files, either directly from GKS, via GRVIEW, or via GRCONV. PostScript files may be printed on any local PostScript printer, although they are less versatile than a metafile and may not be edited or plotted on any other type of hardcopy device. Various laser printers accept files of Tektronix 4014 escape codes as input.

1.5.5 Mainframe Graphics using PCs

After evaluation of several graphics emulators, the TeemTalk package from Pericom has been selected as being suitable to allow IBM compatible PCs to be used as graphics terminals. TeemTalk emulates Tektronix 4107/4207 colour terminals which are supported by GKS on all CERN time-sharing systems, and it is possible for CERN to order licences at very favorable rates. Those people who would like to obtain a licence should contact R. Cailliau or F. Ovett of AS-MI.

1.5.6 Documentation

Apart from this GKS/GKS-3D at CERN, there exists To complete ... the GRVIEW, GRCONV and GRPLOT; Metafile Utility User’s Guide, + ..., all of which are available from the UCO. Copies of the GKSGRAL Reference Manual and the GKSGRAL-3D Reference Manual may be borrowed from the Computer Science Library, purchased from GTS-GRAL, or purchased from CERN under an agreement with GTS-GRAL (see Appendix on Page for details). Note that the GKSGRAL manuals are covered by copyright.

On-Line help files are available using the FIND xxx command on IBM, the HELP xxx command on VMS, and via the command HELP CERN xxx on APOLLO. Keywords (xxx) are: GRAPHICS, GKS, GRPLOT, GRCONV, GRVIEW, and GKSPACK.

1.5.7 The User Consultancy Office (UCO)

General graphics inquiries should be directed to the User Consultancy Office on the ground floor of Building 513 (Tel: 4952, Electronic Mail: UCO@CERNVM.CERN.CH.
Chapter 2: GKS Primer

2.1 Workstations

2.1.1 Definition

GKS defines the concept of an abstract *graphical workstation* which provide the logical interface through which the application program controls the physical devices. A workstation can have one *display surface* and/or a collection of *logical input devices*. Thus, a particular interactive workstation may belong to one of three classes: Output-Only, Input-Only, or Input-Output. (There are a further two classes of workstation: Metafile-Input and Metafile-Output.) GKS allows several workstations to be open simultaneously, for example, one for an interactive graphics terminal and one for a metafile or hardcopy device. The actual maximum number of simultaneously open workstations is installation dependent; in the GTS-GRAL implementation it is set to 15.

GKS allows the appearance of output primitives to vary between workstations in order to take advantage of their differing capabilities. However, it is possible to inquire at run-time what the actual capabilities are, and to design one’s code appropriately. The inquiry functions are too numerous to list in this Primer, and the reader is referred to one of the texts in the bibliography.

2.1.2 Opening and Closing GKS

Before any workstation is opened or activated, GKS itself must be opened. This must be the first call in any GKS program as it performs the initialization. The last call should be to close (exit from) GKS.

CALL GOPKS(ERRFIL, BUF)

CALL GCLKS

ERRFIL defines the FORTRAN logical unit number to which all error messages will be sent. BUF is the amount of memory space to be used. (This parameter is ignored by the GKSGRAL and GKSGRAL-3D implementations.)

2.1.3 Opening and Closing a Workstation

Each workstation is identified in the application program by a unique number, the *workstation identifier*. This number is used in routines which communicate with the device, i.e. when output is to be sent to it, or when it is to be cleared, etc. In order to establish a connection between the application program and a workstation, the workstation has to be opened. When the workstation is closed, the connection is released, segments stored in Workstation Dependent Segment Storage are lost, and no further interactions are possible. The calls to open and close a workstation are:

CALL GOPWK(WKID, CONID, WTYPE)

and

CALL GCLWK(WKID)
2.1. Workstations

The workstation identifier WKID is a positive integer which the application chooses, and is typically a number like 1, 2, 3... The integer CONID is the Connection Identifier which connects the workstation to an I/O channel. It is usually a FORTRAN logical unit number, but depends on the workstation and the host computer being used. Thus, the value of conid must be obtained from the workstation-dependent documentation.

The Workstation Type (WTYPE) is an integer which specifies which type of workstation should be opened, e.g. a Tektronix 4014 or a Versatec plotter. Workstation Types are implementation dependent. A list of workstation identifiers in use at CERN will be found in the appendix.

2.1.4 Activating and Deactivating a Workstation

When primitives and attributes are output they are sent to all open and active workstations. When a workstation is deactivated no further output is sent to it. At the end of the application program all open workstations must be deactivated and closed before GKS itself can be closed. To activate or deactivate the workstation the calls are:

CALL GACWK(WKID)

CALL GDAWK(WKID)

Note that Input-Only and Metafile-Input workstations may not be activated.

2.1.5 Clearing the Workstation

The application has control over clearing the display surface at any time by calling:

CALL GCLRWK(WKID, COFL)

The integer COFL controls the way in which the update happens. If COFL is set to 'GALWAY' (see note at bottom of page 1), then the workstation is cleared even if there was no previous output. If COFL is set to 'GCONDI', then the drawing surface is cleared only if there has been previous output. Clearing the workstation deletes all WDSS segments stored on that workstation (segments are described in Chapter on Page 21), and on a hardcopy device the paper is advanced. For metafile interpretation by the CERN utilities this function is used to indicate ‘end of picture’.

Note that the workstation is cleared automatically when it is opened. Some implementations also clear the workstation when it is closed, but this is not mandated in the standard.

2.1.6 Deferral States

GKS allows the programmer to defer sending output to a workstation by calling the function Set Deferral State. For example, it might be more efficient to send primitives in batches, rather than one at a time. Thus, ‘As Soon As Possible’ often may not be as fast as possible! For most users of GKS the default values should be acceptable (but note that these are device-dependent). The call is:

CALL GSLS(WKID, DEFMOD, REGMOD)
The second argument, DEFMOD, controls when output should be sent to the workstation specified by WKID, and can take the following values:

- ‘GASAP’ send output As Soon As Possible
- ‘GBNIG’ send output Before Next Interaction Globally. This makes sure that the workstation is up-to-date before the next input action on any open workstation
- ‘GBNIL’ send output Before Next Interaction Locally. This makes sure that the workstation is up-to-date before the next input action on the workstation specified in the call
- ‘GASTI’ At Some Time allows GKS to choose some opportune moment to bring the workstation up-to-date

The third argument, REGMOD, controls what should happen on those workstations which require the image on the display to be regenerated in some circumstances. For example, to delete a segment on a direct view storage tube display, such as a TEKTRONIX 4014, would require the screen to be cleared and then the remaining contents re-drawn. This is called implicit regeneration. REGMOD can take the following values:

- ‘GSUPPD’ Re-drawing is suppressed until an update is forced by calling the Update Workstation function, GUWK, or by re-drawing all the segments with GRSGWK.
- ‘GALLOW’ Immediate re-drawing is allowed.

### 2.1.7 Updating the Workstation

Because of the deferral mode set, not all the output may yet have been sent to the display. The routine

```
call guwk(wkid, regfl)
```

is provided to bring the contents of the workstation WKID up-to-date. The argument REGFL can take the values ‘GPOSTP’ or ‘GPERFO’. If ‘GPOSTP’ is specified, then all deferred actions are performed at some time. If ‘GPERFO’ is specified, then all deferred actions are performed, the workstation transformation is updated, the display surface is cleared, and all segments are re-drawn as necessary. If an implicit regeneration of the image has to be performed then non-retained data (primitives not stored in segments) will be lost.

Note that a call to GUWK will not necessarily clear the workstation and re-draw all segments from scratch. However, this can be achieved by calling the function GRSGWK (see section on Page 26).

### 2.1.8 Sending Messages/Prompts

In order to display messages and prompts on the display surface the following routine is provided:

```
call gmsg(wkid, str)
```

WKID specifies on which workstation the text string STR should appear. Where and how the message is written out is both implementation and workstation-dependent.
2.2 The Drawing Primitives

Drawing primitives are the basic elements of graphics output, such as lines, text etc. GKS supports the six output primitives which are described below: polyline, polymarker, fill area, text, cell array, and the generalised drawing primitive. Each primitive has a corresponding set of attributes e.g. line type, colour index, character height, etc. The appearance of an individual primitive is governed by the attributes in force at the time when the primitive is passed to GKS, and attribute setting will be described in the following chapter. A number of example programs may be found in Appendix on Page 76 which illustrate the use of the following features.

Figure 2.1: Examples of some GKS Primitives

Note

The Polyline, Polymarker, and Fill Area primitives are specified by an array of points which may not be arbitrarily large, and the maximum size of this array is implementation-dependent. The GKSGRAL implementation at CERN restricts the number points in a Fill Area to 300. However, calls to Polyline or Polymarker can handle more points than this by internally sub-dividing the array.

2.2.1 Polyline

This draws line segments between two or more points using the currently set attributes for line style, polyline colour index and line width scale factor. The call is:

CALL GPL(N, PXA, PYA)

where N is the number of points, and PXA(N), PYA(N) are real arrays containing the X and Y values of the points in the application program’s own coordinate system (called the World Coordinates System).

2.2.2 Polymarker

This marks a sequence of points with the marker symbol selected by the currently set attributes for polymarker colour index, marker type and marker size scale factor. The marker size to which the scale factor is applied is workstation-dependent. The call is:

CALL GPM(N, PXA, PYA)

where N, PXA and PYA have the same meanings as for GPL.

2.2.3 Text

Text is displayed using the current attribute settings for text colour index, text font and precision, character-up vector, text alignment, character expansion factor, character spacing, text path and character height. The call is:

CALL GTX(PX, PY, STR)

where PX and PY are real numbers which define in World Coordinates the starting position of the text string STR.
2.2.4 Fill Area

Fill Area draws an area which is specified by a polygon. The interior of the polygon can be either not filled (hollow), filled with a solid colour, or filled with various patterns. The fill area style and colour is specified by the current setting of the fill area colour index, fill area style and fill area style index attributes. The call is:

CALL GFA(N, PXA, PYA)

where N, PXA, PYA have the same meaning as for GPL and GPM. If the first and last points are different, they are joined by a line.

2.2.5 Cell Array

The Cell Array is an array of rectangular cells with individual colours. It serves for passing raster images to GKS. Note that in general, colour values within the Cell Array do not correspond to picture elements (pixels) on a hardware display, and that all primitives, including Cell Arrays, are subject to the usual GKS transformations. The call is:

CALL GCA(PX, PY, QX, QY, DIMX, DIMY, ISC, ISR, DX, DY, COLIA)

Neither the Cell Array primitive nor the function call arguments will be described in detail in this Primer, and the reader is referred to the GKSGRAL manual, or one of the texts in the bibliography, for more information.

2.2.6 Generalised Drawing Primitive

Although at present GKS defines only six types of graphics primitives, the standard allows for these to be augmented within particular implementations by using Generalised Drawing Primitives (GDPs). Thus, the Generalised Drawing Primitive may be thought of as a ’standard way to be non-standard’, and in fact a mechanism exists for implementors of GKS to register GDPs with the International Standards Organization group responsible for GKS. The intention is that these registered GDPs will be incorporated as new primitives within future revisions of the standard. However, note that the use of GDPs may make programs non-portable to other GKS implementations. On the other hand, they do permit the GKS driver to make use of special hardware features, circle drawing for example, if these features are available.

GKSGRAL supports 6 GDPs:

1. Circle
2. Circular Arc
3. Ellipse
4. Elliptical Arc
5. Bezier Curve defined by Bezier Polygon
6. Cubic Curve defined by Interpolation Points

The call is:

CALL GGDP(N, PXA, PYA, PRIMID, IDR, DATREC)

See the GKSGRAL Manual for more details.
2.3 The Output Attributes

Before actually outputting any primitives, the application will want to specify the exact way in which the drawing appears on the display surface. This rendering process is controlled by output attributes. Output primitives have geometric and non-geometric attributes. Geometric attributes, such as the character height, affect the size and shape of a primitive, whereas non-geometric attributes are qualities such as colour, line style, etc.

Output attributes affect the appearance or rendering of primitives at the moment when the primitives are sent to GKS, and attributes are said to be bound to primitives at this time. Thus, modification of an attribute has no effect on primitives which have been output already. GKS attributes are said to be modal in character because, after setting an attribute, GKS is in a mode in which the value of that attribute will be bound to all primitives of the appropriate type which follow. Setting a different value for the attribute would then change the mode.

2.3.1 Attribute Bundles

There are two ways of specifying attributes, Bundled and Individually. Attributes may be set individually by calling the appropriate routines one at a time. As an example, for a polyline one could set the line style, the line width, and the colour by calling the routines GSLN, GSLWSC, and GPLCI, before calling GPL to output the polyline. Setting attributes individually will ensure that the primitives to which they are bound appear the same on all workstations, assuming that the workstations have sufficient capability. However, if attributes are set using bundles, then the results will be workstation-dependent.

Bundled attributes are assigned by selecting a bundle index for a particular type of primitive using the routines given in section on Page 13. The bundle index points to an entry in the appropriate workstation bundle table. Hence, each workstation has a bundle table for every primitive type, and each entry in a bundle table contains a pre-defined set of attributes for that particular primitive. For example, the first entry in the polyline bundle table may contain the attributes dotted and red, the second may contain solid, double thickness and blue, etc. Note that attribute bundles do not contain geometric attributes.

Beginners are advised to ignore attribute bundles and to set each attribute individually. However, read the next section on Aspect Source Flags before trying out a program.

As an example of why one might make use of attribute bundles, consider an application which sometimes uses a colour terminal and sometimes a monochrome one. By drawing polylines with, say, bundle table index 5, the actual appearance of the polylines will depend on the contents of polyline bundle 5 on the two workstations. Thus, the application can arrange to distinguish the polylines by using a particular colour on the colour terminal, and a particular dash pattern on the monochrome terminal, without making changes to the body of the code which draws the primitives.

By using attribute bundles to specify attributes, and assuming that the primitives have been stored in segments (segments are described in Chapter on Page 21), the application can also change the appearance of primitives after they have been output to a workstation by re-defining the contents of the bundle table. This effect can not be achieved if the attributes are set individually without deleting and re-drawing the primitives.

2.3.1.1 Aspect Source Flags

To control whether a primitive attribute should be set individually, or using a bundle table index, each primitive has a set of attributes called the Aspect Source Flags (ASFs); one flag for each primitive attribute. If the ASF for a particular attribute is set to ‘GBUNDL’, then the primitive will be bound to the
attributes in the bundle table entry pointed to by the bundle index currently in force. If the ASF for a particular attribute is set to 'GINDIV', then the primitive will be bound to the current individual attribute values for that type of primitive.

Unfortunately, the committee which designed GKS could not agree on whether the default setting for the ASFs should be bundled or individual. Thus, American implementations, such as PLOT10/GKS, tend to default the ASFs to set attributes individually, whilst European implementations, such as GKSGRAL, tend to default ASFs to bundled attributes. In order to be safe, users of GKS are advised to set their own default values for the ASFs as soon as they open GKS. This can be done by calling:

```
CALL GSASF(ASFLST)
```

where ASFLST is an array of 13 integers, one for each attribute, which must be set to 'GBUNDL' or 'GINDIV'. The bundle table index for each attribute is given in parentheses in the attribute list below. An example program which calls GSASF may be found in Appendix on Page 76.

### 2.3.1.2 Defining Attribute Bundles

It is possible to define the entries in the bundle tables by calling one of the following routines:

- **Set polyline representation**
  
  `GSPLR(WKID, PLI, LN, LWSC, PLCI)`

- **Set polymarker representation**
  
  `GSPMR(WKID, PMI, MK, MKSC, PMCI)`

- **Set text representation**
  
  `GSTXR(WKID, TXI, TXF, TXP, CHXP, CHSP, TXCI)`

- **Set fill area representation**
  
  `GSFAR(WKID, FAI, FAIS, FASI, FACI)`

- **Set pattern representation**
  
  `GSPAR(WKID, PAI, DIMX, DIMY, NCS, NRS, DX, DY, PACI)`

As arguments, each routine requires the workstation identifier (WKID) and bundle index (PLI, etc.) to be set, plus a value for each of the non-geometric attributes for that particular primitive, and which are listed below. Details of GSPAR will not be given in this Primer; see the GKSGRAL manual or one of the references in the bibliography for more information.
2.3. The Output Attributes

2.3.2 The List of GKS Attributes

2.3.2.1 Individual Attributes

- **POLYLINE**
  - LN (integer) the polyline line style (ASF 1). Set by GSLN(LN).
  - LWSC (real) the line width scale factor (ASF 2). Set by GSLWSC(LWSC).
  - PLCI (integer) the polyline colour index (ASF 3). Set by GSPLCI(PLCI).

- **POLYMARKER**
  - MK (integer) the polymarker type (ASF 4). Set by GSMK(MK).
  - MKSC (real) the marker size scale factor (ASF 5). Set by GSMKSC(MKSC).
  - PMCI (integer) the polymarker colour index (ASF 6). Set by GSPMCI(PMCI).

- **TEXT**
  - TXF (integer) the text font (ASF 7). Set by GSTXFP(TXF, TXP).
  - TXP (enumerated) the text precision (ASF 7). Set by GSTXFP(TXF, TXP).
  - CHXP (real) the character expansion factor (ASF 8). Set by GSCHXP(CHXP).
  - CHSP (real) the character spacing (ASF 9). Set by GSCHSP(CHSP).
  - TXCI (integer) the text colour index (ASF 10). Set by GSTXCI(TXCI).
  - CHUP (real) the character up vector. Set by GSCHUP(CHUX, CHUY).
  - TXAL (enumerated) the text alignment. Set by GSTXAL(TXALH, TXALV).
  - TXP (enumerated) the text path. Set by GSTXP(TXP).
  - CHH (real) the character height. Set by GSCHH(CHH).

- **FILL AREA**
  - FAIS (enumerated) the fill area interior style (ASF 11). Set by GSFAIS(FAIS).
  - FASI (integer) the fill area style index (ASF 12). Set by GSFAI(FASI).
  - FACI (integer) the fill area colour index (ASF 13). Set by GSFAI(FACI).

- **PATTERN**
  - PA (real) the pattern size. Set by GSPA(PASZX, PASZY).
  - PARF (real) the pattern reference point. Set by GSPARF(RFX, RFY).

2.3.2.2 Bundled Attributes

- **POLYLINE**
  - PLI (integer) the polyline bundle index. Set by GSPLI(PLI).

- **POLYMARKER**
  - PMI (integer) the polymarker bundle index. Set by GSPMI(PMI).

- **TEXT**
  - TXI (integer) the text bundle index. Set by GSTXI(TXI).

- **FILL AREA**
  - FAI (integer) the fill area bundle index. Set by GSFAI(FAI).
2.3.3 Specifying Line Styles For Polylines

It is possible to draw polylines solid, dashed, dotted or dashed-dotted (’GLSOLI’, ’GLDASH’, ’GLDOT’, ’GLDASD’). For example:

CALL GSLN(GLDASH) - sets the line style to dashed

The different line styles available are shown in Figure 2.2. For example:

Figure 2.2: GKS line styles

It is also possible to specify a scale factor which modifies the nominal width of lines on the workstation. For example:

CALL GSLWSC(2.0)

should double the nominal line width. Note, however, that this (Real) parameter is implementation-dependent, and may be ignored by terminal drivers as it is both difficult and expensive to emulate in software if the device does not support the feature in hardware.

2.3.4 Specifying Marker Types for Polymarkers

The five GKS Marker types, point, plus, asterisk, circle and cross (’GPOINT’, ’GPLUS’, ’GAST’, ’GOMARK’, ’GXMARK’), are demonstrated in Figure 2.3. For example:

CALL GSMK(GPOINT) - sets the marker type to point

Figure 2.3: GKS marker types

Markers may be scaled in size by calling GSMKSC. For example:

CALL GSMKSC(3.5)

will scale the following markers by 3.5 times. The implementation of this function is workstation-dependent. In particular, markers drawn on terminals by hardware may only exist in certain fixed sizes.

2.3.5 Specifying Fill Areas

Many applications need shaded or coloured areas as well as lines and points. GKS allows one to draw such an area by specifying an array of points representing a closed polygon. If the last point in the array is not the same as the first, these two will be joined. The interior style of the area can be set to one of the four values hollow, solid, pattern, and hatch, which are demonstrated in Figure 2.4. Examples for setting fill area interior style are:

CALL GSFAIS(GHOLLO)

or

CALL GSFAIS(GSOLID)
2.3. The Output Attributes

For interior style hollow the boundary polygon only is drawn as a solid line. For style solid the interior is completely filled with a uniform colour, as specified by the fill area colour index set by calling GSFACI. Workstations for devices which support area filling of polygons by hardware should normally make use of this feature. However, this is not always possible, as some monochrome terminals do not use the correct algorithm to perform the area fill.

For interior styles hatch and pattern, the particular hatch algorithm or pattern used may be chosen by specifying a fill area style index. This represents a second level of selection on the way the area is filled, and the index points into either a hatch or pattern table stored at the workstation. Thus, hatches and patterns are workstation-dependent. The fill area style index is set by calling:

```
CALL GSFASI(FASI)
```

where FASI is an integer value. To find out the effect of setting a particular index, it is necessary to consult the workstation documentation.

The GKS standard provides calls to modify patterns by setting the pattern reference point (GSPARF) and pattern size (GSPA). The pattern representation attribute bundle is set using GSPAR. The GKSGRAL package also allows the possibility to define one’s own hatch styles by calling the routine GUSHTR, although this is not a standard GKS feature. None of these routines will be described further in this Primer, and the reader is referred to the texts in the bibliography for more information.

Figure 2.4: GKS fill area styles

2.3.6 Specifying Text

Text is the most complex of the GKS primitives because there are so many different variations. The text attributes are described below. Note that only the non-geometric attributes may be set in the text bundle table. (HINT: For FORTRAN programmers, it is possible to use the Internal Write construct to convert numeric variables to character strings for output as text primitives, see section on Page 52).

2.3.6.1 Font and Precision

The text font is specified by an integer font number, and fonts are workstation-dependent. Any text font which is not supported on a given workstation is defaulted to 1. This text font is a simple, roman type font produced by stroking out each character using line segments. To find out which fonts are provided by a particular implementation one should consult the relevant reference manual.

The text precision attribute determines how closely the chosen font has to obey the specifications of the other text attributes. The choices for text precision are: string, char and stroke (’GSTRP’, ’GCHARP’, ’GSTRKP’). In string precision the text string should be placed as close as possible to the position specified by the primitive call. None of the other text attributes need be taken into account except the approximate size specified by character height. Thus, if string precision is specified, the implementation is usually free to make use of hardware characters which can be drawn much faster than generating the fonts in software. If char precision together with a workstation-provided font is chosen, GKS will try a workstation dependent approximation. In stroke precision GKS has to follow precisely the attribute specifications, and always defaults to font 1 if the requested font does not exist. Note that an implementation of GKS may provide stroke precision text on a particular workstation even if the application specified only a lower precision. Both the text font and the text precision are set by the same routine:
2.3.6.2 Expansion Factor

The character expansion factor, a real number, causes each character to appear ‘fatter’ or ‘thinner’ than normal. The default value is 1.0. The height of the character is not affected, nor is the space between the characters.

CALL GSCHXP(CHXP)

2.3.6.3 Spacing

The character spacing attribute defines the amount of extra blank space to be inserted between adjacent characters of a text string. This is a real number defined as a fraction of the character height. The default value is 0.0. A positive character spacing spreads the letters out, a negative one makes the letters overlap.

CALL GSCHSP(CHSP)

2.3.6.4 Character-Up-Vector

The char-up vector defines the orientation of the text. The text is written from left to right along a line perpendicular to the char-up vector, which is specified by its X and Y components (two real numbers):

CALL GSCHUP(CHUX, CHUY)

The effect of choosing different combinations of text font, text precision and character-up-vector is shown in Figure 2.6.

2.3.6.5 Alignment

The text string as a whole is positioned relative to the point specified in the GTX primitive call according to the current setting of the text alignment attribute. The call to GSTXAL has two parameters for horizontal and vertical alignment:

CALL GSTXAL(TXALH, TXALV).

The horizontal alignment is either normal, left edge, centre, or the right edge of the text string (‘GAH-NOR’, ‘GALEFT’, ‘GACENT’, ‘GARITE’). The vertical alignment is either normal, top, cap line, half, base line, or bottom (GAVNOR’, ‘GATOP’, ‘GACAP’, ‘GAHALF’, ‘GABASE’, ‘GABOTT’). The alignment attribute is illustrated in Figure 2.6. The selection of normal for either horizontal or vertical alignment results in GKS choosing the most appropriate value depending on the current Text Path.
2.3. The Output Attributes

2.3.6.6 Text Path

The text path describes the direction in which the text string is written. The possible values are Right, Left, Up and Down (‘GRIGHT’, ‘GLEFT’, ‘GUP’, ‘GDOWN’). The default for the text path is perpendicular to the up vector (i.e. ‘GRIGHT’).

CALL GSTXP(TXP)

2.3.6.7 Height

The character height is set in World Coordinates by calling GSCHH. Increasing the height automatically causes a corresponding increase in the width of the character.

CALL GSCHH(CHH)

Note that the use of World Coordinates to specify the character height may cause problems if the normalization transformation chosen has a very large or very small range in Y. In this case it is possible that when using a metafile as an output workstation the number of digits required to specify the character height may be more than the range used to define real numbers within the metafile. If it would be inconvenient to modify the range of World Coordinate Y values, then another possible solution to the problem is to superimpose two viewports, one on top of the other. All characters may then be drawn on one of the viewports which has been defined using a more reasonable normalization transformation.

2.3.7 Specifying Colour

Colour is specified separately for each primitive type by a colour index value.

Polyline CALL GSPLCI(PLCI)
Polymarker CALL GSPMCI(PMCI)
Text CALL GSTXCI(TXCI)
Fill Area CALL GSFACI(FACI)

The colour index, instead of defining a colour directly, points to an entry in a colour look-up table which is workstation-dependent. If the colour is an attribute of a primitive, then one can specify a colour index either as an individual attribute or as part of a bundle table. The size of the colour table is workstation-dependent, but the table always contains the entries 0 and 1. The background colour is 0, and the default foreground colour is 1. Each entry greater than 1 defines an additional foreground colour. If a colour index greater than the maximum is specified, then the default value 1 is taken, which ensures that a program written using colour will run on a monochrome device.

To set entries in the colour look-up table one must call the function Set Colour Representation (GSCR). It takes as parameters the workstation identifier, the colour index to be set, and a value (from 0 to 1.0) for each of the red, blue, and green intensities. The workstation then uses the closest available colour to that which has been requested.
CALL GSCR(WKID, INDEX, RED, GREEN, BLUE)

The highest possible value for the colour index (NCOLI) depends on the capabilities of the device and can be inquired using GQCF and specifying the workstation type:

CALL GQCF(WTYPE, ERRIND, NCOLI, COLA, NPCI)

The parameter COLA (‘GMONOC’ or ‘GCOLOR’) indicates whether or not colour is available, and NPCI specifies the number of pre-defined colour indices.

The colour index can also be used if part of a previously drawn picture is to be erased. The part to be erased should be re-drawn in the background colour (index=0). All other attribute settings and transformations must be exactly as they were when the picture was produced initially. However, even in this case, the trick may not work if the primitive being erased overlays another primitive, and not just the background.

Colour may be part of a fill area pattern, in which case different parts of the pattern may be defined to be different colours. For the cell array primitive an array of colours is specified as part of the primitive itself.

### 2.4 Coordinates and Transformations

GKS defines three coordinate systems:

1. **WORLD** Coordinates (WC)
2. **NORMALIZED DEVICE** Coordinates (NDC)
3. **DEVICE** Coordinates (DC)

The application program specifies the coordinates of points in primitive calls using a cartesian coordinate system with whichever scale is the most convenient. This system is called the **World Coordinate System**. The WC range can go from negative to positive infinity along both axes, which gives the application an unlimited choice of coordinates. The definition of the WC space is independent of any graphics output device and can be different for each application as the requirements dictate.

All world coordinates are transformed by GKS to a ‘virtual’ device space, the **Normalized Device Coordinate** (NDC) space. The NDC space ranges from 0 to 1 in both x and y. This space is device independent and appears identical for all workstations in the system. A transformation from WC to NDC is called a **Normalization Transformation**. GKS allows more than one such transformation to be defined, and so an application may draw each part of a picture in its own WC system, and then map them into a single NDC space. See [2.7 and 2.8](#).

The third coordinate system is that of the hardware device. GKS uses specific information about the graphics output device to transform normalized device coordinates into coordinates that the device itself understands. Usually the units in the device dependent space will be metres or raster units. The translation from NDC to DC is called the **Workstation Transformation**.

#### 2.4.1 Normalization Transformation

This transformation is specified by mapping a *Window* in the application’s WC space to a *Viewport* in NDC space. Both the WC window and the NDC viewport are rectangles parallel to the x,y axes. They are specified by calls to Set WiNdow and Set ViewPort:
2.4. Coordinates and Transformations

Because there may be several Normalization Transformations, the integer value TNR indicates to which one the call applies. The other values are real numbers defining the bottom left and top right corners of the two rectangles. The window is specified in the WC system, and the viewport in the NDC system. If the x and y sides of the window and the viewport do not have the same Aspect Ratio a distortion effect will take place. This may or may not be intentional!

As has been mentioned, it is possible to generate a complex picture with different world coordinates for different parts of the image, or to place several pictures on different areas of the screen. This is achieved by specifying multiple normalization transformations. Each transformation is assigned a transformation number (TNR) for which the range is 0 to some implementation-dependent maximum (usually about 20). Thus, the transformation corresponding to each number specifies a separate window-to-viewport mapping. If the viewports overlap then their contents will be superimposed.

The default transformation number is 0, for which the WC and NDC coordinate systems are the same. Transformation 0 may not be modified, and so its use is always equivalent to calling:

\[
\begin{align*}
\text{CALL GSWN} & \quad (TNR, 0.0, 1.0, 0.0, 1.0) \\
\text{CALL GSVP} & \quad (TNR, 0.0, 1.0, 0.0, 1.0)
\end{align*}
\]

The application specifies which normalization transformation is active at a given time by a call to GSELNT (SELection Normalization Transformation):

\[
\text{CALL GSELNT(TNR)}
\]

All primitives created subsequently will be transformed according to this transformation until a different one is selected.

Figure 2.7: Windows and Viewports

2.4.2 Workstation Transformation

The method of defining which portion of NDC space is to appear on a specific workstation is similar to the way a viewport is positioned in NDC space (the Normalization Transformation). The Workstation Transformation defines that part of NDC space which will be visible, and where it will appear on the display surface. It is set by calling the functions Set WorKstation WiNdow and Set WorKstation ViewPort:

\[
\begin{align*}
\text{CALL GSWKWN} & \quad (WKID, WXMIN, WXMAX, WYMIN, WYMAX) \\
\text{CALL GSWKVP} & \quad (WKID, VXMIN, VXMAX, VYMIN, VYMAX)
\end{align*}
\]
Set Workstation Window specifies in NDC coordinates the area of the NDC space to be output to the device. Set Workstation Viewport specifies in Display Coordinates (DC) where on the device the window will appear. These functions may be called at any time. However, whether or not the effect is immediately visible is workstation-dependent. For example, if changing the workstation transformation requires an implicit image regeneration, then the timing of when it will take place is affected by the current deferral mode (see section on Page 7).

The aspect ratio for the workstation window and the workstation viewport must always be the same. If they are not, then the specified transformation is ignored, and the complete workstation window is displayed on the device in the correct aspect ratio. A rectangle is used with the bottom left corner set to the bottom left corner of the specified viewport, and which is as large as possible.

The default setting for the workstation transformation is to map the whole unit square of NDC onto the largest square possible for the workstation. For rectangular displays, this means that not all of the display surface is used. Conversely, if the workstation transformation is set so that all of a rectangular display is used, then either: (1) the normalization transformation can not preserve the aspect ratio, or: (2) not all of NDC space can be used.

To inquire the range of device coordinates corresponding to a particular workstation type, it is possible to call the function GQDSP (inQuire Display SPace). (Note: The 3D version is called GQDVOL.)

CALL GQDSP (WTYPE, ERRIND, DCUNIT, RX, RY, LX, LY)

The routine returns the units in which the display surface is measured (DCUNIT), and also the maximum x and y values in each direction. Some devices, for example hardcopy plotters, are measured in metres (DCUNIT='GMETRE'), so the routine will return the actual size of the device. Other devices, for example graphics terminals, will be defined in raster or some other units (DCUNIT='GOTHU'). RX, RY are real device coordinates, and LX, LY are in integer raster units.

As an example of how to use this routine, below is a fragment of code which allows an application to draw on the whole of a rectangular display surface with a 'Landscape' aspect ratio (width > height). See also the example in Appendix on Page 82.

C Inquire the Workstation Display Surface Size
   CALL gqdsp (wtype, errind, dcunit, rx, ry, lx, ly)
C
C Set the Workstation Window and Viewport (assume x > y)
   CALL gswkwn(wkid, 0.0, 1.0, 0.0, ry/rx)
   CALL gswkvp(wkid, 0.0, rx, 0.0, ry)
C
C Make sure Workstation Transformation is used
   CALL guwk (wkid, gperfo)
C
C Set Window and Viewport for Normalization Tfrm 'trn'
   CALL gswn (trn, xmn, xmx, ymn, ymx)
   CALL gsvp (trn, 0.0, 1.0, 0.0, ry/rx)
C
C Select this Normalization Tfrm for future primitives
   CALL gselnt(trn)

Figure 2.8: Normalization and Workstation Transformations
2.5. Segments

2.4.3 Clipping

Clipping is the exclusion of those primitives, or parts of primitives, which lie outside a specified window or viewport. That is to say, if clipping to a particular window is enabled, then only those parts of primitives which lie within the boundaries of the window will be displayed. Both the normalization transformation viewport (in NDC space) and the workstation window can be used to clip a picture. Clipping at the boundary of the normalization transformation viewport (clipping rectangle) effects the image displayed on ALL workstations. However, clipping at the boundary of the workstation window only affects the image on the display surface belonging to that particular workstation. This is illustrated in Workstation Viewport 1 in [7.1.

The application can control whether clipping at the boundary of the clipping rectangle is enabled or not by calling the routine GSCLIP(VAL). This routine takes the value ’GNCLIP’ (no clip) or ’GCLIP’ (clip), and the default is to clip. Clipping at the workstation window may not be switched off.

2.5 Segments

GKS provides a way to collect and store together the primitives that make up all or part of a picture. Such a collection of primitives is called a segment, has a unique name, and may be manipulated as a unit in various ways. Only one segment can be open at a time, and once a segment has been closed further output primitives cannot be added to it, nor can the primitives in it be modified.

Any non-retained data output whilst there is no open segment will be lost if the screen is cleared for some reason. There are several circumstances when this may be useful. For example, a message written on the screen may have only limited validity, or one may wish to output a very complicated image with too much data to be stored, or which needs only to be displayed once. However, much of the power of GKS comes from the ability to interactively modify the image without always having to regenerate the complete picture. To make use of this feature, primitives must be stored in segments.

2.5.1 Segment Storage

Every workstation in a GKS output level 1 or 2 implementation has associated with it a Workstation Dependent Segment Store (WDSS). At the time a segment is created, it is stored in the WDSS of all open and activated workstations. The WDSS may be physically located in the Graphics Terminal, in which case there can be a substantial improvement in performance. Of course, care must be taken not to exceed the available memory space.

In addition to WDSS, GKS output level 2 implementations have also a Workstation Independent Segment Store (WISS). WISS is a mechanism for storing segments in a workstation-independent way, and allows segments to be re-used and manipulated on different workstations at different times. WISS is treated like any another workstation, just like a terminal for example, and if it is open and activated when a segment is created, then that segment will be stored there as well as on the WDSSs of the other active workstations. For example, one might store an object in WISS at the same time as displaying it on a terminal. Then, at a later time, it would be possible to make a hardcopy of the object by extracting the segment(s) in which it was stored from the WISS and copying it (them) to a plotter. A transformation or segment deletion affects the segment wherever it is stored.

Before primitives are stored in WDSS or WISS they first undergo the normalization transformation currently in force. Also, when a segment is created, the current clipping rectangle together with the clipping flags, are stored in the segment, but clipping is not performed on the primitives when they are stored, but only when the contents of the segment are output to a device.
2.5.2 Segment Creation, Deletion, and Re-naming

To use GKS segmentation, first a segment must be created by calling:

```
CALL GCRSG(SGNA)
```

where SGNA is an integer segment name. A segment called SGNA will be created on all active workstations, including WISS, and will remain open and store all primitives and attributes which are output until one calls:

```
CALL GCLSG
```

Only a single segment may be open at one time.

Segments may be renamed, or deleted on a particular workstation or all workstations by the following calls:

```
CALL GRENSG(SGNOLD, SGNNEW)
CALL GDSGWK(WKID, SGNA)
CALL GDSG(SGNA)
```

Once a segment has been deleted the same segment name may be re-used. Note that the Clear Workstation function deletes all the segments stored in WDSS on that workstation.

The following fragment of code shows the use of segments. See also the example in Appendix on Page 82.

```fortran
---------
CALL gcrsg(1) ! Create segment 1
CALL objone ! Draw an object
CALL gclsg ! close segment
CALL gcrsg(2) ! Create segment 2
CALL objtwo ! Draw an object
CALL gclsg ! Close segment
---------
CALL guwk(wkid, gperfo) ! Update Workstation
---------
```

2.5.3 Segment Names

As indicated above, the FORTRAN binding requires segment names to be integers, and these may be restricted by an implementation to a particular range of values (1-32763 for GKSGRAL). However, because segment names are specified by integers does not imply that an implementation has to make available as many segments as there are names, and the maximum number of available segments is also an implementation-dependent feature which may be much smaller than the maximum legal name. (Thus, one can not assume that an implementation stores name \(N\) as the \(N\)th element of an array.) GKS does not provide inquiry functions to obtain either of these maximum values, so one must consult the relevant documentation.

The GKS standard provides a function, GQSGUS, to inquire the set of Segment names in Use. However, as it is difficult to work with variable length sets in FORTRAN, the FORTRAN binding has modified this function to allow only the inquiry of the name of one element of the set at a time:
CALL GQSGUS(N, ERRIND, NSG, SGNA)

where
N  is the N’th segment in the set (an input parameter)
ERRIND is an error number
NSG is the total number of segments in use
SGNA is the name of the N’th segment

Thus, in FORTRAN, it is necessary to call GQSGUS in a loop from 1 to NSG in order to obtain a list of all the segment names in use.

Note that the function:
CALL GQSGWK(WKID, N, ERRIND, NSG, SGNA)

works in a similar way to GQSGUS but provides the list of segment names associated with workstation WKID.

### 2.5.4 Segment Association, Copying, and Insertion,

To view the contents of segments stored in WISS it is necessary to transfer them to a real device. This can be accomplished in three ways:

- **Association**
  Associate SeGment to WorKstation retrieves a segment stored in WISS and sends it to the designated workstation as if this workstation had been activated when the segment was first created. It is thus stored in the WDSS of the designated workstation. Nothing may be added to the segment. There must be no open segment when this function is called.
  
  CALL GASGWK(WKID, SGNA)

- **Copying**
  Copy SeGment to WorKstation transforms the contents of the designated WISS segment using the segment transformation (described below), and then sends the result to the workstation specified. The clipping volume of the copied segment is also sent to the workstation. Note that the copied segment itself is not transferred; its contents are just drawn on the output device without being stored in WDSS. Thus, there must be no open segment when this function is called.
  
  CALL GCSGWK(WKID, SGNA)

- **Insertion**
  INsert SeGment transforms the contents of the designated WIS segment using both the transformation stored in the segment header and then the matrix provided in the call (MTX), before inserting the results into the output stream as if they had come directly from the application. Thus, the output of Insert Segment goes to all activated workstations, including the WISS, and will be stored in a segment if one is open. All clipping rectangles and indicators, etc. are ignored. The transformation matrix, MTX, may be produced using the routine GEVTM as described below.
  
  CALL GINSG(SGNA, MTX)

The flow of graphics data between WISS, WDSS and the application is shown in 2.9.
2.5.5 Segment Attributes

The appearance of segments depends on the following segment attributes:

- segment transformation
- visibility
- highlighting
- segment priority
- detectability

Segment attributes may be modified after the segment has been closed, and the changes will become visible on all active workstations on which the segment has been stored. However, the timing of when these changes take place is workstation-dependent, and may also be affected by the deferral mode which is in force (see sections on Page 7 and on Page ??). This is because some workstations may require picture regeneration to produce the new attribute state on the display.

2.5.5.1 Segment Transformations

A segment transformation is a transformation of all the coordinates within a segment and is performed by a 2 x 3 matrix stored in the segment header. It maps from NDC to NDC. As an example of the use of segment transformations, consider a circuit design application which has symbols for transistors, capacitors, resistors, etc. Such an application would store each symbol in a separate segment, and then call INsert SeGement specifying a transformation matrix in order to duplicate a particular symbol at the positions and orientations required.

When a segment is created GKS sets a default null transformation which leaves the original coordinates unchanged. Before setting a transformation it is necessary to evaluate the transformation matrix by either using one’s own algorithm, or by using the routines:

\[
\begin{align*}
\text{CALL GEVTM}(X_0, Y_0, DX, DY, PHI, FX, FY, SW, MXOUT) \\
\text{and/or} \\
\text{CALL GACTM}(MXIN, X_0, Y_0, DX, DY, PHI, FX, FY, SW, MXOUT)
\end{align*}
\]

GEVTM evaluates a matrix (MXOUT), whilst GACTM accumulates changes to an existing matrix (MXIN). Both routines require the definition of:

- $X_0, Y_0$ (real) a fixed reference point about which 2D rotations take place.
- $DX, DY$ (real) a translation (or shift) vector.
- $PHI$ (real) an angle of rotation about $X_0, Y_0$.
- $FX, FY$ (real) X and Y scale factors.
- $SW$ (enumerated) a switch specifying whether the reference point and shift vector are given in World Coordinates or Normalized Device Coordinates (‘GWC’ or ‘GNDC’).
2.5. Segments

The transformation is composed in the order: scale, rotate, shift. In the case of GACTM, the matrix MXIN is pre-concatenated with that formed from the scale, rotate, and shift parameters, so MX-OUT = SHIFT * ROTATE * SCALE * MXIN.

Once the transformation matrix has been evaluated, it may then be Set in the Segment by calling the routine:

CALL GSSGT(SGNA, MTX)

An example of using a segment transformation may be found in Appendix on Page 82.

2.5.5.2 Visibility

Segment Visibility (GSVIS) determines whether or not the segment is displayed; the default is for the segment to be visible. The values are 'GINVIS' or 'GVISI'. As an example of its use, messages or icons could be created in segments which would normally be invisible, but which could be made visible at the appropriate time by the application program. The call is:

CALL GSVIS(SGNA, GVISI)

Note that if a segment is invisible it is not detectable (cannot be picked), even if detectability is enabled. Also, even if a segment has visibility enabled, it may not actually be drawn if the deferral state is set to something other than 'GASAP' (see section on Page 7).

2.5.5.3 Highlighting

Many display systems have some means for highlighting graphical output, eg by making it brighter, or by causing it to blink. The implementation of this attribute is dependent on the device being used. Note that in order for a Segment to be Highlighted (GSHLIT), it must first be made visible. The default is not highlighted. The possible values are 'GNORM' or 'GHILIT'. The call is:

CALL GSHLIT(SGNA, GHILIT)

2.5.5.4 Priority

Segment Priority, a real number in the range 0.0 to 1.0, enables the control of the order in which segments are re-drawn when the picture has been changed. It also controls the order in which segments are picked. If parts of segments overlap, then the segment with the highest priority will be picked first and displayed on top of all the others. If two segments with the same priority occupy the same area of the display surface, then the last one will be drawn on top. The default priority is 0.0. The call is:

CALL GSSGP(SGNA, PRIOR)

The number of segment priority levels available (NSGP) for a particular workstation, WTYPE, may be inquired using:

CALL GQSGP(WTYPE, ERRIND, NSGP)
2.5.5.5 Detectability

Segment DeTECtability controls which segments can be picked by the operator using a Logical Pick Device, and which ones cannot. Values are 'GUNDET' or 'GDETEC'. If a segment is detectable, it can be picked. The default setting is un-detectable. The call is:

\[ \text{CALL GSDTEC(SGNA, GDETEC)} \]

Note that if a segment is invisible it is not detectable (cannot be picked), even if detectability is enabled.

2.5.6 The Pick Identifier

It is possible to attach an integer Pick Identifier to primitives which are stored in segments using the routine:

\[ \text{CALL GSPKID(PCID)} \]

This inserts one or more Pick Identifiers (PCIDs) into a segment, and these are associated with the subsequent primitives. If one of these primitives is picked sometime later using a Logical Pick input device (see on Page 26), then one of the pieces of information returned to the application is its PCID.

As the application is free to choose the value of the Pick Identifier, this provides a second level of naming, and provides a mechanism to refer back to an application database. For example, if the identifiers 1 to 4 were attached to the four wheels of a car, then picking wheel number three would return the identifier '3' to the application program.

2.5.7 Segment Re-drawing

It is possible to force all segments within the WDSS on a particular workstation to be re-drawn by calling the routine Re-draw SeGments on WorKstation:

\[ \text{CALL GRSGWK(WKID)} \]

The routine clears the screen, performs all deferred actions, and re-draws all segments. All non-retained data is lost. Possible reasons to re-draw all segments are:

1. if a segment had been over-written or deleted and it is desired to regenerate a clean image;
2. if one wishes to remove all non-retained data, system messages, etc;
3. if, on this workstation, image regeneration is required in order to display the effect of modifying a segment attribute, and implicit regeneration is switched off.

The action of this routine differs from that of Update WorKstation (GUWK), which may or may not re-draw segments which have not been changed.

2.6 Graphical Input

2.6.1 Logical Input Devices

GKS organizes data that can be input to an applications program into six types, each related to a Logical Input Device. The actual physical input devices are mapped onto these logical devices, which makes it possible for GKS to organize the different forms of data in a device-independent way, and thus helps to make the code more portable. A logical input device is identified by 3 items:
2.6. Graphical Input

1. a workstation identifier
2. an input class
3. a device number

The six input classes and the logical input values they provide are:

**LOCATOR** Returns a position (an x,y value) in World Coordinates and a Normalization Transformation number corresponding to that used to map back from Normalized Device Coordinates to World Coordinates. The NT used corresponds to that viewport with the highest *Viewport Input Priority* (set by calling GSVPIP). **Warning:** If there is no viewport input priority set then NT 0 is used as default, in which case the coordinates are returned in NDC. This may not be what is expected!

```
call gsvpip(tnr, rtnr, relpri)
```

- **TNR** Transformation Number
- **RTNR** Reference Transformation Number
- **RELPRI** One of the values `GHIGHR` or `GLOWER` defined in the Include File, ENUM.INC, which is listed in the Appendix on Page 100.

**STROKE** Returns a sequence of (x,y) points in World Coordinates and a Normalization Transformation as for the Locator.

**VALUATOR** Returns a real value, for example, to control some sort of analogue device.

**CHOICE** Returns a non-negative integer which represents a choice from a selection of several possibilities. This could be implemented as a menu, for example.

**STRING** Returns a string of characters from the keyboard.

**PICK** Returns a segment name and a pick identifier of an object pointed at by the user. Thus, the application does not have to use the locator to return a position, and then try to find out to which object the position corresponds.

2.6.2 Prompt/Echo Types and Triggers

A Prompt and an Echo type is defined for each logical input device. For example, enabling the pick device might prompt with a cursor of a particular shape which would track the tablet or mouse to constitute an echo. Acceptance of a trigger by the application, hitting a key, for example, causes feedback via an acknowledgment process. Thus, the picked object could be made to blink. Different prompt/echo types can be set when initializing the device. They are implementation and workstation-dependent, so see the relevant reference manual for details.

2.6.3 Input Modes

Logical input devices can be operated in three modes: *Request*, *Sample*, and *Event* (`GREQU`, `GSAMPL`, `GEVENT`). In Request mode the application enables a device and then waits for the user to trigger input, perhaps by pushing a key. This is similar to issuing a FORTRAN READ. In Sample mode the application can measure the current value of the device, for example the locator position, without waiting for a trigger. Finally, in Event mode, the application can enable a set of several logical devices simultaneously.
Output measures from devices which have been triggered will be placed in an *Event Queue*, from whence they can be extracted by the application.

Note that in GKS input level 'b' only Request mode input is available, and that it is not possible to request GKS input and issue a FORTRAN I/O operation at the same time. (However, although not required by the standard, the GKSGRAL level 'b' implementation provides Sample input for some devices.)

### 2.6.4 Request Input Functions

To request input one of the following calls needs to be made.

**Locator**

CALL GRQLC(WKID, LCDNR, STAT, TNR, PX, PY)

- **WKID** (I) Workstation identifier
- **LCDNR** (I) The number of the logical locator device (usually 1).
- **STAT** (O) Error status (integer)
- **TNR** (O) The Normalization Transformation number used to convert the input position to World Coordinates.
- **PX, PY** (O) The returned coordinates in WC.

**String**

CALL GRQST(WKID, STDNR, STAT, LOSTR, STR)

- **WKID** (I) Workstation identifier
- **STDNR** (I) The number of the logical string device (usually 1).
- **STAT** (O) Error status (integer)
- **LOSTR** (O) string length
- **STR** (O) The returned character string

**Stroke**

CALL GRQSK(WKID, STDNR, N, STAT, TNR, NPX, PX, PY)

- **WKID** (I) Workstation identifier
- **STDNR** (I) The number of the logical stroke device (usually 1).
- **N** (I) Maximum number of points (size of point arrays).
- **STAT** (O) Error status (integer)
- **TNR** (O) The Normalization Transformation number used to convert the input position to World Coordinates.
- **NPX** (O) Number of points returned.
- **PX, PY** (O) The returned coordinate arrays (WC).

**Valuator**

CALL GRQVL(WKID, VLDNR, STAT, VAL)

- **WKID** (I) Workstation identifier
- **VLDNR** (I) The number of the logical valuator device (usually 1).
- **STAT** (O) Error status (integer)
- **VAL** (O) The returned value (real number).

**Choice**

CALL GRQCH(WKID, CHDNR, STAT, CHNR)

- **WKID** (I) Workstation identifier
- **CHDNR** (I) The number of the logical choice device (usually 1).
- **STAT** (O) Error status (integer)
2.6. Graphical Input

CHNR  (O) The returned choice number (integer).

Pick   CALL GRQPK(WKID, PCDNR, STAT, SGNA, PCID)

WKID   (I) Workstation identifier.

PCDNR  (I) The number of the logical pick device (usually 1).

STAT   (O) Error status (integer)

SGNA   (O) Picked segment name.

PCID   (O) Pick identifier (integer set by GSPKID).

2.6.5 Initializing Logical Input Devices

For each of the input classes there is an initialization function which can change the values set as default by GKS for use in Request Mode (and which should be called after the mode has been set). The function calls are:

Initialize locator CALL GINLC(...)

Initialize string CALL GINST(...)

Initialize stroke CALL GINSK(...)

Initialize valuator CALL GINVL(...)

Initialize choice CALL GINCH(...)

Initialize pick CALL GINPK(...)

For all the input classes the echo area can be changed (i.e. the portion of the display surface where the prompt appears and which can accept input from the operator). For Locator and Stroke input the initialization function sets the initial normalization transformation and the initial locator position. The initialization functions also select the required prompt/echo type.

A detailed description of the initialization functions is outside the scope of this Primer, so those readers who wish to dig somewhat deeper should consult the reference manual for the GKS implementation being used. An example using GKSGRAL may be found in Appendix on Page 80.

2.6.6 Input Inquiry Functions

There are two types of inquiry functions for GKS input. The first is Inquire Device State, and the information is obtained by calling GQCHS, GQLCS, etc. The second is Inquire Default Device Data and the information is obtained by GQDVL, GQDST etc. There is also a function GQLI which can inquire the number of available logical input devices. A detailed description of these functions is outside the scope of this Primer, and so for more information the reader is referred to the GKSGRAL manual or one of the texts in the bibliography.
Chapter 3: GKS Metafiles

GKS provides metafiles for the storage of graphical information. Their principal uses are:

1. transporting graphical information between computer systems
2. transporting graphical information from one site to another (by magnetic tape for example)
3. device spooling, e.g. for a plotter

There is no official ISO standard for writing a GKS metafile. However, in Appendix E of the ISO GKS Functional Description document a metafile format is described, and its use is recommended. A GKS metafile created using this format is known as an Appendix E metafile. Unfortunately, not all implementations follow the Appendix E format, and so metafiles created by different GKS packages may be incompatible. In fact, even different examples of Appendix E metafiles may be incompatible due to variations in the file record lengths, etc.

3.1 Writing Metafiles

A GKS metafile is produced by a standard GKS output-only workstation. The workstation must first be 'opened' (GOPWK), then 'activated' (GACWK), and all graphical information following these calls is recorded on the metafile as a series of items (listed in an appendix of the GKS/GKS-3D Primer) until a 'deactivate workstation' is encountered (GDAWK). Hence the application must control it in the same way as a terminal or a plotter. Clearly, some of the workstation inquiry functions, such as Inquire TextExtent, can not be used because this knowledge depends on the device(s) ultimately chosen to interpret the metafile. Thus, a GKS metafile does not record a complete 'picture' so much as all the actions performed to make such a picture. If the application deletes segments and then draws different ones, all this will be recorded on the metafile if it is active.

This last point has a number of ramifications. Because the metafile captures all the output generated whilst it is active, it could be used to record a dynamic sequence, like a cartoon, as long as it is re-interpreted onto a dynamic display system. However, if the object of the exercise is to interpret the metafile onto a hardcopy device, it is not clear what the interpreter would be expected to do with, for example, a Delete Segment operation! Thus, for this kind of use, the metafile workstation should only be activated when a finished picture is ready to be output, perhaps by copying the image from WISS. A classic mistake is to leave the metafile workstation activated whilst one works interactively on a terminal, and then crash when all the available disc space is used up.

To open an Appendix E metafile workstation the call is:

CALL GOPWK(WKID, CONID, WTYPE)

where the parameter WTYPE specifies the metafile workstation which is defined in the documentation for the GKS implementation in use (see Appendix on Page 104 for the GKSGRAL values).

The metafile will be written to the logical unit number corresponding to the connection identifier (conid) in the GOPWK call (see section on Page 62 for legal values). On VM/CMS, a conid of XX with no OPEN or FILEDEF statements would result in a metafile name 'FILE FTXXF001'. A convenient way to declare FILEDEFs from a FORTRAN program is to use the CERN library routine VMCMS (code Z305). However, Version 2.3 of the VM/CMS FORTRAN compiler provides the possibility to specify the file name in the OPEN statement, and this is now the recommended mechanism. A '/' is required before the filename:
3.2. Shipping Metafiles

One of the fundamental uses of a metafile is that it can be stored in a device independent manner and transferred to other hosts and other sites where it can be subsequently interpreted. Metafiles at CERN are normal text files written in 80 character records, which makes transfers between host computers straightforward. However, it should be mentioned that for transfers from VAX/VMS to VM/CMS using NFT, the qualifier '/CR' is essential if the file has not been opened using the parameter CARRIAGE-CONTROL='LIST'. The REXX exec below gives an example of an appropriate metafile transfer using interlink:

/* Interlink metafile transfer */
'exec nft receive vxcern::disk$gg:[userid]gts.met gts metafile a/cr'

3.3 GRVIEW and the Interpretation of Metafiles

To use the graphical information stored in the metafile it must be interpreted so that the original picture can be re-created on whatever device the application requests. GKS provides three functions in order to interpret metafiles, and these may be called by applications which require to manipulate the contents of a metafile explicitly:

GGTITM  get item from metafile
GRDITM  read item from metafile
GIITM   interpret item from metafile

In order to view a GKS metafile on a terminal a program is available called GRVIEW. This is an interactive program which allows the user to view and/or edit the pictures (frames) in one or more metafiles. The pictures may be viewed sequentially, or it is possible to skip frames to get to a particular picture. If the frames have been named using the GCNAME function, then it is possible to search for pictures by name, and also to make a directory list of all pictures on the metafile.

After viewing a picture the user has the possibility to write it out onto a GKS metafile, or in various other formats, including PostScript and Tektronix 4014 escape codes. This allows sub-sets of the pictures on the original file to be produced. There is also a feature provided which allows the position and scale of
the Workstation Viewport to be re-set. Thus, the size of the output picture can be changed to suit better a particular device.

Without being quite so sophisticated, GRVIEW attempts to provide the sort of capabilities available on commercial products such as MACDRAW, except that GRVIEW runs on VAX, APOLLO, and IBM computers with any GKS supported terminal. Having said this, it should be clear that the ‘user friendliness’ of an editor is much greater on a colour APOLLO or VAX workstation, or on a Tektronix 4207, than on a machine with only the capabilities of a Tektronix 4014.

GRVIEW operates in one of three modes, which are selected on the command line:

1. **VIEWING** mode is the default. In this case GRVIEW is used to allow the pictures on the metafile to be displayed on a graphics terminal. In this pure viewing mode GRVIEW does not make use of any internal picture storage and so will run more efficiently.

2. **COPY** mode also allows the input metafile to be displayed, but in addition frames may be selected for copying onto an output file. This may be another metafile, a PostScript file, or a file containing Tektronix 4014 escape sequences. In copy mode the contents of each picture may not be changed, but each picture may be named (if it is not already), scaled in size, and several pictures may be packed onto a single page.

3. **EDIT** mode allows the pictures on an input metafile to be edited, or for a completely new picture to be generated from scratch.

Depending on the system in use, GRVIEW allows parameters to be provided on the command line, or via an interactive dialogue. The program also prompts the user to provide any missing information. The user now **HAS** to define which terminal type is being used because the diversity of incompatible terminals available does not permit a useful default to be chosen. There are also features which warn the user if too many GKS errors have been produced (to avoid filling up the user’s file space with messages), and which allow VM/CMS users to regain control after a pre-set number of graphics operations as VM has no simple interrupt facility.

More details of how GRVIEW operates may be found in reference [9], or by typing **HELP GRVIEW** or **FIND GRVIEW**.

### 3.4 GRCONV and the Conversion of Metafiles

As described in section on Page 34, metafiles may be used for the production of hardcopy output via the command **GRPLOT**. However, there are output devices for which there may not be support within GRPLOT, or it may be desired to combine pictures into documents existing in a particular output format or Page Description Language. For these and other reasons it is often useful to be able to transform a metafile into a different format, and to accomplish this the utility **GRCONV** has been written. Note, however, that keeping the original picture in the form of a metafile is the most flexible, as this does not place any restrictions on its final use.

**GRCONV** converts the input metafile to a new format which is stored on one or more output files. For example, GRCONV can produce as output normal or Encapsulated PostScript files, files of Tektronix 4014 escape sequences, or bit maps in IBM 3812 format. The PostScript files may be printed at CERN using the **XPRINT** command, as well as being used as a mechanism for the transmission of pictures for printing at external institutes. Pictures encoded in Encapsulated PostScript format (see section on Page 39)

---

1Will be installed on VM/CMS if there is a demand.
also may be incorporated into documents produced by SGML, BookMaster, or TeX text processing systems. The Tektronix 4014 escape sequences may be used to drive some laser printers, including the DEC LN03, which do not support PostScript.

GRCONV operates in one of three modes: Interactively, in which it prompts for input and allows a selection of pictures chosen by commands from an alpha-numeric terminal to be converted; Batch, in which all the relevant parameters must be given on the command line and in which the whole metafile is converted; and Remote-Batch which is similar to the last case but in which the command is executed on a remote machine. Thus, whilst Interactive and Batch modes run locally, the Remote-Batch mode allows the input metafile to be converted into a form which may only be available on the remote host. For example, this feature allows GRCONV to subsume the old GKSSGML by producing a set of IBM-3812 bitmap files on CERNVM.

More details of how GRCONV operates may be found in reference [9], or by typing HELP GRCONV or FIND GRCONV.
Chapter 4: Making Hardcopies

It is possible to make hardcopy plots either interactively with GKS or via a metafile. Some devices accept acetate sheets, which allow transparencies to be produced for overhead projectors. The GKS plotter or metafile workstations operate just like any other, with an important exception. Namely, functions such as 'Delete Segment' will not undraw what has been output on the paper! Thus, when making a hardcopy, it is suggested that the picture to be plotted is produced first interactively on a terminal screen and stored in WISS (Workstation Independent Segment Storage). During this time the plotter or metafile workstation should be deactivated. When the picture is complete, and does not require further modification, then the plotter or metafile workstation may be activated and the WISS segments containing the picture sent to it by calling Associate SeGment to WorKstation (GASGWK).

4.1 Hardcopies via an Interactive Program

Plotter output may be produced by an interactive program running on a machine directly connected to a hardcopy device. For example, GKSGRAL at CERN supports Hewlett Packard pen plotters, VERSATEC electrostatic and thermal plotters, and any laser printer driven via the PostScript interface. (See the latest version of the include file GTSDEV for a complete up-to-date list of supported devices.)

In the case of VERSATEC or PostScript, an intermediate plot file is generally produced which must be queued to the device. However, use of an HP plotter normally assumes that the device is connected directly with the terminal line, and it is activated by special escape sequences sent by the driver, but it is also possible to produce an HP plotter file by specifying a connection identifier of conid = (100 + n), where n is a small integer greater than 1. This causes the output to be written to FORTRAN Logical Unit number n, and the file has the name PLxxxx.PLT, where 'xxxx' indicates the workstation type. Some laser printers do not support PostScript but do, in fact, support the Tektronix 4014 protocol. In this case it is possible to capture the 4014 escape codes on a file which is later sent to the printer. In any case, to capture the graphics on a file, first open a file on unit n, and then open the corresponding GKS workstation with a conid of (100 + n).

Note that use of a hardcopy device via an interactive program has the advantage that the application can inquire the specific device characteristics, and thus tailor the output accordingly.

4.2 Hardcopies via a Metafile

Using a metafile it is possible to make hardcopy plots on devices connected to the central computing facilities. The devices currently supported include: $==$ To be updated ...

- VERSATEC Model CE 3236E (colour with 36 inch roll paper, 400 dpi)
- VERSATEC Versacolor Model VE2700 (white and transparent A4 cut sheet)
- XEROX Model 4050 Laser Printer (A4 cut sheet)
- IBM 3812 Laser Printers (A4 cut sheet)
- PostScript printers, such as Apple Laser Writers, etc.

It is not necessary to be logged on to one of the computer centre machines to use these services. The command GRPLOT may be used to output metafiles on any of the central plotting devices from all centrally supported machines at CERN connected via either the DECNET or TCP/IP protocols, as the command first transfers the metafile over the network to the correct destination. The GRPLOT command
4.3. Mixed Text and Graphics

has replaced the previous collection of GKSVT, GKSCP, GKSX87, and GKS3812; rather than having a separate command for each plotter, the output device is provided as a parameter. The mechanism for naming output devices is the same as that for the latest version of XPRINT.

More details of how GRPLOT operates may be found in reference [9], or by typing HELP GRPLOT or FIND GRPLOT.

Note

Postscript and VERSACOLOR colour plotter produces output on A4 cut sheets. However, the paper and ink donor rolls are expensive. Thus, users are asked to use the device only for the final version of their plots, and not whilst debugging programs. Plots can not be released automatically, and users have to release them by hand using a terminal next to the device which is installed in the user area on the ground floor of building 513.

4.3 Mixed Text and Graphics

$==> To be updated ...

4.3.1 Via SGML/BookMaster

A version of the document markup language SGML layered on top of the IBM BookMaster product [12] is implemented on the central VM/CMS service and supports the inclusion of pictures into compound documents. The syntax of SGML/BookMaster is not identical to that of the original Waterloo/Script based SGML implementation available at CERN, but the procedure is similar. Namely,

1. Use a graphics application program, including calls to GCNAME, to produce a metafile.
2. Run GRCONV to produce the set of picture files.
3. Mark-up a document with SGML/BookMaster and associate the picture files with particular figures within the document.
4. Run SGML/BookMaster on the document.
5. Edit the document as necessary and repeat previous step until result is satisfactory.

Note that it is not necessary to repeat the first two steps every time SGML/BookMaster is used to re-process the document as long as the picture files are stored on the user’s mini-disk.

The GRCONV command can be used to generate the picture files, but as it is possible to print the resulting document on either IBM-3812 compatible or PostScript printers the user must take care to select the correct output format. As it would be cumbersome to require every picture to be stored in its own metafile and processed separately, GRCONV can handle metafiles which contain several pictures (separated by ‘CLEAR WORKSTATION’ items), and be instructed to produce a separate output file for each picture on the metafile. If the user is not working under VM/CMS GRCONV first transfers the metafile to the user’s VM account, and then interprets it to produce the required set of output files. Note that GRCONV replaces the command GKSSGML, and has a wider range of features.

A complete job to print a PostScript picture is given below. PostScript pictures can be scaled at will, but if a BoundingBox command is present in the Encapsulated PostScript file, the WIDTH parameter of the

1. Care must be taken to ensure that there is no conflict in having GRCONV write to a VM/CMS mini-disk which is accessed in write mode by another process. The user should have a write password on a mini-disk available for access by GRCONV in write-mode.
**ARTWORK** tag is redundant, and the size specified by the BoundingBox or DEPTH inside the file will be used. In order to avoid conflicts in the X and Y scaling, users are advised not to specify both DEPTH and WIDTH.

```xml
<!DOCTYPE USERDOC SYSTEM "USERDOC DTD *" [
<!-- Declare the Encapsulated PostScript file -->
<!ENTITY FIGURE1 SYSTEM "MYFIG1$S EPS" CDATA EPS> ]>
<Userdoc>
<Prolog>
<Document>
<Body>
... some text
<Artwork Name=FIGURE1 DEPTH=17CM ALIGN=CENTER>
```

The procedure for output onto an IBM-3812 compatible printer (IBM-3812 or IBM-3816) is similar, but in this case the graphics image is stored not in an Encapsulated PostScript file but in a PSEG3820 image file which cannot be scaled.

```xml
<!DOCTYPE USERDOC SYSTEM "USERDOC DTD *" [
<!-- Declare the PSEG file -->
<!ENTITY FIGURE1 SYSTEM "MYFIG1$S PSEG3820 *" NDATA PSEG> ]>
<Userdoc>
<Prolog>
<Document>
<Body>
```
Chapter 5: Title text

<ARTWORK NAME=FIGURE1>

Note that the figure name FIGURE1 specified by the NAME attribute of the ARTWORK tag is defined at the beginning of the file via the ENTITY declaration. Thus, the actual file name on the user’s disk and the name of the figure in the text are decoupled; one can change the figure used by changing the ENTITY declaration and without modifying the body of the text. There should be one picture file and one ENTITY declaration for each figure.

SGML/BookMaster does not place restrictions on the file names used, but the original CERN implementation of SGML did. Thus, for backwards compatibility, GRCONV still generates file names conforming to the old convention: the file name must contain eight (8) characters terminating in an ’S’. As described in section on Page 38 if the name supplied by the application which calls GCNAME provides less than 7 characters, then GCNAME pads the name out with ’$’s. Hence ’MYFIG1’ becomes ’MYFIG1$S’. If GCNAME was not used by the application to write names for each figure onto the metafile then GRCONV will generate names for each picture file automatically. The algorithm used is described in [9].

5.0.1 Via Waterloo/Script SGML

The original version of the document markup language SGML [?] implemented on the central VM/CMS service supports the inclusion of pictures. However, note that this SGML implementation is being replaced by a new one layered on top of the IBM BookMaster product (see above).

For each picture to be included when using the IBM-3812 output devices SGML requires two files; one to define the amount of space to be reserved within the document, and one containing the actual bit-map. These files must be accessible to SGML on one of the users mini-disks. As described in [?], the files may originate from several sources. This section will discuss how to use the GRCONV command to produce them from a GKS metafile and store them on the specified mini-disk.

The sequence of operations would be:

1. Use a graphics application program, including calls to GCNAME, to produce a metafile.
2. Run GRCONV to produce the set of picture files and store them on a user disk.
3. Mark-up a document with SGML and associate the picture files with particular figures within the document.
4. Run SGML on the document.
5. Edit the document as necessary and repeat previous step until result is satisfactory.

Note that it is not necessary to repeat the first two steps every time SGML is used to re-process the document so long as the picture files are stored on the user’s disk.

An example of the SGML syntax necessary to include a figure corresponding to a pair of picture files would be:

<FIG>
<PICTURE NAME=XXXXXXXXS>
<FIGCAP>This is the Figure Caption</FIGCAP>
</FIG>
When SGML encounters the tag ’PICTURE’ it reads a file specified by the ’NAME’ attribute. The name must consist of 8 characters ending with an ’S’. The file type under VM/CMS is ’SGML’. (The name of the corresponding bit-map file has the same first 7 characters, but the eighth is ’P’.) The GRCONV command can be used to generate these two files. In fact, as it would be cumbersome to require every picture to be stored in its own metafile and processed separately, GRCONV can handle metafiles which contain several pictures (separated by ’CLEAR WORKSTATION’ items). It then produces one pair of output files for each picture on the metafile. If the user is not working under VM/CMS, GRCONV first transfers the metafile to the user’s VM account, and then interprets it to produce the required set of output files. (Care must be taken to ensure that there is no conflict in having GRCONV write to a VM/CMS mini-disk which is accessed in write mode by another process. The user should have a write password on a mini-disk available for access by GRCONV in write-mode.) Note that GRCONV replaces the command GKSSGML, and has a wider range of features.

5.0.2 Via L\LaTeX

As the mechanism used to incorporate pictures into TeX documents depends not on TeX itself, but on the printer driver used by TeX, no single mechanism exists to achieve this. Thus, unfortunately, the procedure varies and is not described here. For details, see [13].

Assuming the hardcopy will be onto a device which supports PostScript or an IBM-3812, then GRCONV should be used to produce an Encapsulated PostScript or PSEG38PP/PSEG3820 file as for the SGML/BookMaster case above.

5.0.3 Naming Pictures

In order to include pictures within a document clearly it is necessary to have some way in which to designate which picture goes where. Thus, the pictures require naming in some way. To accomplish this, a routine called GCNAME has been added to the GKSGRAL and GKSGRAL-3d libraries. (It is available also for DECGKS, and in source format in GKSPACK.)

GCNAME outputs a user item containing a name string onto the metafile and, for example, this is used by GRCONV to generate the file names, and by GRVIEW to allow the user to choose by name the picture to be edited. GCNAME should be called to name a picture immediately after the previous picture delimiter (Clear Workstation) and before any primitives or attributes for the named frame have been written out. The call is:

\texttt{CALL GCNAME(WKID, NAME)}

where WKID is the metafile workstation identifier, and NAME is a FORTRAN variable or constant of type CHARACTER and with a length of seven (7) characters. The seven characters must correspond to the ’XXXXXXXX’ preceding the ’S’ used in the picture NAME tag used by SGML. If less than 7 characters are used then GRCONV will pad the name out with dollar signs (’$’). Thus, if a picture was named ’FPROJ’ by GCNAME, then the corresponding SGML NAME tag would need to be be ’FPROJ$$S’. This convention is maintained for backwards compatibility, although it is not required by SGML/BookMaster.

Even though not required for other operating systems, in order to remain compatible with the file naming scheme used by VM/CMS, GCNAME restricts the character set allowed for GCNAME to upper-case
5.1. Use of Workstation Transformations to Change Picture Size

alphabetic, the digits 0-9, and the dollar sign (\$). Lower-case characters are automatically converted to upper-case. This is also why the number of characters is limited.

In order to avoid getting spurious or empty picture files produced by GRCONV, application code should issue the Clear Workstation call to the metafile workstation only if a complete picture really has been written to the file.

5.0.4 Encapsulated PostScript

It is possible to use the GKS PostScript driver in order to produce output files in the format Encapsulated Postscript, which is used by various text-processors (TeX, SGML, BookMaster, etc.) in order to combine PostScript-encoded data originating from different sources. Encapsulated Postscript format is similar to standard PostScript but with some additional header information concerning the picture size. The PostScript instruction "showpage" which is normally included at the end of the file in order to tell the interpreter to print the preceding information is removed in the Encapsulated format. This avoids any conflict with the contents of other PostScript files into which the Encapsulated file may be embedded. Thus, sending an Encapsulated PostScript file to a printer directly will not produce any output.

It is possible to use GRCONV to convert a GKS metafile to Encapsulated PostScript format by specifying the requisite qualifier ('EPSPM', ...). If the input metafile contains several pictures it is also possible to use the 'SPLIT' option in order to get one output file per picture.

If a PostScript file is to be produced directly by a GKS application program then the following applies:

1. Open a GKS PostScript Workstation. The Workstation Type is the same for both normal and Encapsulated PostScript formats.
2. As for all drivers, if the connection identifier is less than 100 (see on Page 62) then the PostScript instructions are sent directly to the terminal. (Useful if the terminal supports Display PostScript).
3. If the connection identifier is greater than 100 but less than 200 (conid = 100 + n), then PostScript instructions will be written to a file in normal PostScript format opened on unit number ‘n’. The file may include several pictures.
4. If the connection identifier is greater than 200 (conid = 200 + n), then PostScript instructions will be written to a file in Encapsulated PostScript format opened on unit number ‘n’. Only one picture should be included per file.

5.1 Use of Workstation Transformations to Change Picture Size

When using GKS from an application program it is not necessary to specify a workstation transformation to set the size of the output picture. In this case, the Normalization Viewport will automatically be made to map to the largest possible square which fits onto the output device display surface. This is often acceptable for an interactive graphics session. However, when making a hardcopy image, especially if this must fit within a document, then the actual size of the picture may well be very important. The GKS Workstation Transformation, which is described fully in on Page 6, allows the actual area required to be specified in metres. A brief summary will be given here.

The Workstation Transformation defines which part of the Normalized Device Coordinate (NDC) space will be visible, and where it will appear on the display surface. It is set by calling the functions Set WorKstation WiNdow and Set WorKstation ViewPort:
CALL GSWKWN(WKID, WXMIN, WXMAX, WYMIN, WYMAX)
CALL GSWKVP(WKID, VXMIN, VXMAX, VYMIN, VYMAX)

Set Workstation Window specifies in NDC coordinates the area of the NDC space to be output to the device. Set Workstation Viewport specifies in Display Coordinates (DC) where on the device the window will appear. The aspect ratio for the workstation window and the workstation viewport must always be the same. If they are not, then the specified transformation is ignored, and the complete workstation window is displayed on the device in the correct aspect ratio.

As an example, suppose that the picture occupies the area (wxmin, wymin) to (wxmax, wymax) in World Coordinates. This may be mapped via the Normalization Transformation to a Viewport (vxmin, vymin) to (vxmax, vymax), and this does not have to preserve the aspect ratio. This Viewport must now be output onto an A4 sheet so as to fill the width of the page, assuming an aspect ratio with width $\frac{w}{h}$.

**Note**

Although the width of an A4 page is 21 cm, most output devices do not permit the full width to be used. Consult the Workstation Description Table for the device to find out what is the maximum size.

C Set the Window and Viewport for Normalization Tfrm 'trn'
C and select this Normalization Tfrm for future primitives.
C
CALL gswn (trn, wxmin, wxmax, wymin, wymax)
CALL gsvp (trn, vxmin, vxmax, vymin, vymax)
CALL gseltnt(trn)
C
C Set the Workstation Window and Workstation Viewport
C Note that 18.9 cm is the maximum width for PostScript Portrait Mode
C (The Workstation Window to map from the Normalization Viewport)
C
CALL gswkwn(wkid, vxmin, vxmax, vymin, vymax)
CALL gswkvp(wkid, 0.0, 0.189,
* 0.0, 0.189*(vymax-vymin)/(vxmax-vxmin))
Chapter 6: GKS-3D Primer

Note

Whilst in general the information in this Primer is independent of a particular GKS implementation, this is not entirely the case for GKS-3D. The reason is that the goal of the Primer is not simply to describe the functionality of GKS, but to explain in detail how it may be used via the FORTRAN language binding. However, as the binding has not yet been completely finalised for GKS-3D, this chapter explains its use in terms of a particular implementation, namely GKSGRAL-3D Version 2.0.

The text of the Draft International Standard for the GKS-3D FORTRAN binding was published at the end of 1988, with voting to finish by May, 1989. Discrepancies between the FORTRAN DIS and GKSGRAL-3D Version 2.0 are minor, and are mentioned at the appropriate places within the chapter. Only the setting of the viewing parameters and the aspect source flags are affected, and the text will be revised after the final voting on the standard is complete. The changes between VSN 1.1 and VSN 2.0 of GKSGRAL-3D are described in Appendix on Page ??.

6.1 Introduction to GKS-3D

GKS-3D is a pure super-set of GKS designed to handle 3D graphics in a compatible way. That is to say, a 2D application written to the GKS standard is guaranteed to run in a GKS-3D environment without change. However, apart from the usual GKS functions, GKS-3D provides additional ones to handle 3D primitives, 3D input, and 3D viewing. As for GKS, the standard is in multiple parts with the language bindings separate from functional specification. The FORTRAN binding is defined in [7].

All primitives within GKS-3D are deemed to be three dimensional. Thus, although an application using GKS-3D may make only 2D function calls, all the 2D primitives will be turned immediately into the corresponding 3D versions inside the package by the addition of a Z coordinate. This has several noticeable effects:

- 2D function calls may be slightly slower than 3D ones (using the same GKS-3D package), as a 2D call will add a Z coordinate and then make the corresponding 3D function call.
- 2D applications running on GKS-3D will take more storage space in WDSS and WISS.
- A 2D picture stored on a 3D metafile will require more space than on a 2D metafile. Also, quite clearly, such a picture could not be re-interpreted by reading the metafile into a GKS (2D) implementation.

6.2 The Drawing Primitives

With one addition, GKS-3D supports the same GKS primitive types as described in chapter on Page ??.

Four of the functions are called in a similar way to that for GKS-2D, but with the addition of a third coordinate:

Polyline 3D CALL GPL3(N, PXA, PYA, PZA)
Polymarker 3D CALL GPM3(N, PXA, PYA, PZA)
Fill Area 3D CALL GFA3(N, PXA, PYA, PZA)
GDP 3D CALL GGDP3(N, PXA, PYA, PZA, PRIMID, IDR, DATREC)
However, the 3D routines for Cell Array (GCA3) and Text (GTX3) have calling sequences which are substantially changed. GCA3 is not described here, and those brave enough to peruse the standards document will see why. Concerning GTX3, it should be emphasized that this function is only required if it is desired to place text on a surface which is not parallel to the X-Y plane, otherwise the 2D text function (GTX) is sufficient. For example, one would need to use GTX3 to write ‘MIGROS’ on the side of a truck drawn with some random orientation.

The function call for 3D Text is:

```
CALL GTX3(PX, PY, PZ, TDX, TDY, TDZ, STR)
```

where the arguments are as follows:

- **PX**, **PY**, **PZ**: Reference Point (start point of text string)
- **TDX/Y/Z(2)**: Two Text Direction vectors (three real arrays of length 2).
- **STR**: The character string to be output.

For both GTX and GTX3 the character string, STR, is drawn in a plane. The difference is that in the 3D case this **text plane** can be oriented in space using two **text direction vectors**, U and V, specified by TDX(1), TDY(1), TDZ(1), and TDX(2), TDY(2), TDZ(2). The text is drawn in a plane perpendicular to the vector formed from taking the vector product U x V, and the origin of the local coordinate system in which the geometric text attributes are measured is defined by the reference point, P. The X axis of the local text coordinate system is parallel to the first direction vector, U, and the Y axis is perpendicular to this direction. Thus, the system reduces to the 2D case if U and V are defined as (1, 0, 0) and (0, 1, 0).

An example program using 3D Text is given in Appendix on Page 82.

The additional primitive, called **Fill Area Set** (or **Fill Area Set 3**), generates a set of polygonal areas which may be hollow or filled with a uniform colour, pattern, or hatch style. Thus, it allows the application to specify regions with holes, or disjoint regions which must be treated as a single entity. There are both 2D and 3D versions of Fill Area Set, and the 2D function will be added to the next revision of GKS. Unlike the Fill Area primitive, Fill Area Set and Fill Area Set 3 have separate attributes, described below, to control the edge style of the polygons. The calls are:

```
CALL GFAS(NPTR, PTR, PX, PY)
CALL GFAS3(NPTR, PTR, PX, PY, PZ)
```

where the arguments are as follows:

- **NPTR**: (integer) Number of fill areas in set
- **PTR(NPTR)**: (integer) Array of pointers to starting elements of lists in PX(N), PY(N), PZ(N). Total number of coordinates is PTR(NPTR)-1.
- **XA/YA/ZA(*)**: List of points for all fill areas of the set.

Thus, all the polygons are packed into the arrays, XA(N), YA(N), ZA(N), and the start point of each polygon is indicated by the pointers in the array PTR(NPTR).

**Note**

Although they are specified in a 3D space, the primitives Text, Cell Array, Fill Area, and Fill Area Set are all **coplanar**. It is the responsibility of the application program to ensure that the coordinates supplied fulfil this condition. What happens if they are not coplanar is implementation-dependent!
6.3 The Output Attributes

As for the primitives, GKS-3D uses almost the same output attributes as GKS-2D (described in Chapter on Page 11), but with the three following additions:

1. Extra attributes are required for the new Fill Area Set primitive.
2. The View Index attribute. This is analogous to the Normalization Transformation index attribute, and specifies which viewing transformation is to be used to process the primitive.
3. The Hidden Line/Hidden Surface (HLHSR) attribute specifies which HLHSR algorithm should be used to process the primitive (if HLHSR is supported by the workstation).

The Fill Area Set primitive uses the same attributes as Fill Area to control the interior of the polygons (see section on Page 13), plus the following which allow independent control of the edges:

- **EDFLAG**: the Fill Area Set edge flag (ASF 1). Set by GSEDGF(EDFLAG). EDFLAG may take the values 'GON' or 'GOFF'.
- **EDTYPE**: the Fill Area Set edge type (ASF 2). Set by GSEDT(EDTYPE). EDTYPE is an integer value which is workstation-dependent.
- **EDWSF**: the Fill Area Set edge width scale factor (ASF 3). Set by GSEWSC(EDWSF). The value of EDWSF is a real number which modifies the width of the line used to draw the edge of the Fill Area Set.
- **EDCI**: the Fill Area Set edge colour index (ASF 4). Set by GSEDCI(EDCI).

The numbers in brackets are, for the GTS-GRAL implementation, the indices into the integer array ASFLST(4) used to set the corresponding Attribute Source Flags by calling:

```call GASF3(ASFLST)```

However, the DIS FORTRAN binding uses GASF3 to set all the Attribute Source Flags in one go by defining the array ASFLST to be of length 17, where elements 1 to 13 correspond to the 2D case and the last four elements are those listed here.

Note that the Fill Area Set primitive is rendered using two independent sets of attributes, one for the interior, and one for the edge. This separate set of edge attributes also has its own attribute bundle, selected by calling GSEDI, plus a corresponding routine to Set the EDge Representation:

```call GSEDI(EDI)
and GSEDR(WKID, EDI, EDFLAG, EDTYPE, EDWSF, EDCI)```

6.4 Viewing in 3D

Setting up the Viewing Parameters is undoubtedly the most complicated part of any 3D graphics system. When primitives are output to a workstation they (conceptually) pass through a series of processes called the Viewing Pipeline before they finally reach the display surface. This pipeline is briefly described below in order that the reader is aware of the complete process (see 6.1). The transformations will then be covered in more detail.
The primitives are transformed by the *Normalization Transformation* from World Coordinates (WC3) to Normalized Device Coordinates (NDC3), which are always in the range [0.0, 1.0]. This transformation is composed of a translation and change of scale, but no rotation. GKS-3D allows for the existence of many World Coordinates systems, and their corresponding Normalization Transformations are numbered from 0 upwards. Normalization Transformation 0 always corresponds to the identity matrix. Normalization in 3D is exactly analogous to the 2D case described in section on Page 13.

Primitives which are stored in segments are also processed by the *Segment Transformation* before proceeding to the next stage. In the 3D case this requires a 3 x 4 matrix which is described below. The segment transformation maps NDC3 to NDC3, and includes scaling, rotation, and translation.

Having assembled the components in a unique NDC3 space, primitives may next be clipped to a box to remove extraneous information. This is called the *Normalization Clip*, and may be switched on or off using the *Normalization Clip Flag*.

The primitives are now "viewed" from some arbitrary direction. The *View Orientation Transformation* performs a rotation only to take Normalized Device Coordinates to View Reference Coordinates (VRC3). The application is free to calculate the corresponding matrix itself, or to use a utility routine which is described below.

The *View Mapping (Projection) Transformation* next takes View Reference Coordinates to Normalized Projection Coordinates (NPC3) in order to provide parallel or perspective projection of the image. As for the View Orientation Transformation, the application is free to calculate the required matrix using its own algorithm, or to call a utility function.

At this point the *View Clip* takes place. It is positioned at this stage in the pipeline so that the clip box may be defined as a rectangular parallelepiped with its sides parallel to the axes of the NPC3 system, and thus the clipping algorithm is more efficient. The View Clip is controlled by three *Clip Flags* which allow clipping to be turned on or off separately for the front plane, back plane, and window.

Finally, the *Workstation Transformation* takes Normalized Projection Coordinates to Display Coordinates (DC3) in order to position the projected image in the device coordinate space. It preserves the aspect ratio, and includes a clipping operation which cannot be disabled. As their clip faces are parallel, the View Clip and Workstation Clip are usually combined internally for efficiency. DC3 coordinates may be in metres or raster units. The Workstation Window limits are [0,1]x[0,1]x[0,1].

A good implementation of the graphics pipeline will attempt to combine as many as possible of the stages in the pipeline using matrix concatenation in order to reduce the amount of computation necessary.

### 6.4.1 The Normalization Transformation

As in the 2D case, this is specified by a Window volume in World Coordinates, and a Viewport volume in Normalized Device Coordinates. The Normalization Clip is controlled as for GKS-2D (see section on Page 21). The calls are:

```
CALL GSW3(TNR, WN)
CALL GSV3(TNR, VP)
```
6.4. Viewing in 3D

where WN and VP are real arrays of dimension 6 containing (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX).

6.4.2 The View Orientation Transformation

The View Orientation Transformation algorithm provided by a utility function in the GKS-3D standard performs a rotation in three dimensions in order to take Normalized Device Coordinates to View Reference Coordinates (VRC3), where the axes are labeled U, V and N (see 6.2). The definition of the UVN system requires the application to specify:

1. The View Reference Point (VRP), which is a point on or near the object to be viewed, and is the origin of the VRC3 system.
2. The View Plane Normal (VPN), which points from the View Reference Point towards the eye point. The VPN is the third axis of the VRC3 system, and the plane perpendicular to the View Plane Normal through the View Reference Point is called the View Reference Plane.
3. The View Up Vector (VUV), which defines the direction to be considered as ‘UP’ within the View Plane. It is the second axis of the VRC3 system. Clearly, the View Plane Normal and the View Up Vector must not be defined to be collinear.

Thus, in the View Reference system, the axis N is along the View Plane Normal, the axis V is the projection of the View Up Vector onto the View Reference Plane, and the axis U is constructed to form the third axis of a right-handed coordinate system. The default transformation sets the identity matrix. The limits of the VRC3 system are [0,1]x[0,1]x[0,1].

Figure 6.2: The GKS-3D View Reference System. The View Reference Point, defined in World Coordinates, should be situated near the object to be viewed. The View Plane Normal is directed at the eye point.

6.4.3 The View Mapping (Projection) Transformation

The View Mapping (Projection) Transformation provided by a utility function in the GKS-3D standard takes View Reference Coordinates to Normalized Projection Coordinates (NPC3), and the projection may be either parallel or perspective (see 6.3). The default View Mapping Transformation sets the identity matrix.

The transformation maps the View Volume in VRC3 space to a rectangular parallelepiped in Normalized Projection Coordinates. The horizontal and vertical boundaries of the View Volume are specified by the projectors drawn from the Projection Reference Point (the centre of projection) to the corners of the View Window, which is a rectangle parallel to the View Reference Plane, and with edges parallel to the U and V axes. The View Window cuts the N axis of the VRC3 system at the View Plane Distance (VPD) from the View Reference Point. The hither (front) and yon (back) boundaries of the View volume are specified by the Front Plane and the Back Plane, which are planes parallel to the View Reference Plane at distances from the View Reference Point called the Front Plane Distance (FPD) and the Back Plane Distance (BPD).

The rectangular parallelepiped in NPC3 to which the View Volume is mapped, called the Projection Viewport Limits, is specified by the two points (XMIN, YMIN, ZMIN) and (XMAX, YMAX, ZMAX).
Chapter 6. GKS-3D Primer

Although the View Mapping operation maps a volume to a volume, which is slightly different from the usual idea of projection onto a plane, a little thought should (hopefully) convince the reader that the effect is equivalent. Consider looking at the result along the N axis; the effect is that the contents of a truncated pyramid are distorted to map into a rectangular volume, and thus objects closer to the viewer will be magnified.

For parallel projection, the projectors are parallel to a line drawn from the Projection Reference Point to the centre of the View Window, and thus the PRP should be set in the middle of the View Window to produce an orthographic projection. In this case the actual Z value of the PRP is unimportant, so long as it is not identical to the View Plane Distance.

The View Clip takes place at the edges of the View Clipping Limits according to the setting of the clipping indicators. There are three of these: the x-y clipping indicator, the front clipping indicator, and the back clipping indicator. The default clipping limits are set to [0,1]x[0,1]x[0,1] and all clipping indicators set to clip on ("GCLIP"). It would normally be the case that the View Clipping Limits and the Projection Viewport Limits were set to the same values, but this is not a requirement. If not set to be co-incident, there is clearly the possibility for an application to clip away all of the image by mistake!

Figure 6.3: The GKS-3D Projection System. The figure shows the definition of the View Volume with Perspective Projection.

6.4.4 Setting the 3D View Representation

Having described the concepts, this section will attempt to explain how all these parameters are actually set. This is an area of the GKS-3D standard which was modified fairly late, and release 2.0 of GKS/GRAL-3D still took place before the publication of the Draft International Standard for the FORTRAN binding [7].

As for the Normalization Transformation, there may be more than one Viewing Transformation (actually, combined Viewing and Projection Transformation plus Clip), and the different transformations are specified by a View Index. However, unlike the Normalization Transformation, which applies to all primitives no matter on which workstation they are displayed, the Viewing Transformation is workstation-dependent, and so the same View Index (VWI) may produce a different effect on each active display. The View Index is set using the routine:

CALL GSVWI(VWI)

Following this call all primitives will be transformed according to the parameters specified by viewing attribute bundle VWI, assuming that the deferral mode set has allowed the workstation to be brought up-to-date. The default viewing attributes, corresponding to VWI = 0, define identity matrices for the View Orientation and View Mapping transformations, and place the clip limits at the boundary of NPC3 space.

As indicated, the attribute values contained in the viewing attribute bundle specified by VWI must be defined separately for each workstation using the call:

CALL GSVWR(WKID, VWI, VWM, PRM, VCLP, CLW, CLB, CLF)
6.4. Viewing in 3D

WKID Workstation Identifier
VWI The View Index
VWM The View Orientation Matrix (a 4 x 4 real array), which may be calculated by the function GEVVWM
PRM The Projection (View Mapping) Matrix (a 4 x 4 real array), which may be calculated by the function GEVPJM
VCLP The View Clipping Limits (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
CLW Clip Indicator for Window Clipping (‘GNCLIP’, ‘GCLIP’)
CLB Clip Indicator for Back Plane Clipping (‘GNCLIP’, ‘GCLIP’)
CLF Clip Indicator for Front Plane Clipping (‘GNCLIP’, ‘GCLIP’)

The utility functions provided to evaluate the matrices are EVAluate VieW orientation Matrix and EVAluate ProJection (View Mapping) Matrix:

CALL GEVVWM(VRPX, VRPY, VRPZ, VUPX, VUPY, VUPZ, 
VPNX, VPNY, VPNZ, CSW, ERR, VWM)

VRPX/Y/ZThe View Reference Point in NDC3 or WC3
VUPX/Y/ZThe View Up Vector in NDC3 or WC3
VPNX/Y/ZThe View Plane Normal in NDC3 or WC3
CSW Switch specifying whether the vectors are given in World Coordinates or Normalized Device Coordinates (‘GWC’ or ‘GNDC’)
ERR (out) Error indicator
VWM (out) View Matrix (a 4 x 4 real array)

CALL GEVPJM(UMIN, UMAX, VMIN, VMAX, PRVP, PROTYP, 
PRUP, PRPV, PRPN, VPD, BPD, FPD, ERR, PRM)

U, V (MIN,MAX) The limits of the View Window on the View Reference Plane measured relative to the View Reference Point. UMIN, VMIN is the bottom left corner, UMAX, VMAX is the top right corner.

PRVP The Projection Viewport Limits (a 6 element real array containing XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
PROTYP The Projection Type flag integer (‘GPARL’ or ‘GPERS’)
PRUP/V/N The Projection Reference Point.

VDP The View Plane Distance from the View Reference Point along the N axis. (Note that the View Reference Point does not have to be contained in the View Plane).

BPD, FPD The Back and Front Plane Distances from the View Reference Point along the N axis. FPD must be greater than BPD.
ERR (out) Error indicator
PRM (out) Projection Matrix (a 4 x 4 real array)
6.4.5 Workstation Transformation

This is specified by a Workstation Window volume in Normalized Projection Coordinates, and a Workstation Viewport volume in Device Coordinates. The Workstation Transformation always preserves the aspect ratio, and the default transformation will be used if the window or viewport volumes are not correctly set.

CALL GSWKW3(WKID, WKWN)
CALL GSWKV3(WKID, WKVP)

where WKWN and WKVP are real arrays of dimension 6 containing (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)

To inquire the range of device coordinates corresponding to a particular workstation type, it is possible to call the function GQDVOL (inQuire Display VOLume).

CALL GQDVOL (WTYPE, ERRIND, DCUNIT, RX, RY, RZ, LX, LY, LZ)

The routine returns the units in which the display surface is measured (DCUNIT), and also the maximum x, y, and z values in each direction. Some devices, for example hardcopy plotters, are measured in metres (DCUNIT='GMETRE'), so the routine will return the actual size of the device. Other devices, for example graphics terminals, will be defined in raster or some other units (DCUNIT='GOTHU'). RX, RY, RZ are real device coordinates, and LX, LY, LZ are in integer raster units. The use of GQDVOL is analogous to the use of GQDSP, which is described in section on Page 19.

6.4.6 But I don’t see anything!

The first (and second and maybe third) time one tries out a 3D application, even after having checked the code very carefully, there may be nothing on the screen. The following is a list of possible pitfalls:

1. Is the View Reference Point really on or near the object to be viewed?
2. Is it defined in WC or NDC, and does this match the coordinate switch?
3. Is the View Plane Normal pointing the correct way? A classical bug is to look 180 degrees in the wrong direction. GKS-3D uses a Right-Handed system, so if the object to be viewed is at the origin, and is viewed along the Z axis, then one looks in the negative Z direction, but the VPN points in the positive Z direction (0.0, 0.0, 1.0).

Note

As only the direction and not the length of View Plane Normal vector matters, one can simply use the coordinates of the eye point to specify this vector, so long as the View Reference Point is placed at the origin.

4. Assuming one wants an orthographic projection, is the Projection Reference Point in the middle of the View Window?
5. Has the View Window (UMIN, VMIN), (UMAX, VMAX) really been set around the object?
6. Have the Front Plane and Back Plane distances been set correctly in front and behind the object?
7. Is the aspect ratio of the Normalization Viewport volume the same as that of View volume?
Are the Clipping Limits which define the projection parallelepiped correctly set within the NPC system limits? Do the Clipping Limits and Projection Viewport Limits match? **It is always safer to start with clipping switched off!**

Have the Workstation Window and Workstation Viewport been set to the same aspect ratio?

### 6.5 Segments

Segmentation operates in GKS-3D in the same way as for GKS, described in section on Page 24, except that the segment transformation matrix is defined to be 3 x 4. (3 x 3 for scaling and rotation plus 3 x 1 for translation). Thus, the 2D utility routines which calculate a segment transformation matrix, GEVTM and GACTM, are modified as follows:

```fortran
CALL GEVTM3(X0, Y0, Z0, DX, DY, DZ,
             ROTX, ROTY, ROTZ, FX, FY, FZ, SW, MXOUT)
CALL GACTM3(MXIN, X0, Y0, Z0, DX, DY, DZ,
             ROTX, ROTY, ROTZ, FX, FY, FZ, SW, MXOUT)
```

Similarly to the 2D case, GEVTM3 evaluates a matrix (MXOUT), whilst GACTM3 accumulates changes to an existing matrix (MXIN). Both routines require the definition of:

- **X0, Y0, Z0** (real) a fixed reference point about which 3D rotations take place.
- **DX, DY, DZ** (real) a translation (or shift) vector.
- **ROTX, Y, Z** (real) angles of rotation about the X, Y, and Z axes.
- **FX, FY, FZ** (real) X, Y, and Z scale factors.
- **SW** (enumerated) a switch specifying whether the reference point and shift vector are given in World Coordinates or Normalized Device Coordinates (‘GWC’ or ‘GNDC’).
- **MXOUT** (real) 3 x 4 output matrix composed in the order: scale, rotate, shift. In the case of GACTM, the matrix MXIN is pre-concatenated with that formed from the scale, rotate, and shift parameters, so MXOUT = SHIFT * ROTATE * SCALE * MXIN.

Once the transformation matrix has been evaluated, it may then be set in the segment by calling the routine:

```fortran
CALL GSSGT3(SGNA, MTX)
```

To **INsert a SeGment into the output stream** GNSG becomes:

```fortran
CALL GINSG3(SGNA, MTX)
```

Because GKS-3D is upwards compatible to GKS, one can still use the 2D versions of these routines. In this case, 2 x 3 matrices will be automatically filled-out to 3 x 4 by the suitable additions of 0s and 1s.

**Note**

GKS segment transformations are provided in order to orient the contents of segments with respect to the coordinate system in which their primitives were originally specified. In most cases it is extremely inefficient to modify the transformations in each segment in order to view a scene from a different direction. The viewing transformation should be used for this purpose.
6.6 Graphical Input

As only the Locator and Stroke logical devices return coordinate values, only these two have 3D versions of the routines used for the three input modes. Thus, for request mode, the calls are:

\[
\text{CALL GRQLC3}(\text{WKID}, \text{LCDNR}, \text{STAT}, \text{TNR}, \text{VWI}, \text{PX}, \text{PY}, \text{PZ})
\]

and

\[
\text{CALL GRQSK3}(\text{WKID}, \text{SKDNR}, \text{STAT}, \text{N}, \text{TNR}, \text{VWI}, \text{NP}, \text{PX}, \text{PY}, \text{PZ})
\]

Where, apart from the extra dimension, PZ, the only difference to the 2D calls is the addition of the View Index, VWI. (Note that as for TNR, VWI is an output parameter.) This corresponds to the index of the viewing attribute bundle used to convert the position(s) back from NPC3 coordinates to NDC3 coordinates. Of course, when using a physical 2D device, it is awkward for the user to provide the third coordinate for 3D input.

Although only Locator and Stroke have 3D functions to obtain input, all six logical devices have 3D versions of the initialization functions. For example, these allow the echo areas to be positioned in space, rather than in a plane. They will not be discussed further in this Primer.

6.7 GKS-3D Metafile

As for GKS, the GKS-3D standard has an Appendix E metafile. The logical format of the 2D and 3D Appendix E metafiles are the same. However, the contents are incompatible, as in one case points are represented by two values, and in the other by three values. The two types of metafile are clearly distinguishable by inspecting the metafile header record. In all other respects, the control and usage of the 2D and 3D metafiles are the same.

The Computer Graphics Metafile, CGM, will not initially have a 3D version. Therefore, if a CGM metafile driver is added to GKS-3D, the output could contain only a 2D projection. This would be sufficient to make a hardcopy, but only with the viewing parameters chosen when the metafile was created.

Note

The GKS-3D Appendix E metafile has never been implemented by GTSGRAL. Nevertheless, an additional output workstation is provided to produce a 2D metafile by carrying out the viewing and projection operations before outputting data to the file. This feature is useful for making hardcopies.
Chapter 7: General Hints for GKS Use

7.1 System and Implementation Dependencies

As mentioned elsewhere in this document certain features of GKS and GKS-3D are system or implementation dependent. System dependencies are described in the manual appropriate for the implementation in use.

The principle implementation dependencies to watch for are text fonts and fill area hatch styles but it is also possible to have difficulties by inadvertently trying to exceed some maximum value, such as a table length. GKS has many inquiry function which may be used to discover the current and/or maximum value of various parameters, and it is not the intention of the Primer to describe all of them. However, one should take note of the following routines:

\[
\begin{align*}
\text{CALL GQWKM}(&\text{ERRIND}, \text{MXOPWK}, \text{MXACWK}, \text{MXWKAS}) \\
\text{CALL GQMNTN}(&\text{ERRIND}, \text{MAXTNR}) \\
\text{CALL GQLWK}( &\text{WTYPE}, \text{ERRIND}, \text{MPLBTE}, \text{MPMBTE}, \text{MTXBTE}, \text{MFABTE}, \text{MPAI}, \text{MCOLI}) \\
\text{or} \\
\text{CALL GQLWK3}( &\text{WTYPE}, \text{ERRIND}, \text{MPLBTE}, \text{MPMBTE}, \text{MTXBTE}, \text{MFABTE}, \text{MPAI}, \\
&\text{MEDBTE}, \text{MCOLI}, \text{MVTE}) \\
\text{CALL GQSGF}( &\text{WTYPE}, \text{ERRIND}, \text{NSGP})
\end{align*}
\]

where the parameters are as follows:

\[
\begin{align*}
\text{WTYPE} & \text{ workstation type (input parameter)} \\
\text{ERRIND} & \text{ error number} \\
\text{MXOPWK} & \text{ maximum number of simultaneously open workstations} \\
\text{MXACWK} & \text{ maximum number of simultaneously active workstations} \\
\text{MXWKAS} & \text{ maximum number of workstations associated with a segment} \\
\text{MAXTNR} & \text{ maximum normalization transformation number} \\
\text{MPLBTE} & \text{ maximum number of polyline bundle table entries} \\
\text{MPMBTE} & \text{ maximum number of polymarker bundle table entries} \\
\text{MTXBTE} & \text{ maximum number of text bundle table entries} \\
\text{MFABTE} & \text{ maximum number of fill area bundle table entries} \\
\text{MPAI} & \text{ maximum number of pattern indices} \\
\text{MEDBTE} & \text{ maximum number of edge bundle table entries} \\
\text{MCOLI} & \text{ maximum number of colour indices} \\
\text{MVTE} & \text{ maximum number of view table entries} \\
\text{NSGP} & \text{ maximum number of segment priorities}
\end{align*}
\]

There is unfortunately no function provided to inquire the maximum available number of segments or the maximum available segment name so one must consult the relevant documentation.
7.2 Integrating separately written modules of code.

As is the case when any independent software modules are combined into a single program, care must be taken that on entry a module saves the state of the current environment and sets up its own defaults. The original environment must then be restored on exiting from the module. This applies to saving and restoring registers when making a subroutine call no more than saving and restoring (parts of) the GKS State List and Workstation State Lists when entering a graphics module. For example, two modules of graphics code may use the same Normalization Transformation indices. If module B sets different windows and viewports than module A, then on re-using routines in A after calling module B the transformations will no longer produce the expected results.

GKS provides a mechanism to handle this situation in the form of a large set of inquiry functions. These enable a module of code to inquire at run time the values of those parameters it intends to modify in order that they may be correctly restored afterwards. In particular, functions exist to inquire attribute values and bundle table entries, values of the aspect source flags, and the normalization and viewing (for GKS-3D) transformations corresponding to a particular normalization or view index. As an example:

```fortran
REAL WINDOW(4)
REAL VIEWPT(4)

C Inquire current window and viewport for Transformation Number X
CALL GQNT(TNX, ERRIND, WINDOW, VIEWPT)
CALL GSWN(TNX, WXMINA, WXMAXA, WYMINA, WYMAXA)

C Restore window which was modified for Transformation Number X
CALL GSWN(TNX, WINDOW(1), WINDOW(2), WINDOW(3), WINDOW(4))
```

If several modules of code which are to be combined use GKS segments then clearly they must not both attempt to use the same segment names. Either one can agree before hand on the allocation of a range of names to each module, or else code can be written which assigns segment names at run time after checking that the values are not in use (see section on Page 22).

7.2.1 Using GKS-3D libraries for GKS (2D) applications

As GKS-3D is a super-set of GKS it is possible to run a GKS 2D application linked to a GKS-3D library. However, if a 2D code module is incorporated into a larger 3D program then the programmer must be aware that primitives will be bound to the current View Index as well as the current Normalization Transformation Index, possibly entailing 3D transformations plus view clipping. Thus, to be sure that the code within the 2D module will behave as if linked to a 2D library the View Index should be set to 0 (identity matrix) in the 3D code before calling the 2D module. This can not be done in the 2D module itself, as otherwise the linker would give an error when linking to GKS which does include the 3D function GSVWI.

7.3 Plotting numbers as text strings

For FORTRAN programmers it is possible to use the Internal Write construct to convert numeric variables to character strings for output as text primitives or via GMSG:
7.4. **GKSPACK**

GKSPACK contains routines which may be placed into several categories.

- One category of routines is intended to ease the use of GKS for those people with simple applications who do not need further GKS features, NAG graphics users for example. These routines are built on top of GKS and are not implementation dependant. They include routines to initialize and stop GKS (GCINIT, GCSTOP), to ask the user for input (GPRMPT), etc.

- The next category of routines provide facilities to allow applications which must run on several GKS implementations to obtain information about workstation types and connection identifiers from a data file. This avoids having to ask the user interactively to supply implementation dependant values. These routines include GCGIMP, GCGWTC, etc.

- Some routines provide access to GKS features which might be awkward to use, or which may be implementation dependant. For example, use of GDPs, or the construction of menus. GKS-GRAL includes a large number of utility routines of this type starting with the sentinel characters GU, and GKSPACK contains CERN-written emulations of many of them for use with other GKS implementations.

- Another category of routines is for non-standard GKS utilities which allow the user to perform specific actions not foreseen by the GKS standard, and which may require internal knowledge of the GKS implementation. Thus, it may not be possible to implement these routines for GKS implementations other than the one from GTS-GRAL if access to the source code is not available. The principal examples of routines in this category, heavily used at CERN by PAW for instance, are GCATOG and GCGTOA. These are used to switch the graphics terminal between graphics and alphanumeric mode in order for applications to intermix graphics and Fortran I/O on the same device.

For various technical reasons it has been decided to include entry points for all routines in GKSPACK in the GTS-GRAL libraries maintained by CERN. This means that users of GKSGRAL do not need to link to a separate library in order to have access to GKSPACK routines, which is a change to the previous situation when it was necessary to link applications also with the CERN library `NGRAFLIB`.

For users of other GKS implementations a PAM file, named GKSPACK, is available. Initially, apart from GKSGRAL, this has support only for DECGKS. The PAM file distributed by the CERN Program Library will be included also on the GTS-GRAL distribution tape, and is available at CERN as follows:

**IBM:** via **GIME CERNPAMS**

---

1. At the time of preparing this document there is a possibility that there may be a delay in introducing a few of the GKSPACK routines into the Apollo library due to the change over to SR 10.
Chapter 7. General Hints for GKS Use

VXERN: in CERN_ROOT:PAM/GKSPACK.PAM
Apollo: in /cern/pro/pam/gkspack.pam
UNIX: in $CERN/pro/pam/gkspack.pam

The compiled version for DECGKS is available at CERN:

VXERN: GKS_ROOT:LIB/GKSPACK_DEC.OLB

As mentioned above, GKSPACK includes some routines for which it may not be possible to produce correctly working versions for implementations other than GKSGRAL. For example, this is the situation for GCATOG and GCGTOA with DECGKS. In these cases GKSPACK contains dummy routines with the correct calling sequences, but which write out an error message to a file named GKSPACK.ERR. In fact, for GCATOG and GCGTOA, calling the dummy routines when using a terminal actually stops the application program as well. This is because if the application mixes Fortran I/O and graphics without using correctly working versions of GCATOG and GCGTOA then probably the terminal will block and need to be re-set. (If used via VM/CMS this also can block the communications.)

The GKSPACK routines which provide access to the implementation dependant values for workstation types and connection identifiers require access to a data file. Examples of this file are distributed with GKSPACK, and at CERN are available as follows:

IBM The file is called GKS$IMPL_DAT. A GTS-GRAL GKS version is available on the 'Q' disk, but users may provide their own if they wish.

VAX The file is available via the Logical Name GKS_IMPLEMENTATION. Both GTS-GRAL and DECGKS versions of the file exist, and users must assign the one they wish to use.

    GKS_ROOT:DAT/GKS_IMPLEMENT.GTS
    GKS_ROOT:DAT/GKS_IMPLEMENT.DEC

UNIX The file is accessed via $GKS_IMPLEMENT, which must be defined as an environment variable. A GTS-GRAL GKS version called gks_implem.gts is available in $gkshome, or in /cern/gks/pro/dat.

An example of such a file would be:

```
DECQKS
NB WKTYPES 5

PSTSCR 61 9
T4107 82 0
VT340 17 0
VXUIS 41 0
WISS 5 1
```

2 As opposed to a using a workstation where the Fortran and graphics I/O are directed at different windows.
7.4.1 List of routines

1. CALL GCATOG(WKID) or CALL GCGTOA(WKID)
   **Input:**
   - WKID: GKS workstation identifier of the terminal (INTEGER).
   GCATOG and GCGTOA work only on GKSGRAL with the CERN-supplied driver versions as the routines require modifications to the driver code. The DECGKS version tests the workstation type; if it is VXUIS or VXXW it does nothing, otherwise it writes an error message on the file GKSPACK.ERR.

   The routines change a terminal from graphics to alpha mode (GCGTOA) and from alpha to graphics mode (GCATOG). The terminal must be an activated GKS workstation. Thus they allow the application to perform FORTRAN I/O to the terminal during a graphics session. The effect on the terminal depends on its capabilities. For example, as Pericom PG terminals do not have a dialog area, calling GCGTOA causes the bell to ring and the program then halts until the operator presses return. This gives the user time to look at the image drawn in graphics mode before switching the screen to the alpha-numeric bit plane. However, on terminals with a dialog area (e.g. Pericom MG series and Tektronix 4107 compatible terminals), the FORTRAN I/O will appear immediately overlayed with the graphics.

   After the FORTRAN I/O has been performed the application must call GCATOG to reset the graphics environment. Note that GCGTOA returns immediately, without waiting for a user action. Thus, if one wishes the program to wait until the user has read a message and is ready to continue, then it is essential to include a READ statement, as the following program fragment illustrates:

   ```
   CALL GCGTOA (WKID)
   WRITE(*,*) ' GIVE X COORDINATE: '   
   C Wait for a reply
   READ (*,*) X
   CALL GCATOG (WKID)
   ```

2. CALL GCCDWN(WKID, RWIN, TITLE)
   This routine is described fully in section on Page 67. The DECGKS version is a dummy.
   **Input:**
   - WKID (I): Workstation Identifier
   - RWIN (R*4): Window Size
   - TITLE (C): Window title

3. CALL GCINIT(IDTERM, TERMTMP, IDFILE, LUFILE, KERFIL)
   **Input:**
   - IDTERM: GKS workstation ID for terminal (INTEGER). If ≤ 0 or in batch no terminal is assigned.
   - TERMTMP: GKS terminal workstation type (INTEGER). Note that this number depends on the terminal and the GKS implementation you are using.
   - IDFILE: GKS workstation ID for disk file output (INTEGER). No file is output if ≤ 0.
   - LUFILE: FORTRAN logical unit number for disk file output (INTEGER). Not used if IDFILE ≤ 0.
   - KERFIL: FORTRAN logical unit number for the GKS error file (INTEGER).

   GCINIT provides an easy initialization of GKS for interactive or batch applications. If TERMTMP=0 GCINIT will prompt the interactive user for the terminal type and connection ID. LUFILE and
KERFIL are Fortran logical unit numbers and no file names are assigned as these can easily be provided by the user. Note that the logical unit numbers 91, 92, 93 are reserved for GKSGRAL and, to avoid nasty surprises, do not use 5, 6 or 7. The current values for LUFILE and their effect are:

- \( \text{LUFILE} \leq 0 \): An interactive dialog will guide the user.
- \( 0 < \text{LUFILE} < 100 \): GKS Appendix E metafile on unit LUFILE.
- \( 100 < \text{LUFILE} < 200 \): PostScript file on unit LUFILE-100.
- \( 200 < \text{LUFILE} < 300 \): Encapsulated PostScript file on LUFILE-200.
- \( 1000 < \text{LUFILE} < 1100 \): Tektronix 4014 style file on unit LUFILE-1000

For example, LUFILE=109 would produce PostScript output on the Fortran file defined by logical unit=9.

**Note**

Output of Tektronix 4014 escape codes is available under VMS and Aegis. It can be provided under VM/CMS if requested.

4 **CALL GCGIMP (MAXTYP, GKSNWT, GKSSYN, GKSNCN, GKSSTP)**

Supsplies information in implementation file. See introduction to this section for file naming details.

**Input:**

- **MAXTYP** (I): maximum number of wk types (i.e. dimension of output arrays)

**Output:**

- **GKSNWT** (I): number of wk types in the implementation file
- **GKSSYN** (C*6): array (dimension MAXTYP) with the workstation types names (e.g. "VXUIS ")
- **GKSNCN** (I): array (dimension MAXTYP) with the workstation connection id’s
- **GKSSTP** (I): array (dimension MAXTYP) with the workstation types integer values (e.g. 41)

5 **CALL GCIMPL (IMPLEMENT)**

Supplies implementation name string.

**Output:**

- **IMPLEMENT** (C*6): Name of implementation

6 **CALL GCGWTC (WKYP, CONID)**

Asks the user to enter interactively a workstation type, and provides the list of the available workstation types if the user enters ‘?’. Automatically returns to the program the connection id to be used for the selected workstation type.

**Output:**

- **WKYP** (I): Workstation Type
- **CONID** (I): Connection Id

7 **CALL GCNAME (WKID, STRING)**

**Input:**

- **WKID** (I): GKS workstation identifier of the metafile receiving the frame (INTEGER).
- **STRING** (C*7): The name of the frame (CHARACTER*(*)).

GCNAME enables the user to give the name ‘STRING’ to the current frame when output to a metafile for inclusion in compound documents. The name ‘STRING’ may also be used by
GRVIEW and GRCONV to select frames on the metafile. Users of this facility must place the call to GCNAME before outputting any primitives to a new frame. For examples of its use, see on Page 38 and on Page ??.

Restrictions: Only the first 7 characters of `STRING' are used. If less than 7 characters are given `STRING' will be padded with `$'s (dollars). For VM/CMS system reasons, the character set for `STRING' can only contain upper-case alphabetic, the digits 0-9, and the dollar sign ('$'), so lower-case characters are converted to upper-case.

8 CALL GCQWKN(WKID, IERR, CONID, WKNAM)
Analog to the routine GQWKC but returns a workstation type name instead of a workstation type integer value.

   Input:
   WKID (I) Workstation Id
   Output:
   IERR (I) value returned by GQWKC
   CONID (I) Connection Id
   WKNAM (C*6) Workstation Name (e.g. "VXUIS ")

9 CALL GCSDWN(WKTyp, RWIn, TITLE)
This routine is described fully in section on Page 67. The DECGKS version is a dummy.

   Input:
   WKTyp (I) Workstation Type
   RWIn (R*4) Window Size
   TITLE (C) Window title

10 CALL GCSTOP
GCSTOP deactivates and closes all GKS workstations and closes GKS.

11 CALL GCWTPC(WKTyp, WKTSYN)
Returns the workstation type name corresponding to a workstation type integer value.

   Input:
   WKTyp (I) Workstation Type (e.g. 41 for DECGKS)
   Output:
   WKSTYN (C*6) Workstation Type Name (e.g. "VXUIS ")

12 CALL GCWTPi(WKTSYN, CONID, WKTyp)
Get workstation type and connection id corresponding to a given workstation type name (e.g. 'VXXW').

   Input:
   WKSTYN (C*6) Workstation Type Name
   Output:
   WKTyp (I) Workstation Type
   CONID (I) Connection Id

13 CALL GPRMPT(WKID, PROMPT, LSTRI, REPLY)
Input:
   WKID GKS workstation identifier of the terminal (INTEGER).
Chapter 7. General Hints for GKS Use

Chapter 7. General Hints for GKS Use

PROMPT Application prompt (CHARACTER *(*)).

Output:

LSTRI Length of reply (INTEGER).

REPLY User reply (CHARACTER).

GPRMPT gives a prompt and waits for a reply from the user in a GKS interactive graphics program. As an example, GPRMPT could be used to hold a picture on the screen until the user was ready to view the next one, or terminate the program. If the user hits only a Carriage Return then LSTRI=0 and ‘REPLY’ is not defined. For example:

\[
\begin{align*}
\text{CALL GPRMPT(WKID, 'Type RETURN or QUIT', LSTRI, REPLY) } \\
\text{IF(LSTRI.GT.0) THEN } \\
\text{C Call STOPPG to do whatever you want to do on QUIT } \\
\text{IF REPLY(1:4) .EQ. 'QUIT') CALL STOPPG } \\
\text{ENDIF}
\end{align*}
\]

Currently the prompt is put into a GKS segment which is deleted after the reply. If a segment is open when GPRMPT is called, the prompt will be added to it but it will not be deleted. This could be confusing and should be avoided by closing an open segment before calling GPRMPT. If the workstation is not a terminal or the job is in batch GPRMPT does nothing.

Restrictions: If GPRMPT is used repeatedly within one picture, the prompts will overprint if the terminal does not have selective erasure. If long prompts and/or small workstation windows are used the prompt will be truncated.

14 CALL GRQSK(WKID, LDSTK, NMAX, ISTATS, IT, NP, PX, PY)

Emulates the GTSGRAL request stroke (locator loop) which requires a button push to input each point. This is in contrast to the DECGKS implementation of GRQSK which reads the current cursor position in a loop with fixed time or position intervals.

Note

If it is intended to use this routine to replace the version of GRQSK in the GKS library it must be linked ahead of the library.

Input:

WKID (I) Workstation Identifier
LDSTK (I) Stroke logical device
NMAX (I) Maximum number of points

Output:

ISTATS (I) Status
IT (I) Normalization Transformation Number
NP (I) Number of points returned
PX (I*NMAX) X coordinates
PY (I*NMAX) Y coordinates

15 CALL GUARC(XM, YM, XP, YP, XH, YH, XQ, YQ)

Emulation of GTS-GRAL utility to draw a circular arc defined by 4 points.

Input:

XM, YM Mid point of arc
7.4. **GKSPACK**

XP, YP  Start point
XH, YH  Point on arc
XQ, YQ  Point on end radius of arc

16 **CALL GUBEZ1(N,XP,YP)**
Emulation of GTS-GRAL utility to draw a Bezier curve defined by a Bezier polygon.
**Input:**
N (I)  Dimension of XP, YP
XP, YP  Points on Bezier polygon

17 **CALL GUCIR1(XM,YM,XP,YP)**
Emulation of GTS-GRAL utility to draw a circle defined by (midpoint, peripheral point).
**Input:**
XM, YM  Mid point of circle
XP, YP  Peripheral point on circle

18 **CALL GUCIR2(XM,YM,R)**
Emulation of GTS-GRAL utility to draw a circle defined by (midpoint, radius).
**Input:**
XM, YM  Mid point of circle
R  Radius

19 **CALL GUCUR1(N,XP,YP)**
Emulation of GTS-GRAL utility to draw a curve defined by interpolating points.
**Input:**
N (I)  Dimension of XP, YP
XP, YP  Points in polygon.

20 **CALL GUELL1(XM,YM,A,B)**
Emulation of GTS-GRAL utility to draw an ellipse defined by (midpoint, semi-axes). An ellipse is drawn with midpoint XM,YM; the length of semi-axis in the X-direction is A, and in the Y-direction is B.
**Input:**
XM, YM  Midpoint of Ellipse
A, B  Semi-axes of ellipse in X and Y directions

21 **CALL GUELL2(XM,YM,A,B,BEGRAD,ENDRAD,ROTATE)**
Emulation of GTS-GRAL utility to draw an elliptical arc specified by the midpoint XM,YM, the size of the semi-axes in direction X and Y (A, B), and BEGRAD and ENDRAD which define the radius of the start and end points. The ellipse is rotated with angle ROTATE in an anti-clockwise direction.
**Input:**
XM, YM  Midpoint of Ellipse
A, B  Semi-axes of ellipse in X and Y directions
BEGRAD  Angle of arc start point
ENDRAD  Angle of arc end point
**Chapter 7. General Hints for GKS Use**

**22 CALL GUMEN2(WK,DNR,CHEXL,CHEXH,CHECYL,CHECYH,MENU)**

Emulation of GTS-GRAL utility to define a menu in a given echo area.

**Input:**

- **WKID (I)** Workstation Id
- **DNR (I)** Device Number
- **CHEX<sub>XL</sub>** Echo area X Lower Bound
- **CHEX<sub>H</sub>** Echo area X Higher Bound
- **CHEC<sub>YL</sub>** Echo area Y Lower Bound
- **CHEC<sub>YH</sub>** Echo area Y Higher Bound
- **MENU (C)** String of menu items separated by ‘,’ and terminated by ‘.’

**23 CALL GUNERR(N)**

Dummy routine which writes error message to file GTSTODEC.ERR.

**Output:**

- **N** Number of GKS errors which occurred.

**24 CALL GUSIGD(FLAG)**

Dummy routine which writes error message to file GTSTODEC.ERR.

**Input:**

- **FLAG (L)** Set (.TRUE.) or reset (.FALSE.) simulation flag.

**7.4.2 GKSPACK Error Messages**

**GTSGRAL** If GOPKS has been called before any call to GKSPACK then any GKSPACK errors will be written to the GKS error file. Otherwise, if an error occurs from a call to a GKSPACK routine preceeding the GOPKS call, then the error message will be sent to the screen.

**DECGKS** All errors are written to GKSPACK.ERR.

The list of GKSPACK error messages is as follows:

- **-1** Cannot open implementation file. The file does not exist or: (VMS) the logical name GKS_IMPLEMENTATION is not correctly defined; or (UNIX) the environment variable GKS_IMPL is not correctly defined.
- **-2** Internal error in GCGIMP. The arrays are too small to contain the information for all the workstations listed in the implementation file. Contact Graphics Section at CERN (CN/US/GR).
- **-3** The user arrays are too small to contain the information for all the workstations listed in the implementation file.
- **-4** The GKSPACK routine called is a dummy for this GKS implementation.
- **-5** Error in GCNAME: invalid characters in frame name.
7.4.3  GKSPACK Example Program

The following complete program illustrates the use of some routines available in GKSPACK:

```fortran
PROGRAM DEMOC1

* Include file defines GKS$GRAL workstation types

INCLUDE 'GKS$GTSDEV'

CHARACTER*80 REPLY
REAL XSINX(51),XSINY(51)
REAL XTWOPI
DATA XTWOPI/6.28318/

* DEFINE THE WORKSTATION TYPE (Pericom MG600) AND METAFILE
* CALL GCINIT (1, MG600, 2, 30, 11)

* DEFINE AND SELECT A TRANSFORMATION THAT DIRECTS OUTPUT
* TO A WINDOW OF (0.0, 1.0) X (-1.0, 1.0)
* CALL GCNAME (2, 'TITLE01')
CALL GSWN (1, 0.0, 1.0,-1.0, 1.0)
CALL GSELNT (1)

* BEGIN OUTPUT
* DO 10 K=1,51
XSINX(K) = FLOAT(K-1)*0.02
XSINY(K) = SIN(XSINX(K)*XTWOPI)
10 CONTINUE
CALL GPL (51, XSINX, XSINY)

* ALL DONE WITH GKS, CLOSE THE SYSTEM
* CALL GPRMPT (1, 'TYPE RETURN', LSTRI, REPLY)
CALL GCSTOP
END
```
Chapter 8: GKSGRAL and GKSGRAL-3D

GKSGRAL is a full implementation of the ISO GKS standard. It is written in FORTRAN, and the application interface follows the final version of the FORTRAN binding [6]. The kernel is written to the level '2c', although versions on some systems (and some of the drivers) only support level '2b'. In particular, the version on IBM is only '2b'.

The main reason not to support '2c' is that Event Mode requires the operating and communications systems to support asynchronous interrupts. However, this feature is not required for Sample Mode. Thus, although it requires 'bending' the standard slightly, a pseudo Sample Mode is available on some devices which do not conform to level '2c' using the following procedure:

C Set string mode to sample
CALL GSSTM (WKID,1,1,0)
C Request the locator position
CALL GRQLC (WKID,1,STAT,TNR,PX,PY)
C Sample the character typed
CALL GSMST (WKID,1,1,NCH,STRING)
C Set string mode back to request
CALL GSSTM (WKID,1,0,1)

The current release of GKSGRAL-3D, Version 2.0, follows the final version of the ISO functional standard. It is also very close to the final FORTRAN binding apart from some minor differences which will be resolved at a future date.

8.1 Devices Drivers

Both GKSGRAL and GKSGRAL-3D use the same 2D device drivers. Thus, any 2D device supported by GKSGRAL can also be driven from GKSGRAL-3D. In addition, GKSGRAL-3D supports the IBM 5080, TEKTRONIX 4235/6, and the MEGATEK WHIZZARD series (models 72xx and 33xx) which have 3D hardware transformations.

The list of devices supported may be found in the include file GTSDEV reproduced in Appendix on Page [104] although one should check the latest machine-readable version to see if there have been any changes.

The precise definition of the facilities provided by each workstation driver, such as the assignment of keys, number of available colours, and so on, is given in the Workstation Description Tables. An abridged version of the most common of these may be found in Appendix on Page [112].

8.2 Connection Identifiers

For terminal connections on VAX VMS and VM/CMS, assuming that there are no special instructions in the Workstation Description Table for the device, any number in the range from 1 to 90 may be used for the conid in calls to GOPWK (but see note below). On VAX VMS, GKSGRAL uses an automatically-generated logical name to connect to the terminal, but it is possible to override this with the command:

DEFINE GKSDEVICE_n TTxx:
where \( n \) is the conid to be used and \('TTxx'\) (or \('TXxx'\), \('LTxx'\), \('RTxx'\), \('NVxx'\), etc. depending on the type of controller) is the name of the terminal. Note that this will only allow use of a terminal other than the one on which the user is logged in if the user has the requisite privileges, or if the second terminal is not protected. Moreover, if the second terminal is not hard-wired, for example, if it is connected via a Local Area Network terminal server, then one must take care at the time of use to find out the correct name of the controller and port to which the physical terminal is connected.

If, for debugging or other purposes, one does not want to get any graphical output on the terminal (and no graphical input is requested), then one can connect to the NULL device with the command:

```plaintext
DEFINE GKS_DEVICE_n NL:
```

where \( n \) is the conid to be used.

On VMS, it is possible to capture graphics output command sequences from the HP Plotter, PostScript (see section on Page 39 for information on Encapsulated PostScript) and Tektronix 4014 drivers on a file. To do this, open a file with unit number = \( n \) and define the connection identifier to be conid = \((100 + n)\), where \( n \) is a small integer greater than 1. If there is sufficient interest, this feature could be extended to other devices.

**Note**

At CERN, the GKSGRAL and GKSGRAL-3D packages reserve FORTRAN Logical Unit Numbers (and hence conids) 91-93 for internal use. Apart from these three numbers the actual range of permissible values for connection identifiers is in fact 1-100.

### 8.3 Implementation-Dependent Features

Although an ISO standard, GKS still includes some features which vary from implementation to implementation or device to device. The principle features to watch are text fonts and fill area hatch styles. GKSGRAL supports 11 stroke-precision Latin text fonts plus Greek and a solid filled font. These are illustrated in 8.1. The fonts are available both in italic and non-italic scripts, as well as proportionally and non-proportionally spaced. The font indices are given in Appendix on Page 112.

At CERN, GKSGRAL has been modified to provide 24 Fill Area Hatch Styles (indices -101 to -124), and these should produce the same result on all workstations (see 8.2). Other parameters with which the package has currently been configured are:

- **Workstations**: 10 simultaneously open workstations (6 on VM)
- **Segments**: 2000 segments, names in range 1-32763
- **Normalization Transformations**: 21 (0 to 20)
- **Viewing Transformations**: 21 (0 to 20)
- **Fill Area**: The number of points in a Fill Area primitive is limited to 300.
- **Polyline**: If the number of points in a Polyline is larger than 300 then they will be split into several primitives.

The actual values of these and other parameters may be inquired at run-time by calling the routines:
CALL GQWKM(ERRIND, MXOPWK, MXACWK, MXWKAS)
CALL GQMNTN(ERRIND, MAXTNR)
CALL GQLWK(WTYPE, ERRIND, MPLBTE, MPMBTE, MTXBTE, MFABTE, MPAI, MCOLI)
or
CALL GQLWK3(WTYPE, ERRIND, MPLBTE, MPMBTE, MTXBTE, MFABTE, MPAI, MEDBTE, MCOLI, MVTE)
CALL GQSGP(WTYPE, ERRIND, NSGP)

where the parameters are as follows:

WTYPE  workstation type (input parameter)
ERRIND  error number
MXOPWK  maximum number of simultaneously open workstations
MXACWK  maximum number of simultaneously active workstations
MXWKAS  maximum number of workstations associated with a segment
MAXTNR  maximum normalization transformation number
MPLBTE  maximum number of polyline bundle table entries
MPMBTE  maximum number of polymarker bundle table entries
MTXBTE  maximum number of text bundle table entries
MFABTE  maximum number of fill area bundle table entries
MPAI   maximum number of pattern indices
MEDBTE  maximum number of edge bundle table entries
MCOLI   maximum number of colour indices
MVTE   maximum number of view table entries
NSGP   maximum number of segment priorities

There is unfortunately no function provided to inquire the maximum available number of segments.

picture name=FONTS$$S

Figure 8.1: GTS-GRAL italic, proportionally-spaced fonts

picture name=HATCH$$S

Figure 8.2: CERN-defined hatch patterns
8.4 System-Dependent Considerations

8.4.1 IBM

The recommended way of linking GKS is to use the GKS or GRAFLIB parameters to the CERNLIB command to provide automatic access to just the GKS library, or to both GKS and various higher level graphics packages which use GKS, notably the CERN Program Library packages GKSPACK (J551), HPlot (Y251) and HIGZ (Q120). Further details on what follows may be obtained via the command FIND CERNLIB.

CERNLIB GKS
or
CERNLIB GKS3D
or
CERNLIB GRAFLIB (GTS2D [or (GTS3D]

The optional parameter '(GTS2D' or '(GTS3D' gives access to the 2D or 3D GTS-GRAL libraries. In addition, the CERNLIB command provides access to three versions of each library: OLD, PRO (default), and NEW. For example, to access the previous library version use: 'GRAFLIB!OLD (GTS2D'. Application programs must use VS FORTRAN. The optional INCLUDE files for the GKS parameters are kept in the data set 'GKSINCL MACLIB' on the automatically accessed Q-Disk.

The basic commands and tools to use GKS are:

CERNLIB GKS
To access just GKS (or GKS3D for 3D)
CERNLIB GRAFLIB (GTS2D or GTS3D
To access the GKS and higher level packages
VFORT gksprog
To compile.
LOAD gksprog (START
To execute.

These last two commands may be replaced by the sequence:

VFORT gksprog (GO

In this sequence the CERNLIB Exec makes available the VS FORTRAN libraries, the CERN program library and the GKS library in the correct order, as well as other associated files. Any number of user libraries may be specified.

To use the GKS include files one must give the command:

GLOBAL MACLIB GKSINCL

before compiling the program. Within the code, the files may be included by using the construct:

INCLUDE (ENUM)
INCLUDE (GTSDEV)

where the first file contains the ENUMeration types, and the second contains the GTS-GRAL DEVice types. No compiler options are necessary.

An alternative method to access the information in the include files would be via the PATCHY utility by using the KEEP sequences of GTSDEV and ENUM stored in the PATCH GKSINCL on the GKSPACK Pam file. To access this Pam file on IBM type GIME CERNPAMS.

The example programs listed in the Appendix of the GKS/GKS-3D Primer are available on the disk accessed via GIME GKS with the file names:
Chapter 8. GKSGRAL and GKSGRAL-3D

GKSEXn FORTRAN (where \( n = 1, 6 \))

The GKS and GSK-3D libraries released in October 1989 are compatible with the IBM XA exploitation mode.

### 8.4.2 Use of Segment Storage on IBM

The CERN GKS installation uses logical unit 91 for the dynamic segment storage file. A small file is created on the mini-disk with the largest free writeable space, dynamically extended as necessary, and deleted when GKS is closed. Jobs which crash or fail to call GCLKS will leave the file GKSDYNAM FT91F001 on a mini-disk and it is best to delete it. Applications which create a large number of segments may require up to 5 cylinders of free disk space which can be put on a temporary disk for example.

### 8.4.3 Debugging on IBM

Debugging *interactive* graphics programs on VM is very difficult, but by using two terminals it can be done more comfortably, and one can also use the interactive debugger. The procedure is as follows:

1. Login on the alphanumeric terminal as usual, then inform VM that the graphics output will be on another terminal by the command:
   
   ```
   DEF GRAF 019 3270
   ```
   
   (Where 019 is simply a free address.)

2. Then on the graphics terminal which probably has to be connected through index class 125, connect using the DIAL command:
   
   ```
   DIAL userid
   ```

Now all alphanumeric i/o, debug commands and FORTRAN run time errors will be on the alpha terminal and only the graphics i/o on the DIALled terminal.

### 8.4.4 VAX/VMS

The recommended way of linking GKS is to use the GKS or GRAFLIB parameters to the CERNLIB command to define the symbol\(^1\) `LIB$` which provides access to just the GKS library, or to both GKS and various higher level graphics packages which use GKS, notably the CERN Program Library packages GKS PACK (J551), HPLOT (Y251) and HIGZ (Q120). Further details on what follows may be obtained via the command `HELP CERNLIB`.

```
CERNLIB GKS
or
CERNLIB GKS3D
or
CERNLIB GRAFLIB/GTS2D [or /GTS3D]
```

\(^1\)These commands also define the logical name GKS\_ROOT.
The optional parameter '/GTS2D' or '/GTS3D' gives access to the 2D or 3D versions of the GTS-GRAL package. In addition, the CERNLIB command provides access to three versions of the libraries: OLD, PRO (default), and NEW. For example, to access the previous version of GRAFLIB use the switch 'GRAFLIB/OLD'. The current default version of GKS is the version from GTS-GRAL, and so the switch '/GTS2D' may be omitted.

The default situation is to link to shared versions of the libraries. For non-shared versions it is necessary to add the extra switch '/NOSHAREABLE'.
Chapter 8. GKSGRAL and GKSGRAL-3D

SUBROUTINE GCSDWN (IWTYPE, RWIN, TITLE)
INTEGER IWTYPE
REAL RWIN(4)
CHARACTER*(*) TITLE

Where IWTYPE is the workstation type (8601, 8602, etc...), RWIN is the window size and position, and TITLE contains the name given to the window by the display manager.

- The CERN utility routine GCCDWN (Change Display Window) can be used for VAXstations in order to change interactively inside a program the display window size and position after the window has been created. The calling sequence is:

SUBROUTINE GCCDWN (WKID, RWIN, CODE)
INTEGER WKID
REAL RWIN(4)
INTEGER CODE

Where WKID is the workstation identifier, RWIN is the window size and position, and CODE takes either the value ‘1’ or ‘2’. If CODE=1, then any window size can be defined but the workstation viewport is not changed, which may change the aspect ratio. If CODE=2, then there is a restriction placed on the window size that can be chosen to ensure that the aspect ratio of the window contents remains unchanged. The function uses the largest window that can fit into the size the user has chosen and that has the same height/width ratio as the initial window. The workstation viewport is automatically resized and the workstation updated.

For both GCSDWN and GCCDWN, the window size and position have to be given in the same order as in the file WINDOW.DAT:

RWIN(1) = window size in X in metres
RWIN(2) = window size in Y in metres
RWIN(3) = window position X on screen in metres
RWIN(4) = window position Y on screen in metres

8.4.6 UNIX

$==$ Uptodate ???

Note

File names and directory paths under UNIX are case sensitive.

Since October 1989 the CERN GTS-GRAL licence has been extended to include general use of the company’s software on any UNIX platform, including UNICOS on the CRAY. In addition, from July, 1990, GKSGRAL on Apollo platforms is supported under UNIX, rather than Aegis. However, affiliated institutes with a UNIX licence will still need a licence specifically for Apollo if they wish to receive the Apollo screen drivers.

As the UNIX system runs on many hardware platforms, only a few of which are available at CERN, it may not be possible for the CERN Program Library to distribute binary libraries for the particular machine a user requires. Thus, UNIX users may have to install the software from a TAR file as explained in Appendix on Page 137.

2UNIX is a trade mark of AT&T
To use GKSGRAL or GKSGRAL-3D one requires access to the libraries, font files and include files. These should be made available via environment variables provided by the system manager. For example, if the C shell is being used, the following should be included in the .login file:

```
setenv gkshome /user/gts-gral_root_directory
setenv gkslib $gkshome/gks/libs/gkslib.a
setenv gksdriv $gkshome/gks/libs/gksdriv.a
setenv gks3dlib $gkshome/gks3d/libs/gks3d.a
setenv GKS_FONTS $gkshome/gks/fonts
```

Thus, all variables are defined in terms of $gkshome, which should be set to point to the local root directory. Conversely, one can also store the files in a way analogous to that used for the CERN Program Library, in which case the environment variables should be set to:

```
setenv gkslib /cern/gks/pro/lib/gkslib.a
setenv gksdriv /cern/gks/pro/lib/gksdriv.a
setenv gks3dlib /cern/gks/pro/lib/gks3dlib.a
setenv GKS_FONTS /cern/gks/pro/dat
```

Environment variables may be set under the Bourne or Aegis shells as in the example:

```
GKS_FONTS=/cern/gks/pro/dat; export GKS_FONTS #Bourne
or
GKS_FONTS := '/cern/gks/pro/dat'; export GKS_FONTS #Aegis.
```

To drive terminals attached via RS232 terminal lines, it is necessary also to make a logical connection between the Connection Identifier specified in the call to Open Workstation and the device. GKSGRAL communicates with a set of devices defined by environment variables gksch01 to gksch18, where the numbers 1 to 18 correspond to the connection identifier. Thus, if the connection identifier is set to '1', and one wishes to perform graphics on the same terminal used for alpha-numeric commands, then under the C shell one would need to type:

```
setenv gksch01 /dev/tty
```

To compile and link a Fortran program 'myprog.f' use the commands:

```
f77 myprog.f $gkslib $gksdriv -o myprog.exe
```

In this case the f77 command is used both to compile and link the program. One can also compile and link the program separately. However, even in this case one should probably use the f77 command to link, rather than ld. For example, on DECstations the f77 command automatically includes all the system libraries, whilst ld does not.

```
f77 -c myprog.f -o myprog.o
```

**Note**

In general under UNIX the linker searches libraries in order. Thus, a reference from a library at the end of the sequence to a routine stored in an earlier library will result in an unresolved external reference. This is the case for GKS-3D. A solution is to specify the earlier library twice:
Chapter 8. GKSGRAL and GKSGRAL-3D

The include files used when compiling graphics programs which define the Workstation Types and GKS Parameters (Enumerated Types) are to be found in the directory $gkshome/utl and/or, on systems which follow the CERN library conventions, /cern/gks/pro/utl. They may be accessed from a FORTRAN program as follows:

```fortran
INCLUDE '/cern/gks/pro/utl/gks_gtsdev'
and
INCLUDE '/cern/gks/pro/utl/gks_enum'
```

At run time GKS reads in the files defining the software fonts. These are accessed via an environment variable GKS_FONTS which should defined either by the system or user login procedure.

The example programs listed in the appendix of the GKS/GKS-3D Primer are available either in $gkshome/dmo or in /cern/gks/pro/dmo, with the names gksexN.f (N=1 to 6). However, one may need to edit them in order to set the desired workstation type and to use the correct paths to the include files. On some systems the top directory /cern may not exist, in which case one should use the environment variable $CERN to point to the root directory.

**Note**

*File names and directory paths under UNIX are case sensitive; \$CERN is not equivalent to \$cern!*

A selection of help files, including this one, are to be found in $gkshome/doc.

8.4.7 APOLLO

From July 1990 onwards only Aegis SR 10 and later system releases will be supported. Aegis SR 9.7 library versions are obtainable, but will not be updated. Aegis SR 10 is compatible with UNIX, and so simplifies support. However, this affects file formats and access control, as well as the case sensitivity of file names. Although earlier versions of AEGIS were insensitive to the case of file names, this is no longer true from for SR 10.0 onwards. Thus, for example, the program statement:

```fortran
INCLUDE '/CERN/GKS/PRO/UTL/GKS_GTSDEV'
```

will not work with SR 10.0. The path name within quotes **must be in lower case**. Other points to watch for are that a '/' character is used after the tilde when accessing files via the naming directory, and that when using UNIX shells every command creates a new process, so care must be taken when using scripts to set variables that they run in the context of the current process. For example, by typing `source my_script` under the C shell, or `. my_script` under the Bourne shell.

Apart from the Appendix E metafile and PostScript drivers, the standard CERN APOLLO libraries include drivers for both the GPR and GSR graphics interfaces. For machines such as the DN590, with GSR hardware support, the GSR driver will produce improved performance. This driver also allows a single process to open multiple GKS workstations, each corresponding to a new APOLLO window. The next release of this driver is scheduled to support level C input, as well as a display surface size which may be changed dynamically. To use multiple GSR windows from a single graphics process it is necessary to call GOPWK once to open each workstation with a different value in the series of GSR workstation types.
8.4. System-Dependent Considerations

At run time GKS reads in the files defining the software fonts (which are used by all drivers). These are accessed via an environment variable `GKS_FONTS` which may be created by a command in the startup or `.login` files. At CERN, this environment variable points to the `cernlib` directory `/cern/gks/pro/dat` (or `.../new/dat` or `.../old/dat`, depending on the version required). The fonts are distributed in the directory `$gkshome/gks/fonts`. (In addition to the software fonts used by all drivers, the GSR driver can use also the hardware fonts described below.)

For use of the GPR interface, the user’s home directory should contain a file (or link) called `gks_characteristic` which contains set-up information describing the display. An example of this file may be found in:

```
/cern/gks/pro/dat/gks_characteristic
```

If the user does not have a link from the naming directory to a private copy of `gks_characteristic`, then GKSGRAL will attempt to read a default version which should be made available by creating the following link:

```
crl /dev/gks_characteristic @
/cern/gks/pro/dat/gks_characteristic
```

(As `/dev` is protected, this must be done from a privileged account.) A private copy of the file may be edited to give the desired window size. As an example, the Workstation Type 10002 corresponds to the second line of the file. The standard versions of this file is distributed in `$gkshome/gks/drivers/adgpr`.

The GSR driver requires access to two configuration files, one called `gks_workstations.config` which is similar to the GPR `gks_characteristic` file containing set-up information describing the display, and one called `gks_fonts.config` which lists the available hardware fonts. Copies of these files, modified as necessary, may be stored in (or pointed to by links from) the user’s home directory, or default versions should be made available by creating the links:

```
crl /sys/node_data/gks_workstations.config @
/cern/gks/pro/dat/gks_workstations.config
```
```
crl /sys/node_data/gks_fonts.config @
/cern/gks/pro/dat/gks_fonts.config
```

The standard versions of these files are distributed in `$gkshome/gks/drivers/adgsr`.

As for other UNIX machines, the libraries are stored either in `$gkshome/gks/libs` or `/cern/gks/pro/lib`, and it is easiest to access them via environment variables:

```
setenv gkslib /cern/gks/pro/lib/gkslib_3000.a
setenv gksdriv /cern/gks/pro/lib/gksdriv_3000.a
setenv gks3dlib /cern/gks/pro/lib/gks3d_3000.a
```

Where the ‘3000’ refers to the compiler option used to produce executable modules targeted at the DN3000 style machines with a Motorola 68020/30/40. One can replace ‘3000’ by ‘10000’ for the DN10000 library versions.

To compile and link directly to the libraries use:

```
/com/ftn myprog.ftn -indexl
/com/bind myprog.bin $gkslib $gksdriv -b myprog.exe
```
where the parameter `-indexl` is optional but recommended. The parameter -save may also be used for safety if variables which must be available on re-entry to a routine have not been stored in a COMMON BLOCK. However, use of -save is very detrimental to code optimization, especially on the DN10000. The Aegis versions of the Fortran compiler must be used to ensure that the external global symbols use the same naming conventions as for the libraries (i.e. lower case names with no trailing underscore).

The include files used when compiling graphics programs which define the Workstation Types and GKS Parameters (Enumerated Types) are to be found in the directory `/cern/gks/pro/utl`. (Distributed in `$gkshome/utl`.) They may be accessed from a Fortran program as follows:

```
    INCLUDE '/cern/gks/pro/utl/gks_gtsdev'
    and
    INCLUDE '/cern/gks/pro/utl/gks_enum'
```

(Under SR 10.2 the Fortran compiler accepts VMS syntax for the INCLUDE statement.)

As linking to INLIB (shared) versions of the libraries takes up much less space, and is much faster, INLIB libraries are defined as follows:

```
setenv gkslib_inlib /cern/gks/pro/lib/gkslib_3000.inlib
setenv gks3dlib_inlib /cern/gks/pro/lib/gks3d_3000.inlib
```

These may be linked using the command:

```
/com/bind myprog.o -b myprog.exe -inlib $gkslib_inlib
or
/com/bind myprog.o -b myprog.exe -inlib $gks3dlib_inlib
```

The example programs listed in the appendix of the *GKS/GKS-3D Primer* may be found in:

```
/cern/gks/pro/dmo/gksexN.ftn (N = 1,6)
```

They are distributed in `$gkshome/dmo`.

### 8.4.7.1 APOLLO Models

APOLLO nodes exit with various CPU options. The latest machine, the DN10000, will only work with libraries compiled specifically for this hardware. All other recent APOLLO models use the standard M68020 instruction set, on which can be run libraries compiled with the `3000` option. Note that CERN no longer produces libraries compiled with the `any` switch.

### 8.4.7.2 Apollo Workstation Types

The following consists of a list of the workstation types for the GTS-GRAL workstation drivers installed on Apollo. The full list may be consulted in `gks_gtsdev`, or in Appendix on Page 104.

3     WISS
4     Metafile Output
5     Metafile Input
9701-9708 Apollo GSR interface
10002  Apollo DN300, DN3000, Monochrome (GPR interface)
8.5. **DECGKS: Coexistence with GKSGRAL and Implementation**

10003  Apollo DN550, DN660, Colour (GPR interface)
10004  Apollo DN3000/4000, Colour (GPR interface)
12201  Postscript colour portrait
12202  Postscript colour landscape
12203  Postscript monochrome portrait
12204  Postscript monochrome landscape
10201  2D-Metafile for 3D GKS

### 8.4.8 CRAY under UNICOS

$==>$ Uptodate ???

As the CRAY is used as a batch production service, only the WISS and metafile workstations have been made available in the GKSGRAL library, which should be accessed via the 'cernlib' command. Detailed information on use of the 'cernlib' command for the CRAY is available under VM/CMS by typing: **FIND CRAY CERNLIB**. However, for those users wishing simply to make use of GKS, then the commands to type are:

```
cernlib gks
```

or

```
cernlib gks3d
```

which create a file '.LIB$' in the user’s working directory which contains the required libraries (the case matters). The cernlib command may take the switches -o and -n to allow the selection of old or new library versions.

To compile and link the FORTRAN program 'myprog.ftn' use the commands:

```
cft77 myprog.ftn
segldr -o myprog.exe myprog.o LIB$
```

At run time GKS reads in the files defining the software fonts. These are accessed via an environment variable GKS_FONTS which is defined by the system login procedures.

The include files used when compiling graphics programs which define the Workstation Types and GKS Parameters (Enumerated Types) are to be found in the directory '/cern/gks/pro/utl'. They may be accessed from a FORTRAN program as follows:

```
#include '/cern/gks/pro/utl/gks_gtsdev'
and
#include '/cern/gks/pro/utl/gks_enum'
```

The example programs listed in the appendix of the *GKS/GKS-3D Primer* are not available on the CRAY.

### 8.5 DECGKS: Coexistence with GKSGRAL and Implementation

One of the advantages of using an ISO standard graphics package is that users are not limited to the products of a single supplier. Thus, although the principal GKS implementation supported at CERN is that of GTS-GRAL, users of Digital Equipment Corporation (DEC) machines may wish to use the DEC implementation of GKS. This might be because of the availability of drivers, or because the performance
of DEC software on the company’s own machines is likely to be better than that of third party software suppliers who are constrained to ensure that their libraries operate in many different environments.

Whilst there are no major problems in moving between DECGKS and GKSGRAL there are several implementation dependencies, and these are documented below. A large number of routines have been added to the set of tools in GKSPACK (see section on Page 53). Some of these have been written in order to aid the portability of applications between different GKS implementations by supplying information about the Workstation Types and Connection Identifiers of a particular implementation. Other routines have been provided to emulate extensions of GKS available in the GKSGRAL implementation. Whilst users of GKSGRAL will have these routines available in the GKSGRAL library, users of DECGKS will need to link to an additional library containing a version of GKSPACK tailored for the DECGKS implementation. This library is called GKSPACK_DEC.OLB, and on the CERN VAX cluster may be found in:

GKS_ROOT:\[LIB]\GKSPACK_DEC.OLB

Implementation Dependencies:

- Workstation Types and Connection Ids
  Implementations are free to choose whichever Workstation Types and Connection Identifiers they wish. Thus, those in use by GKSGRAL and DECGKS do not match. The routines in the library GKSPACK go some way to alleviating this problem (see section on Page 53).

- Fonts and Attributes
  Neither GKS, nor any other Graphics Standard, defines the shapes of the characters corresponding to a particular font number. In addition, a particular implementation may provide access to hardware fonts on some devices. This also applies to hatch styles and patterns, etc. The fonts and hatch styles available from GKSGRAL are defined in on Page 63 and a brief comparison of the two implementations follows:

<table>
<thead>
<tr>
<th>GTSGRAL</th>
<th>DECGKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Fonts:</td>
<td></td>
</tr>
<tr>
<td>See wk descr. tables</td>
<td>DECWINDOWS : -101 to -113</td>
</tr>
<tr>
<td>.pa</td>
<td>UIS : -200 to -202</td>
</tr>
<tr>
<td>Software Fonts</td>
<td></td>
</tr>
<tr>
<td>-1 to -11: normal,proport.</td>
<td>font 1 = font -1 = DEC GKS</td>
</tr>
<tr>
<td>-13 : greek</td>
<td>multinational font</td>
</tr>
<tr>
<td>-51 : solid filled font</td>
<td></td>
</tr>
<tr>
<td>same font numbers - 100:</td>
<td>-2 to -23: Hershey fonts</td>
</tr>
<tr>
<td>idem but italics</td>
<td></td>
</tr>
<tr>
<td>same font numbers - 200:</td>
<td></td>
</tr>
<tr>
<td>idem but monospaced</td>
<td></td>
</tr>
<tr>
<td>same font numbers - 300:</td>
<td></td>
</tr>
<tr>
<td>idem but italics monospaced</td>
<td></td>
</tr>
<tr>
<td>Line types</td>
<td>-1 to -8 DEC specific</td>
</tr>
<tr>
<td>Marker Types</td>
<td></td>
</tr>
</tbody>
</table>
8.5. **DECGKS: Coexistence with GKSGRAL and Implementation**

-101 to -114 GKSGRAL specific | -1 to -13 DEC specific

**Fill Area Hatch Styles**

-101 to -124 (CERN specific) | -1 to -33 (UIS specific)  
| -1 to -9 (DECwindows specific)

**Fill Area Patterns**

| None | 1 to 196 (UIS specific)  
| 1 to 28 (DECwindows specific)  
| -29 to -58 (DECwindows specific)

- **Data Records**

Both the contents and internal format of data records used by GKSGRAL and DECGKS are different. The format should not affect the majority of users, who would not require to access data record explicitly. However, users will be affected by the differences in data record contents if they make use of facilities to initialize input devices or use GDPs.

To help solve this problem, higher-level routines have been provided by GTS-GRAL which hide details of the data record contents. These include GUARC, GUBEZ1, GUCIR1, GUCIR2, GUCUR1, GUELL1, GUELL2, and GUMEN2. The library `GKSPACK_DEC.OLB` contains emulations of these routines which work with DECGKS.

- **Metafiles**

Whilst the content of the GTS-GRAL and DECGKS metafiles are logically the same, the file formats are not. In order that the CERN metafile utility programs GRVIEW and GRPLOT may be used with metafiles produced with DECGKS, an option will be introduced into GRCONV to convert them to the same format as those written by GTS-GRAL (but not vice versa). Until this feature is installed, anyone wishing to convert a DECGKS metafile should contact the UCO.

- **Input**

Whilst stroke input requires a trigger for each locator position in the GTS-GRAL GKS implementation, that of DEC does not, but simply samples the locator position at fixed time or distance intervals. Thus, GTS-GRAL’s stroke input is more or less equivalent to calling Request Locator in a loop. In order to provide functionality when using DECGKS equivalent to that in GKSGRAL, a CERN-written version of GRQSK may be found in the library `GKSPACK_DEC.OLB`. The DECGKS implementation uses separate windows for messages and also for string, choice, and valuato input. The window size depends on the echo area specified in GINST, GINCH, and GINVVL.
Appendix A: Example Programs

The programs which follow have been written in VAX FORTRAN-77 as it is easier to read. The
VAX-specific features which would need to be changed in order for the programs to become standard
FORTRAN-77 are:

1 Upper case only
2 In-line comments moved to new line with 'C' in column 1
3 Include statements replaced by the expanded code.

Machine-readable versions of these programs are available at CERN on all the supported systems.

A.1 Polylines and Fill Areas

This program demonstrates the use of polylines and fill area. The workstation chosen here is the Tek-
tronix 4107 (or a compatible).

```
PROGRAM BOX
INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'
INTEGER errfil
PARAMETER(errfil = 10)
INTEGER wkid
PARAMETER(wkid = 1)
INTEGER conid
PARAMETER(conid = 1)
REAL pxa(5), pya(5)
INTEGER asflst(1:13)
DATA pxa /3.0,3.0,7.0,7.0,3.0/
DATA pya /2.0,8.0,8.0,2.0,2.0/
DATA asflst /13 * gindiv/ ! set all ASFs
CALL gopks(errfil, 0) ! open gks (BUFA not used)
CALL gopwk(wkid, conid, T4107) ! open workstation
CALL gsasf(asflst) ! set attributes individually
CALL gacwk(wkid) ! activate workstation
CALL gswn(1, 0.0,10.0,0.0,10.0) ! set window
CALL gsln(gldash) ! set line type
CALL gsfais(ghatch) ! set fill area style
CALL gpl(5, pxa, pya) ! draw polyline
CALL gfa(5, pxa, pya) ! draw fill area
CALL gdawk(wkid) ! deactivate workstation
CALL gclwk(wkid) ! close workstation
CALL gclks ! close GKS
END
```

A.2 Viewports, Polymarkers, and Text

This program draws two ducks in different viewports on the same display surface (see A.1). As well as
the polyline and fill area primitives, it also uses polymarkers and text. The position and size of the text
A.2. Viewports, Polymarkers, and Text

are determined by setting the character up vector and the character height. The workstation selected in this example is the Hewlett Packard 7470A 2 pen plotter.

```
PROGRAM DUCK
INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'
INTEGER errfil, wkid
PARAMETER(errfil = 10, wkid = 1)
REAL pxa(44), pya(44)
INTEGER asflst(1:13)
DATA pxa/0.0,2.0,4.0,6.0,8.0,10.0,12.0,14.0,
    * 16.4,17.0,17.3,17.8,18.5,20.0,22.0,24.0,
    * 26.0,28.0,29.0,28.8,27.2,25.0,23.0,21.5,
    * 21.1,21.5,22.8,24.1,25.1,25.2,24.2,22.1,
    * 20.0,18.0,16.0,14.0,12.0,10.0,8.0,6.1,
    * 4.2,3.0,1.3,0.0/
DATA pya/8.8,7.6,7.1,7.4,8.0,8.9,9.6,9.9,
    * 9.4,9.7,12.0,14.0,16.1,17.0,17.0,16.0,
    * 13.9,13.1,13.2,12.3,11.5,11.5,11.5,11.5,
    * 10.5,9.0,8.0,7.0,5.1,3.6,1.9,1.1,
    * 0.9,0.7,0.1,0.1,1.2,1.8,2.1,
    * 2.9,4.1,6.0,8.8/
DATA asflst /13 * gindiv/ ! set all ASFs

CALL gopks(errfil, 0) ! open gks (BUFA not used)
CALL gopwk(wkid, 6, H475L4) ! open workstation
CALL gsasf(asflst) ! set attributes individually
CALL gacwk(wkid) ! activate workstation

CALL gstxfp(1, gstrkp) ! set font 1, stroke precision
CALL gschh(1.5) ! set char. height

CALL gswn(1, 0.0,30.0,0.0, 30.0)! set window
CALL gsvp(1, 0.0, 0.5, 0.0, 0.4)! set viewport in lower left
CALL gselnt(1)

CALL gsln(glsoli) ! set solid line type
CALL gpl(44, pxa, pya) ! polyline
CALL gsfa(is(g hatch)) ! set hatch fill area
CALL gfa(44, pxa, pya) ! fill area
CALL gschup(-1.0, 8.0) ! set char. up vector
CALL gtx(8.0, 20.0, "Quack") ! text at position 8,20

CALL gsvp(1, 0.5, 1.0, 0.4, 0.8)! move vpt to upper right
CALL gsmk(gast) ! set asterisk marker type
CALL gpmp(44, pxa, pya) ! polymarker
CALL gsfa(is(gsolid)) ! set solid fill area
CALL gfa(44, pxa, pya) ! fill area
CALL gschup(0.5, 1.0) ! reset char. up vector
CALL gtx(7.0, 20.0, ""Quack")! text at position 8,20

CALL gdawk(wkid) ! deactivate workstation
CALL gclwk(wkid) ! close workstation
```
A.3  Text Fonts

This program demonstrates the setting of different text Fonts, precisions and character up vectors. String precision may ignore the character-up-vector. In Font 1, char precision is treated like stroke. Font -2 is implementation-dependent.

```fortran
PROGRAM FONTS
INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'
INTEGER errfil
PARAMETER(errfil = 6)
INTEGER wkid
PARAMETER(wkid = 1)
INTEGER asflst(1:13)
DATA asflst /13 * gindiv/ ! set all ASFs
C
CALL gopks(errfil, 0) ! open gks (BUFA not used)
CALL gopwk(wkid, 6, T4107) ! open workstation
CALL gsasf(asflst) ! set attributes individually
CALL gacwk(wkid) ! activate workstation
CALL gselnt(1) ! select norm transf.
CALL gswin(1, 0.0,10.0,0.0,10.0) ! set window
C
CALL gschh(0.3) ! set character height
CALL gstxfp(1, gstrp) ! set font 1, string precision
CALL gtx(1.0, 1.5, 'Font 1 string prec')
CALL gschup(1.0, 3.0) ! set character up vector
CALL gstxfp(1, gcharp) ! font 1, char precision
CALL gtx(5.0, 8.0, 'Font 1 char prec')
CALL gschup(-1.0, 0.5)
CALL gstxfp(1, gstrkp) ! font 1, stroke precision
CALL gtx(2.0, 4.5, 'Font 1 stroke prec')
CALL gstxfp(-2, gstrp) ! font -2, string precision
CALL gtx(2.0, 3.0, 'Font -2 string prec')
CALL gschup(0.5, -1.0)
CALL gstxfp(-2, gcharp) ! font -2, char precision
CALL gtx(7.0, 6.5, 'Font -2 char prec')
CALL gschup(1.0, 0.0)
CALL gstxfp(-2, gstrkp) ! font -2, stroke precision
CALL gtx(8.0, 5.5, 'Font -2 stroke prec')
CALL gdawk(wkid) ! deactivate workstation
CALL gclwk(wkid) ! close workstation
CALL gclks ! close GKS
END
```
A.4 Request Input

This program demonstrates REQUEST input for the Choice, Locator, Valuator and String input classes. It prompts for locator positions corresponding to the vertices of a polygon, and indicates them with markers. Request Choice is then used to find out the interior style for the fill area, and a title is requested for the picture. The character height of the title is selected using valuator input and is followed by another locator request so that the operator can position the text on the screen.

```
PROGRAM INP
INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'

INTEGER errfil
PARAMETER (errfil=6)
INTEGER wktyp, wkid, conid
PARAMETER (wkid = 1, conid = 1)
INTEGER chcdev, locdev, strdev, valdev ! device numbers
PARAMETER (chcdev = 1, locdev = 1, strdev = 1, valdev = 1)
INTEGER tnr, i, status
INTEGER lstr, nsides, fill
REAL px(10), py(10)
REAL chrht, sides
INTEGER errind ! error flag
CHARACTER*80 str
INTEGER asflst(13)
DATA asflst/13 * gindiv/! set all ASFs

C Open error log file, GKS and a Workstation
C
OPEN (unit=errfil, file='errors', status='unknown')
wktyp = T4107 ! set workstation type
CALL gopks(errfil, 0) ! open gks (bufa not used)
CALL gopwk(wkid, conid, wktyp) ! open workstation
CALL gsasf(asflst) ! set attributes individually

C request locator positions
C
nsides = 6 ! Why not a hexagon?
CALL gmsg(wkid, 'Point to 6 vertices')
DO 30 i = 1, nsides
   CALL grqlc(wkid, locdev, status, tnr, px(i), py(i))
   CALL gpm(1, px(i), py(i)) ! Plot positions
30 CONTINUE

C Request choice for fill area interior style
C
50 CALL gmsg(wkid, 'Type 1-4: hollow,solid,pattern,hatch')
CALL grqch(wkid, chcdev, status, fill)
CALL gsafas(fill-1) ! Set fill area style
CALL gfa(nsides, px, py) ! Draw fill area

C Request string for title of picture
C
CALL gmsg(wkid, 'Give the title of the picture')
CALL grqst(wkid, strdev, status, lstr, str)
```
Appendix A. Example Programs

A.5 Input Initialization

This program is an extension of the last one. It starts by prompting for the number of sides for the polygon, and then continues as before. However, rather than just accept the default settings for the input devices it initializes them in a suitable way. As an example, the choice device is set-up to provide a menu, and the valuator has reasonable limiting values. Note that initialization is implementation-dependent and may also be device-dependent. This example works for GKSGRAL.

```fortran
PROGRAM initst
INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'

INTEGER errfil
PARAMETER (errfil=6)
INTEGER wktyp, wkid, conid
PARAMETER (wkid = 1, conid = 1)
INTEGER chcdev, locdev, strdev, valdev ! device numbers
PARAMETER (chcdev = 1, locdev = 1, strdev = 1, valdev = 1)
INTEGER tnr, pet, i, status, dcunit, lx, ly
INTEGER lstr, nsides, fill, lfi(4)
REAL px(10), py(10)
REAL chrht, sides, rx, ry, inival
PARAMETER (inival = 3.0)
INTEGER errind ! error flag
REAL dum(2) ! dummy array
CHARACTER*40 filename
CHARACTER*80 str(4) ! array used by GPREC
CHARACTER*30 fi
INTEGER asflst(13)
DATA asflst/13 * gindiv/! set all ASFs

OPEN (unit=errfil, file='errors', status='unknown')
wktyp = T4107 ! set workstation type
```
A.5. Input Initialization

CALL gopks(errfil, 0)  ! open gks (bufa not used)
CALL gopwk(wkid, conid, wktyp)  ! open workstation
CALL gcawk(wkid)  ! activate workstation
CALL gsasf(asflst)  ! set attributes individually
CALL gqdsp(wktyp, status, dcunit, rx, ry, lx, ly)

C
C Request valuator for no of sides of polygon (3-10)
C Initialize echo area and valuator upper and lower limits
C
CALL gmsg(wkid, 'Enter the number of sides for the polygon')
    pet = 1  ! define prompt/echo type
    lstr = 0  ! data record not used
CALL ginvl(wkid, valdev, inival, pet, 0.0, rx, 0.0, ry/5,
    * 3.0, 10.0, lstr, str)
CALL grqvl(wkid, valdev, status, sides)
nsides = ifix(sides)

C
C request locator positions
C
CALL gmsg(wkid, 'point to vertices')
CALL grqlc(wkid, locdev, status, tnr, px(1), py(1))
CALL gpm(1, px(1), py(1))
DO 30 i = 2,nsides
C
C Request locator for other positions
C Initialize cursor to previous position
C (for devices with this capability (eg not tek 4014))
C
CALL ginlc (wkid,locdev,0,px(i-1),py(i-1),pet,
    * 0.,rx,0.,ry,lstr,str)
CALL grqlc(wkid, locdev, status, tnr, px(i), py(i))
CALL gpm(1, px(i), py(i))
30 CONTINUE

C
C Request choice for fill area interior style (0-3)
C Use gprec to set up the DATA record used to initialize
C the logical choice device with 4 menu strings.
C
C Note:
C The call to GPREC does not conform to the final FORTRAN Binding.
C
fi(1:6) = 'hollow'
ifi(1) = 6
ifi(7:11) = 'solid'
ifi(2) = 5
ifi(12:18) = 'pattern'
ifi(3) = 7
ifi(19:23) = 'hatch'
ifi(4) = 5
CALL gprec(4, lfi, 0, dum, 23, fi, 4, status, lstr, str)
CALL ginch(wkid, chcdev, 1, 2, 3, 0.8*rx, rx, 0.5*ry, ry,
    * lstr, str)
CALL grqch(wkid, chcdev, status, fill)
CALL gsfais(fill-1)  ! Set fill area style
CALL gfa(nsides, px, py)  ! Draw fill area

C
C Request string for title of picture
Appendix A. Example Programs

Appendix A. Example Programs

A.6 GKS-3D Segments and Viewing

This example illustrates the use of segments, segment transformations, and 3D viewing. The program first sets up the Normalization Transformation to map the whole of the WC space onto the display surface, assuming that this is either a square or a landscape-oriented rectangle. The viewing parameters are set to look along the Z axis towards the origin. The program then draws a tetrahedron in a segment with a Text 3 character string along the front bottom edge, and in the plane of the front face. Next, it re-draws the same tetrahedron in another segment, which is positioned with a segment transformation.

Note that the second tetrahedron is first drawn with visibility off, so that when the transformation is set the un-drawing and re-drawing in the new orientation does not wipe out the first tetrahedron, which would have been exactly underneath. As an alternative, the segment transformation for the second segment could have been set before calling the routine to draw the tetrahedron.

Finally, the program modifies the View Plane Normal, so that the two tetrahedra may be seen from a different direction.

PROGRAM demo3d

Version 2.0, 18.03.88 - New FORTRAN binding

INCLUDE 'GKS$GTSDEV'
INCLUDE 'GKS$ENUM'

INTEGER errfil
PARAMETER (errfil=10)
INTEGER wktyp, wkid, conid

Initialize default string and echo area

CALL gmsg(wkid, 'Give the title of the picture')
CALL ginst(wkid, strdev, 14, 'picture title', pet, * 0.0 , rx, 0.0, 0.1*ry, 80, 1, 1, str)
CALL grqst (wkid, strdev, status, lstr, str)

Request valuator for CHARACTER height

CALL gmsg(wkid, 'Give CHARACTER height (0.01 to 0.1)')
CALL ginvl(wkid, valdev, 0.1, pet, 0.0, rx, 0.0, ry/5, * 0.01 ,0.1, 1, str)
CALL grqvl(wkid, valdev, status, chrht)
CALL gschh(chrht)

Request locator for text position

CALL gmsg(wkid, 'Give text position')
CALL grqlc(wkid, locdev, status, tnr, px(1), py(1))
CALL gstxfp(1, gstrkp) ! font 1, stroke precision
CALL gtx(px(1), py(1), str(1)(1:lstr))
CALL gdawk(wkid) ! deactivate workstation
CALL gclwk(wkid) ! close workstation
CALL gclks ! close gks
END
PARAMETER (wkid = 1, conid = 1)
INTEGER chcdev, locdev, strdev, valdev ! device numbers
PARAMETER (chcdev = 1, locdev = 1, strdev = 1, valdev = 1)
INTEGER errind
REAL vrpx, vrpy, vrpz
REAL vupx, vupy, vupz
REAL vpnx, vpnpy, vpnz
REAL prpu, prpv, prpn
REAL vp(6), wn(6), wkvp(6), wkwn(6), prvp(6)
REAL vpd, bpd, fpd
REAL umin, umax, vmin, vmax
INTEGER iclw, iclb, iclf
INTEGER dcunit, lx, ly, lz
REAL rx, ry, rz
REAL sgmtx(3,4)
REAL vwmtx(4,4)
REAL prmtx(4,4)
INTEGER tnr, vwi
PARAMETER (tnr=1, vwi=1)
CHARACTER*80 str
INTEGER lstr
INTEGER asflst(13)
DATA asflst/13 * gindiv/ ! set all ASFs
DATA asflst/13 * gindiv/ ! set all ASFs
C Set viewing parameters. Look along Z axis TOWARDS origin
C
DATA vrpx, vrpy, vrpz / 0.5, 0.5, 0.5 /
DATA vupx, vupy, vupz / 0.0, 1.0, 0.0 /
DATA vpnx, vpnpy, vpnz / 0.0, 0.0, 1.0 /
DATA prpu, prpv, prpn / 0.0, 0.0, 1.0 /
DATA prvp / 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 /
DATA umin, umax / -0.5, 0.5 /
DATA vmin, vmax / -0.5, 0.5 /
DATA bpd, fpd, vpd / -0.5, 0.5, 0.0 /
C
DATA vp / 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 /
DATA wn / -15.0, 15.0, -15.0, 15.0, -15.0, 15.0 /
DATA wkwn / 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 /
DATA iclw / gnclip /
DATA iclb, iclf / gnclip, gnclip /
C
Open error log file, GKS and a Workstation
C
OPEN (unit=errfil, file='errors', status='unknown')
C
C request the workstation type on which the program is to be run
C
wktyp = T4014 ! set workstation type
CALL gopks(errfil, 0) ! open gks (bufa not used)
CALL gopwk(wkid, conid, wktyp) ! open workstation
CALL gacwk(wkid) ! activate workstation
CALL gsasf(asflst) ! set attributes individually
CALL gsds(wkid, gasap, gperfo) ! As Soon As Possible + regen
C
Set Workstation Window and Viewport to use whole display
C and also the Normalization Transformation Window and Viewport.
C
If the window is not square the Aspect Ratio will be distorted.
Appendix A. Example Programs

CALL gqvol(wktyp, errind, dcunit, rx, ry, rz, lx, ly, lz)
IF (rx .ge. ry) THEN
   wkwn(4) = ry/rx
   vp(4) = ry/rx
ELSE
   wkwn(2) = rx/ry
   vp(2) = rx/ry
ENDIF
CALL gwkw3(wkid, wkwn)
CALL gsv3(tnr, vp)
wkvp(1) = 0.0
wkvp(2) = rx
wkvp(3) = 0.0
wkvp(4) = ry
wkvp(5) = 0.0
wkvp(6) = rz
CALL gwkv3(wkid, wkvp)
CALL gsv3(tnr, wn)

CALL gselnt(tnr) ! Select Normalization Tfrm
CALL gsvwi(vwi) ! Select Viewing Tfrm
CALL gsclip(gnclip) ! Set clipping off

CALL gevwm(vrpx, vrpy, vrpz, vupx, vupy, vupz,
           * vpnx, vpany, vpzn, gndc, errind, vwmtx)
IF (errind .ne. 0) THEN
   WRITE(6, *) 'Error in Evaluate View Matrix ', errind
   GOTO 9999
ENDIF
CALL gevpm(umin, umax, vmin, vmax, prvp, gparl,
           * prpu, prpv, prpn, vpd, bpd, fpd, errind, prmtx)
IF (errind .ne. 0) THEN
   WRITE(6, *) 'Error in Evaluate Projection Matrix ', errind
   GOTO 9999
ENDIF
CALL gsvwr(wkid, vwi, vwmtx, prmtx, prvp, iclw, iclb, iclf)
CALL guwk(wkid, gperfo) ! Update Workstation

CALL gcrgs(1) ! Create segment 1
CALL tetra ! Draw a tetrahedron
CALL gcclg ! close segment
CALL gcrgs(2) ! Create segment 2
CALL gsvvis(2, ginvis) ! Make it invisible
CALL tetra ! Draw a tetrahedron
CALL gcclg ! Close segment

CALL gevtm3(2.0, 3.0, 4.0, -6.0, 3.0, 0.5,
           * 0.0, 0.3, 0.7, 0.5, 0.5, 0.7, gwc, sgmtx)
CALL gssgt3(2, sgmtx) ! Transform segment
CALL gsvi(2, gvisi)  ! and make it visible
C
C  Give user a chance to see result, then transform view
C
CALL gmsg (wkid,'Hit <return> to continue')
CALL grqst(wkid, strdev, errind, lstr, str)
C
vpnx = 0.3  ! Change direction of
vpny = 0.6  ! View Plane Normal
vpnz = 1.0
CALL gevwm(vrpx, vrpy, vrpz, vupx, vupy, vupz,
*          vpnx, vpny, vpnz, gndc, errind, vwmtx)
IF (errind .ne. 0) THEN
  WRITE(6,*) 'Error in EValculate VieW Matrix ', errind
  GOTO 9999
ENDIF
CALL gsvwr(wkid, vwi, vwmtx, prmtx, prvp, iclw, iclb, iclf)
CALL guwk(wkid, gperfo)  ! Update Workstation
C
C  Give user a chance to see result, then exit
C
CALL gmsg(wkid,'Hit <return> to continue')
CALL grqst(wkid, strdev, errind, lstr, str)
C
9999 CALL gdawk(wkid)  ! deactivate workstation
CALL gc1wk(wkid)  ! close workstation
CALL gc1ks  ! close gks
END
SUBROUTINE tetra
C
C  Draw a tetrahedron with a 3D text string along one edge
C
INCLUDE 'GKS$ENUM'
C
REAL plax(6), play(6), plaz(6)
REAL plbx(2), plby(2), plbz(2)
REAL vx(2), vy(2), vz(2)
DATA plax / 0.0, -5.7, 5.7, 0.0, 0.0, -5.7 /
DATA play / -5.0, -5.0, -5.0, -5.0, 10.0, -5.0 /
DATA plaz /-10.0, 5.0, 5.0,-10.0, 0.0, 5.0 /
DATA plbx / 0.0, 5.7/
DATA plby / 10.0, -5.0 /
DATA plbz / 0.0, 5.0 /
C
CALL gp13(6, plax, play, plaz)
CALL gp13(2, plbx, plby, plbz)
CALL gschh(1.5)  ! Set character height
vx(1) = plax(3) - plax(2)  ! Text Direction Vector 1
vy(1) = play(3) - play(2)  ! Text in front plane of
vz(1) = plaz(3) - plaz(2)  ! the tetrahedron
vx(2) = plax(5) - plax(2)  ! Text Direction Vector 2
vy(2) = play(5) - play(2)
vz(2) = plaz(5) - plaz(2)
CALL gschup(0.0, 1.0)  ! Set character up vector
CALL gstxfp(1, gstrkp)  ! font 1, stroke precision
CALL gtx3(-5.0, -5.0, 5.0,  ! Write along bottom line
*          vx, vy, vz, 'Demo-3D')
Appendix A. Example Programs

END
Appendix B: GKS/GKS-3D Error Codes

The error codes defined in this section are of two types:

1. Those which are defined by GKS/GKS-3D.
2. Those which are implementation-dependent. These are between 900 and 2000.

B.1 STATES

1. GKS not in proper state: GKS shall be in the state GKCL
2. GKS not in proper state: GKS shall be in the state GKOP
3. GKS not in proper state: GKS shall be in the state WSAC
4. GKS not in proper state: GKS shall be in the state SGOP
5. GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
6. GKS not in proper state: GKS shall be either in the state WSOP or in the state WSAC
7. GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, or SGOP
8. GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC or SGOP

B.2 WORKSTATIONS

20. Specified workstation identifier is invalid
21. Specified connection identifier is invalid
22. Specified workstation type is invalid
23. Specified workstation type does not exist
24. Specified workstation (wkid or [condi, wtype] pair) is open
25. Specified workstation is not open
26. Specified workstation cannot be opened
27. Workstation Independent Segment Storage is not open
28. Workstation Independent Segment Storage is already open
29. Specified workstation is active
30. Specified workstation is not active
31. Specified workstation is of category MO (Metafile Output)
32. Specified workstation is not of category MO (Metafile Output)
33. Specified workstation is of category MI (Metafile Input)
34. Specified workstation is not of category MI (Metafile Input)
35. Specified workstation is of category INPUT
36. Specified workstation is Workstation Independent Segment Storage
37. Specified workstation is not of category OUTIN
38. Specified workstation is neither of category INPUT nor of category OUTIN
Appendix B. GKS/GKS-3D Error Codes

39 Specified workstation is neither of category OUTPUT nor of category OUTIN
40 Specified workstation has no pixel store read-back capability
41 Specified workstation type is not able to generate the specified drawing primitive
42 Maximum number of simultaneously open workstations would be exceeded.
43 Maximum number of simultaneously active workstations would be exceeded.

B.3 2D TRANSFORMATIONS

50 Transformation number is invalid
51 Rectangle definition is invalid
52 Viewport is not within the Normalized Device Coordinate unit square
53 Workstation window is not within the Normalized Device Coordinate unit square
54 Workstation viewport is not within the display space

B.4 2D OUTPUT ATTRIBUTES

60 Polyline index is invalid
61 A representation for the specified polyline index has not been defined on this workstation
62 Line type is less than or equal to zero
63 Specified line type is not supported on this workstation
64 Polymarker index is invalid
65 A representation for the specified polymarker index has not been defined on this workstation
66 Marker type is less than or equal to zero
67 Specified marker type is not supported on this workstation
68 Text index is invalid
69 A representation for the specified text index has not been defined on this workstation
70 Text font is less than or equal to zero
71 Requested text font is not supported for the specified precision on this workstation
72 Character expansion factor is less than or equal to zero
73 Character height is less than or equal to zero
74 Length of character up vector is zero
75 Fill area index is invalid
76 A representation for the specified fill area index has not been defined on this workstation
77 Specified fill area interior style is not supported on this workstation
78 Style (pattern or hatch) index is less than or equal to zero
79 Specified pattern index is invalid
80 Specified hatch style is not supported on this workstation
B.5 2D OUTPUT PRIMITIVES

81 Pattern size value is not positive
82 A representation for the specified pattern index has not been defined on this workstation
83 Interior style PATTERN is not supported on this workstation
84 Dimensions of colour array are invalid
85 Colour index is less than zero
86 Colour index is invalid
87 A representation for the specified colour index has not been defined on this workstation
88 Colour is outside range [0,1]
89 Pick identifier is invalid
90 Interior style PATTERN not supported by this workstation.
91 Dimensions of colour array invalid
92 Colour index less than zero
93 Colour index invalid
94 A representation for the specified colour index has not been specified on this workstation.
95 A representation for the specified colour index has not been pre-defined on this workstation.
96 Colour is outside range [0,1].
97 Pick identifier is invalid

B.5 2D OUTPUT PRIMITIVES

100 Number of points is invalid
101 Invalid code in string
102 Generalized drawing primitive identifier is invalid
103 Content of generalized drawing primitive data record is invalid
104 At least one active workstation is not able to generate the specified generalized drawing primitive
105 At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformations and clipping rectangle.

B.6 SEGMENTS

120 Specified segment name is invalid
121 Specified segment name is already in use
122 Specified segment does not exist
123 Specified segment does not exist on specified workstation
124 Specified segment does not exist on workstation Independent Segment Storage
125 Specified segment is open
126 Segment priority is outside the range [0,1]
-127 (GTS-GRAL only) Number of segments exceeded, create failed.
-128 (GTS-GRAL only) Transformation matrix invalid
B.7 INPUT

140 Specified input device is not present on workstation
141 Input device is not in REQUEST mode
142 Input device is not in SAMPLE mode
143 EVENT and SAMPLE input are not available at this level of GKS
143 (GTS-GRAL only) EVENT and SAMPLE input are not available for the specified workstation.
144 Specified prompt and echo type is not supported on this workstation
145 Echo area is outside display space
146 Contents of input data record are invalid
147 Input queue has overflowed
148 Input queue has not overflowed since GKS was opened or the last invocation of Inquire Input Queue Overflow
149 Input queue has overflowed, but associated workstation has been closed
150 No input value of the correct class is in the current event report
151 Timeout is invalid
152 Initial value is invalid
153 Length of initial string is greater than the implementation defined maximum
154 Length of initial string is greater than the buffer size

B.8 METAFILES

160 Item type is not allowed for user items
161 Item type is invalid
162 No item is left in GKS metafile input
163 Metafile item is invalid
164 Item type is not a valid GKS item
165 Content of item data record is invalid for the specified item type
166 Maximum item data record length is invalid
167 User item cannot be interpreted
168 Specified function not supported by this level of GKS-3D
170 (GTS-GRAL only) Wrong CGM code during TEST OPEN
171 (GTS-GRAL only) CGM not in state TEST OPEN during APPEND TEXT
172 (GTS-GRAL only) Local text buffer too small

B.9 ESCAPE

180 Specified function is not supported
181 Specified escape function identification is invalid
182 Contents of escape data record are invalid
B.10 MISCELLANEOUS

200 Specified error file is invalid

B.11 SYSTEM

300 Specified function is not supported in this level of GKS
301 Storage overflow has occurred in GKS
302 Input/Output error has occurred whilst reading (eg font files)
303 Input/Output error has occurred whilst writing
304 Input/Output error has occurred whilst sending data to a workstation
305 Input/Output error has occurred whilst receiving data from a workstation
306 Input/Output error has occurred during program library management
307 Input/Output error has occurred whilst reading workstation description table
308 Arithmetic error has occurred

B.12 3D TRANSFORMATIONS

400 View up vector and view plane normal are collinear
401 View plane normal is null a vector
402 View up vector is a null vector
403 Projection viewport limits are not within NPC range
404 Projection reference point is between front and back clipping planes
405 Projection reference point is on the view plane
406 Box definition is invalid
407 Viewport is not within NDC unit cube
408 Specified view index is invalid
409 A representation for the specified view index has not been defined on this workstation.
410 A representation for the specified view index has not been pre-defined on this workstation.
411 Workstation window limits are not within the NPC unit cube
412 Back clipping plane is in front of front clipping plane

B.13 3D OUTPUT ATTRIBUTES

420 Edge index is invalid
421 A representation for the specified edge index index has not been defined on this workstation.
422 A representation for the specified edge index index has not been pre-defined on this workstation.
423 Edge type equal to zero
Appendix B. GKS/GKS-3D Error Codes

424 Specified edge type not supported by this workstation
425 Edge width scale factor less than zero
426 Pattern reference vectors are collinear
427 Specified HLHSR mode not supported on this workstation
428 Specified HLHSR identifier invalid
429 Specified HLHSR mode invalid

B.14 3D OUTPUT PRIMITIVES

430 The text direction vectors are collinear
431 List of points is invalid
432 At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformations and clipping volume.

B.15 IMPLEMENTATION DEPENDENT

B.15.1 Device independent errors

-600 (GTS-GRAL only) Invalid values for cubic curve
900 Enumeration type out of range
901 Output parameter size insufficient
902 List member or set element not available (for inquiry routines)
903 Invalid data record - cannot be decoded (Errors 900-903 may also appear as errors 500-503. Stay tuned.)

-920 (GTS-GRAL only) DSYS-error: keys outside range (MINKEY,MAXKEY)
-921 (GTS-GRAL only) DSYS-error: key (K1,K2) already present
-922 (GTS-GRAL only) DSYS-error: key (K1,K2) does not exist
-924 (GTS-GRAL only) DSYS-error: buffer size in GKDSGC too small, records truncated
-925 (GTS-GRAL only) DSYS-error: incorrect record size used (installation problem)
-927 (GTS-GRAL only) DSYS-error: whilst reading record from external file
-928 (GTS-GRAL only) DSYS-error: whilst writing record to external file
-929 (GTS-GRAL only) DSYS-error: whilst reading record from external file (Garbage file)
-930 (GTS-GRAL only) DSYS-error: whilst writing record to external file (Garbage file)
-950 (GTS-GRAL only) Missing font files
-1000 (GTS-GRAL only) Implementation Error
B.15.2 Device dependent errors

-7301 (GTS-GRAL only) UIS driver error in file WINDOW.DAT
-7302 (GTS-GRAL only) UIS driver error: not enough coordinates to open window
-7303 (GTS-GRAL only) UIS driver error: no logical unit is free

Typical 7xxx errors include the inability to open or read Font files, Apollo gks_characteristic files, etc. These are usually due to incorrect installation. For example, files may be missing or have the wrong format, windows may have illegal sizes, etc.

B.15.3 Error codes specific to the FORTRAN language binding

2000 Enumeration type out of range; the integer passed as a GKS enumerated type is not within the range of valid values.
2001 Output parameter size insufficient; a FORTRAN array or string being passed as an output parameter is too small to contain the returned information.
2002 List element or member not available; for a non-empty list or set, a value less than zero or greater than the size of a list or set was passed as the requested list element or set member in an inquiry routine.
2003 Invalid data record; the data record cannot be decoded, or there was a problem encountered when the data record was created, making the result invalid.

B.16 RESERVED ERRORS

Unused error numbers less than 2000 are reserved for future standardization.
Error numbers 2000-3999 are reserved for language bindings.
Error numbers greater than 4000 are reserved for future registration.
Appendix C: Metafile Items

Note that whilst the structure of the 2D and 3D GKS Appendix E metafiles is the same, their contents are not inter-changeable. For example, item 11 corresponds to a polyline in both cases. However, in the 3D metafile each point in item 11 is represented by three coordinates, rather than by 2 coordinates in the 2D case.

C.1 GKS

The following is a list of the GKS Appendix E Metafile item numbers (2D).

**General**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>End item (last item of every metafile)</td>
</tr>
<tr>
<td>1</td>
<td>Clear workstation</td>
</tr>
<tr>
<td>2</td>
<td>Re-draw all segments on workstation</td>
</tr>
<tr>
<td>3</td>
<td>Update workstation</td>
</tr>
<tr>
<td>4</td>
<td>Set deferral state</td>
</tr>
<tr>
<td>5</td>
<td>Message</td>
</tr>
<tr>
<td>6</td>
<td>Escape</td>
</tr>
</tbody>
</table>

**Items for Output Attributes**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Polyline</td>
</tr>
<tr>
<td>12</td>
<td>Polymarker</td>
</tr>
<tr>
<td>13</td>
<td>Text</td>
</tr>
<tr>
<td>14</td>
<td>Fill Area</td>
</tr>
<tr>
<td>15</td>
<td>Cell Array</td>
</tr>
<tr>
<td>16</td>
<td>Generalized Drawing Primitive</td>
</tr>
</tbody>
</table>

**Items for Output Primitive Attributes**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Polyline index</td>
</tr>
<tr>
<td>22</td>
<td>Line type</td>
</tr>
<tr>
<td>23</td>
<td>Line width scale factor</td>
</tr>
<tr>
<td>24</td>
<td>Polyline colour index</td>
</tr>
<tr>
<td>25</td>
<td>Polymarker index</td>
</tr>
<tr>
<td>26</td>
<td>Marker type</td>
</tr>
</tbody>
</table>
C.1. GKS

Marker size scale factor
Polymarker colour index
Text index
Text font and precision
Character expansion factor
Character spacing
Text colour index
Character vectors
Text path
Text alignment
Fill area index
Fill area interior style
Fill area style index
Fill area colour index
Pattern size
Pattern reference point
Aspect source flags
Pick identifier

Items for Workstation Attributes

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>Polyline representation</td>
</tr>
<tr>
<td>52</td>
<td>Polymarker representation</td>
</tr>
<tr>
<td>53</td>
<td>Text representation</td>
</tr>
<tr>
<td>54</td>
<td>Fill area representation</td>
</tr>
<tr>
<td>55</td>
<td>Pattern representation</td>
</tr>
<tr>
<td>56</td>
<td>Colour representation</td>
</tr>
</tbody>
</table>

Items for Transformation

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>Clipping area</td>
</tr>
<tr>
<td>62</td>
<td>Clipping indicator</td>
</tr>
<tr>
<td>71</td>
<td>Workstation window</td>
</tr>
<tr>
<td>72</td>
<td>Workstation viewport</td>
</tr>
</tbody>
</table>
### Items for Segment Manipulation

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>Create segment</td>
</tr>
<tr>
<td>82</td>
<td>Close segment</td>
</tr>
<tr>
<td>83</td>
<td>Rename segment</td>
</tr>
<tr>
<td>84</td>
<td>Delete segment</td>
</tr>
</tbody>
</table>

### Items for Segment Attributes

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>Set segment transformation</td>
</tr>
<tr>
<td>92</td>
<td>Set visibility</td>
</tr>
<tr>
<td>93</td>
<td>Set highlighting</td>
</tr>
<tr>
<td>94</td>
<td>Set segment priority</td>
</tr>
<tr>
<td>95</td>
<td>Set detectability</td>
</tr>
</tbody>
</table>

### User Items

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;100</td>
<td>User item (comments etc)</td>
</tr>
</tbody>
</table>

### C.2 GKS-3D

The following is a list of the GKS-3D Appendix E Metafile item numbers.

### General

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>End item (last item of every metafile)</td>
</tr>
<tr>
<td>1</td>
<td>Clear workstation</td>
</tr>
<tr>
<td>2</td>
<td>Re-draw all segments on workstation</td>
</tr>
<tr>
<td>3</td>
<td>Update workstation</td>
</tr>
<tr>
<td>4</td>
<td>Set deferral state</td>
</tr>
<tr>
<td>5</td>
<td>Message</td>
</tr>
<tr>
<td>6</td>
<td>Escape</td>
</tr>
</tbody>
</table>
### Items for Output Attributes

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Polyline 3</td>
</tr>
<tr>
<td>12</td>
<td>Polymarker 3</td>
</tr>
<tr>
<td>13</td>
<td>Text 3</td>
</tr>
<tr>
<td>14</td>
<td>Fill Area 3</td>
</tr>
<tr>
<td>15</td>
<td>Fill Area Set 3</td>
</tr>
<tr>
<td>16</td>
<td>Cell Array 3</td>
</tr>
<tr>
<td>17</td>
<td>Generalized Drawing Primitive 3</td>
</tr>
</tbody>
</table>

### Items for Output Primitive Attributes

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Polyline index</td>
</tr>
<tr>
<td>22</td>
<td>Line type</td>
</tr>
<tr>
<td>23</td>
<td>Line width scale factor</td>
</tr>
<tr>
<td>24</td>
<td>Polyline colour index</td>
</tr>
<tr>
<td>25</td>
<td>Polymarker index</td>
</tr>
<tr>
<td>26</td>
<td>Marker type</td>
</tr>
<tr>
<td>27</td>
<td>Marker size scale factor</td>
</tr>
<tr>
<td>28</td>
<td>Polymarker colour index</td>
</tr>
<tr>
<td>29</td>
<td>Text index</td>
</tr>
<tr>
<td>30</td>
<td>Text font and precision</td>
</tr>
<tr>
<td>31</td>
<td>Character expansion factor</td>
</tr>
<tr>
<td>32</td>
<td>Character spacing</td>
</tr>
<tr>
<td>33</td>
<td>Text colour index</td>
</tr>
<tr>
<td>34</td>
<td>Character vectors</td>
</tr>
<tr>
<td>35</td>
<td>Text path</td>
</tr>
<tr>
<td>36</td>
<td>Text alignment</td>
</tr>
<tr>
<td>37</td>
<td>Fill area index</td>
</tr>
<tr>
<td>38</td>
<td>Fill area interior style</td>
</tr>
<tr>
<td>39</td>
<td>Fill area style index</td>
</tr>
<tr>
<td>40</td>
<td>Fill area colour index</td>
</tr>
<tr>
<td>41</td>
<td>Set edge index</td>
</tr>
<tr>
<td>42</td>
<td>Set edge flag</td>
</tr>
<tr>
<td>43</td>
<td>Set edge type</td>
</tr>
</tbody>
</table>
Appendix C. Metafile Items

44  Set edge width scale factor
45  Set edge colour index
46  Pattern size
47  Pattern reference point and vectors
48  Aspect source flags 3
49  Aspect source flags
50  Pick identifier

**Items for Workstation Attributes**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>Polylne representation</td>
</tr>
<tr>
<td>52</td>
<td>Polymarker representation</td>
</tr>
<tr>
<td>53</td>
<td>Text representation</td>
</tr>
<tr>
<td>54</td>
<td>Fill area representation</td>
</tr>
<tr>
<td>55</td>
<td>Edge representation</td>
</tr>
<tr>
<td>56</td>
<td>Pattern representation</td>
</tr>
<tr>
<td>57</td>
<td>Colour representation</td>
</tr>
</tbody>
</table>

**Items for Transformation**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>Clipping volume</td>
</tr>
<tr>
<td>62</td>
<td>Clipping indicator</td>
</tr>
<tr>
<td>63</td>
<td>View index</td>
</tr>
<tr>
<td>64</td>
<td>View representation 3</td>
</tr>
<tr>
<td>65</td>
<td>HLHSR identifier</td>
</tr>
<tr>
<td>66</td>
<td>HLHSR mode</td>
</tr>
<tr>
<td>71</td>
<td>Workstation window 3</td>
</tr>
<tr>
<td>72</td>
<td>Workstation viewport 3</td>
</tr>
</tbody>
</table>

**Items for Segment Manipulation**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>Create segment</td>
</tr>
<tr>
<td>82</td>
<td>Close segment</td>
</tr>
<tr>
<td>83</td>
<td>Rename segment</td>
</tr>
<tr>
<td>84</td>
<td>Delete segment</td>
</tr>
</tbody>
</table>
### Items for Segment Attributes

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>Set segment transformation 3</td>
</tr>
<tr>
<td>92</td>
<td>Set visibility</td>
</tr>
<tr>
<td>93</td>
<td>Set highlighting</td>
</tr>
<tr>
<td>94</td>
<td>Set segment priority</td>
</tr>
<tr>
<td>95</td>
<td>Set detectability</td>
</tr>
</tbody>
</table>

### User Items

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;100</td>
<td>User item (comments etc)</td>
</tr>
</tbody>
</table>
Appendix D: GKS Include Files

At CERN there are several machine-readable include files which should be used when specifying GKS enumerated types and workstations. The use of symbolic names greatly aids programming and, even more important, program debugging.

The include file GTSDEV.INC contains the symbolic names and codes for the various Workstation Types available for GKSGRAL.

The following is a listing of the file ENUM.INC, containing mnemonic FORTRAN names and their values for the GKS ENUMERATION types.

```c
C ENUM.INC
C GKS and GKS-3D Enumeration Types
C ISO/DIS-8651-1 & ISO/IEC DIS 8806-1
C (Last Update: 27-04-89)
C
C aspect source: bundled individual
  INTEGER GBUNDL, GINDIV
  PARAMETER (GBUNDL=0, GINDIV=1)
C
C clear control flag: conditionally, always
  INTEGER GCONDI, GALWAY
  PARAMETER (GCONDI=0, GALWAY=1)
C
C clipping disable and enable
  INTEGER GNCLIP, GCLIP
  PARAMETER (GNCLIP=0, GCLIP=1)
C
C colour available: monochrome, colour
  INTEGER GMONOC, GCOLOR
  PARAMETER (GMONOC=0, GCOLOR=1)
C
C coordinate switch: World Coordinates, Normalized Device Coordinates
  INTEGER GWC, GNDC
  PARAMETER (GWC=0, GNDC=1)
C
C deferral mode: ASAP, BNIG, BNIL, ASTI
  INTEGER GASAP, GBNIG, GBNIL, GASTI
  PARAMETER (GASAP=0, GBNIG=1, GBNIL=2, GASTI=3)
C
C detectability: undetectable, detectable
  INTEGER GUNDET, GDETEC
  PARAMETER (GUNDET=0, GDETEC=1)
C
C device coordinate units: meters, other
  INTEGER GMETRE, GOTHU
  PARAMETER (GMETRE=0, GOTHU=1)
C
C display surface: empty not-empty, empty
  INTEGER GNEMPT, GEMPTY
  PARAMETER (GNEMPT=0, GEMPTY=1)
C
C dynamic modification: IRG, IMM
  INTEGER GIRG, GIMM
  PARAMETER (GIRG=0, GIMM=1)
```
C echo switch: no-echo, echo
   INTEGER  GNECHO, GECHO
   PARAMETER (GNECHO=0, GECHO=1)
C
C fill area interior style: hollow, solid, pattern, hatch
   INTEGER  GHOLLO, GSOLID, GPATTR, GHATCH
   PARAMETER (GHOLLO=0, GSOLID=1, GPATTR=2, GHATCH=3)
C
C highlighting: normal, highlighted
   INTEGER  GNORML, GHILIT
   PARAMETER (GNORML=0, GHILIT=1)
C
C input device status: none, ok, no-pick, no-choice
   INTEGER  GNONE, GOK, GN PICK, GNCHOI
   PARAMETER (GNONE=0, GOK=1, GN PICK=2, GNCHOI=2)
C
C input class: none, locator, stroke, valuator, choice, pick, string
   INTEGER  GNCLAS, GLOCAT, GSTROK, GVALUA,
      *  GCHOIC, GPICK, GSTRIN
   PARAMETER (GNCLAS=0, GLOCAT=1, GSTROK=2, GVALUA=3,
      *  GCHOIC=4, GPICK=5, GSTRIN=6)
C
C implicit regeneration: mode suppressed, allowed
   INTEGER  GSUPPD, GALLOW
   PARAMETER (GSUPPD=0, GALLOW=1)
C
C level of GKS: L0a, L0b, L0c, L1a, L1b, L1c, L2a, L2b, L2c
   INTEGER  GL0A, GL0B, GL0C, GL1A, GL1B,
      *  GL1C, GL2A, GL2B, GL2C
   PARAMETER (GL0A=0, GL0B=1, GL0C=2, GL1A=3, GL1B=4,
      *  GL1C=5, GL2A=6, GL2B=7, GL2C=8)
C
C new frame action necessary: no, yes
   INTEGER  GNO, GYES
   PARAMETER (GNO=0, GYES=1)
C
C off/on switch for edge flag
   INTEGER  GDFF, GDN
   PARAMETER (GDFF=0, GDN=1)
C
C operating mode: request, sample, event
   INTEGER  GREQU, GSAMPL, GEVENT
   PARAMETER (GREQU=0, GSAMPL=1, GEVENT=2)
C
C operating state value: GKS closed, GKS open, Workstation open,
C Workstation active, Segment open
   INTEGER  GGKCL, GGKOP, GWSOP, GWSAC,
      *  GSOP
   PARAMETER (GGKCL=0, GGKOP=1, GWSOP=2, GWSAC=3,
      *  GSOP=4)
C
C presence of invalid values: absent, present
   INTEGER  GABSNT, GPRSNT
   PARAMETER (GABSNT=0, GPRSNT=1)
C
C projection type for 3D: Parallel or Perspective
INTEGER GPRL, GPER
PARAMETER (GPRL=0, GPER=1)

C C regeneration flag: postpone, perform
INTEGER GPOSTP, GPERFO
PARAMETER (GPOSTP=0, GPERFO=1)

C C relative input priority: higher, lower
INTEGER GHIGHR, GLOWER
PARAMETER (GHIGHR=0, GLOWER=1)

C C simultaneous events flag: no-more, more
INTEGER GNMORE, GMORE
PARAMETER (GNMORE=0, GMORE=1)

C C text alignment: horizontal normal, left, center, right
INTEGER GAHNOR, GALEFT, GACENT, GARITE
PARAMETER (GAHNOR=0, GALEFT=1, GACENT=2, GARITE=3)

C C text alignment: vertical normal, top, cap, half, base, bottom
INTEGER GAVNOR, GATOP, GACAP, GAHALF,
* GABASE, GABOTT
PARAMETER (GAVNOR=0, GATOP=1, GACAP=2, GAHALF=3,
* GABASE=4, GABOTT=5)

C C text path: right, left, up, down
INTEGER GRIGHT, GLEFT, GUP, GDOWN
PARAMETER (GRIGHT=0, GLEFT=1, GUP=2, GDOWN=3)

C C text precision: string, character, stroke
INTEGER GSTRP, GCHARP, GSTRKP
PARAMETER (GSTRP=0, GCHARP=1, GSTRKP=2)

C C type of returned values: set, realized
INTEGER GSET, GREALI
PARAMETER (GSET=0, GREALI=1)

C C update state: not-pending, pending
INTEGER GNPEND, GPEND
PARAMETER (GNPEND=0, GPEND=1)

C C vector/raster/other: type vector, raster, other
INTEGER GVECTR, GRASTR, GOTHWK
PARAMETER (GVECTR=0, GRASTR=1, GOTHWK=2)

C C visibility: invisible, visible
INTEGER GINVIS, GVISI
PARAMETER (GINVIS=0, GVISI=1)

C C workstation category: Output, Input, Output+Input, Workstation
C Independent Segment Storage, Metafile Output, Metafile Input
INTEGER GOUTPT, GINPUT, GOUTIN, GWISS,
* GM0, GMI
PARAMETER (GOUTPT=0, GINPUT=1, GOUTIN=2, GWISS=3,
* GM0=4, GMI=5)

C C workstation state: inactive, active
INTEGER GINACT, GACTIV  
PARAMETER (GINACT=0, GACTIV=1)

C list of GDP attributes: polyline, polymarker, text, fill area
INTEGER GPLATT, GPMATT, GTXATT, GFAATT,
  * GEDATT
PARAMETER (GPLATT=0, GPMATT=1, GTXATT=2, GFAATT=3,
  * GEDATT=4)

C line type: solid, dash, dot, dash-dot
INTEGER GLSOLI, GLDASH, GLDOT, GLDASD
PARAMETER (GLSOLI=1, GLDASH=2, GLDOT=3, GLDASD=4)

C marker type: '.', '+', '*', 'o', 'x'
INTEGER GPOINT, GPLUS, GAST, GOMARK,
  * GXMARK
PARAMETER (GPOINT=1, GPLUS=2, GAST=3, GOMARK=4,
  * GXMARK=5)

C For use in Inquiry Functions returning both Current & Requested Values
INTEGER GCURVL, GRQSVL
PARAMETER (GCURVL=0, GRQSVL=1)

C
Appendix E: GKSGRAL Workstation Types

The include file GKS$GTSDEV.INC contains the symbolic names and codes for the various Workstation Types available in GKSGRAL and GKSGRAL-3D. It is reproduced below for reference. However, it should be noted that the version currently available on a particular machine may include later changes and corrections. To find how it should be accessed on a particular system check the index entry for 'include files'.

C GKS$GTSDEV.INC
C
C GTS-GRAL GKS and GKS-3D Workstation Types
C
C (07.11.88, new PostScript orientation and colour)
C (22.12.88, Add QMS Lasergrafix plotter)
C (31.05.89, Add VT240/340, MX7000, correct HP550P4, etc.
C Add special workstation for Pericom MX series)
C (06.10.89, Correct VE27A3 and VE27A4)
C (27.02.90, Add VT240M and FALCO)
C
C ** Workstation Independent Segment Storage **
C
INTEGER WISS
PARAMETER (WISS=3)
C
C ** Metafile Output/Input **
C
C GKS-2D: Metafile Appendix E 2D Output/Input
INTEGER MAE2DO
PARAMETER (MAE2DO=4)
INTEGER MAE2DI
PARAMETER (MAE2DI=5)
C
C GKS-3D: Metafile Appendix E 2D Output/Input
C (** May not yet be installed **)
INTEGER MAE3DO
PARAMETER (MAE3DO=0)
INTEGER MAE3DI
PARAMETER (MAE3DI=0)
C
C GKS Appendix E 2D Output for GKS-3D
INTEGER MAEXDO
PARAMETER (MAEXDO=10201)
C
C Computer Graphics Metafile !NOT YET SUPPORTED!
INTEGER CGMO
PARAMETER (CGMO=0)
C
INTEGER CGMI
PARAMETER (CGMI=0)
C
C ** APOLLO Workstations **
C N.B. The Window size may be altered by editing
C the file: 'gks_characteristic' (GPR driver) or
C the file: 'gks_workstations.config' (GSR driver)
C ('GPXXX2' corresponds to the second line, etc.)
C
C GPr interface, models DN300, DN3000, Monochrome
INTEGER GP30M2
PARAMETER (GP30M2=10002)

C GPr interface, model DN3000, Colour
INTEGER GP30C4
PARAMETER (GP30C4=10004)

C GPr interface, models DN550, DN660, Colour
INTEGER GP55C3
PARAMETER (GP55C3=10003)

C GSR interface, all models
C (eight simultaneous workstations/windows are provided)
INTEGER GSR001
PARAMETER (GSR001=9701)
INTEGER GSR002
PARAMETER (GSR002=9702)
INTEGER GSR003
PARAMETER (GSR003=9703)
INTEGER GSR004
PARAMETER (GSR004=9704)
INTEGER GSR005
PARAMETER (GSR005=9705)
INTEGER GSR006
PARAMETER (GSR006=9706)
INTEGER GSR007
PARAMETER (GSR007=9707)
INTEGER GSR008
PARAMETER (GSR008=9708)

C ** FALCO terminals **
C
INTEGER FALCO
PARAMETER (FALCO=7878)

C ** PERICOM terminals **
C
INTEGER GRAFPAC PG7800
PARAMETER (PG7800=7878)

C Monterey MG600
INTEGER MG600
PARAMETER (MG600=7800)

C Monterey MX2000
INTEGER MX2000
PARAMETER (MX2000=221)

C Monterey MX7000
INTEGER MX7000
PARAMETER (MX7000=221)

C Monterey MX8000
INTEGER MX8000
PARAMETER (MX8000=227)
** TEKTRONIX terminals **

** TEK 4010 **
\[
\text{INTEGER T4010} \\
\text{PARAMETER (T4010=101)}
\]

** TEK 4014 **
\[
\text{INTEGER T4014} \\
\text{PARAMETER (T4014=101)}
\]

** TEK 4014 with Enhanced Graphics Option **
\[
\text{INTEGER T4015} \\
\text{PARAMETER (T4015=103)}
\]

** TEK 4107/4207 **
\[
\text{INTEGER T4107} \\
\text{PARAMETER (T4107=121)} \\
\text{INTEGER T4207} \\
\text{PARAMETER (T4207=121)}
\]

** TEK 4109/4209 **
\[
\text{INTEGER T4109} \\
\text{PARAMETER (T4109=122)} \\
\text{INTEGER T4209} \\
\text{PARAMETER (T4209=122)}
\]

** TEK 4111 **
\[
\text{INTEGER T4111} \\
\text{PARAMETER (T4111=123)}
\]

** TEK 4113 **
\[
\text{INTEGER T4113} \\
\text{PARAMETER (T4113=125)}
\]

** TEK 4115 **
\[
\text{INTEGER T4115} \\
\text{PARAMETER (T4115=127)}
\]

** VAXstation 2000 and GPX **

N.B. The Window size may be altered by editing the file: 'window.dat'.

Six workstations are available.

via UIS interface
\[
\text{INTEGER VXUIS} \\
\text{PARAMETER (VXUIS=8601)} \\
\text{INTEGER VXUIS1} \\
\text{PARAMETER (VXUIS1=8601)} \\
\text{INTEGER VXUIS2} \\
\text{PARAMETER (VXUIS2=8602)} \\
\text{INTEGER VXUIS3} \\
\text{PARAMETER (VXUIS3=8603)} \\
\text{INTEGER VXUIS4} \\
\text{PARAMETER (VXUIS4=8604)} \\
\text{INTEGER VXUIS5} \\
\text{PARAMETER (VXUIS5=8605)} \\
\text{INTEGER VXUIS6}
\]
PARAMETER (VXUIS6=8606)

C
C via X-Windows interface !NOT YET SUPPORTED!
INTEGER VXXW
PARAMETER (VXXW=0)

C ** VT-240/340 (DEC) **
C
INTEGER VT240
PARAMETER (VT240=1020)
INTEGER VT240M
PARAMETER (VT240M=1021)
INTEGER VT340
PARAMETER (VT340=1030)

C ** PostScript **
C 4th character: Portrait or Landscape Orientation (P/C)
C 5th character: Colour or Monochrome (C/M)
C
INTEGER PSTPC
PARAMETER (PSTPC=12201)
INTEGER PSTLC
PARAMETER (PSTLC=12202)
INTEGER PSTPM
PARAMETER (PSTPM=12203)
INTEGER PSTLM
PARAMETER (PSTLM=12204)

C The following two parameters are kept for compatibility
INTEGER PSTSCR
PARAMETER (PSTSCR=12201)
INTEGER PSTSCL
PARAMETER (PSTSCL=12202)

C ** QMS Lasergrafix **
C 4th character: Portrait or Landscape Orientation (P/C)
C 5th character: Upper Box or Lower Box (U/L)
C
INTEGER QMSPU
PARAMETER (QMSPU=13001)
INTEGER QMSLU
PARAMETER (QMSLU=13002)
INTEGER QMSPL
PARAMETER (QMSPL=13003)
INTEGER QMSLL
PARAMETER (QMSLL=13004)

C ** IBM PC (From VAX via Kermit) ** !NOT SUPPORTED AT CERN!
C
INTEGER IBMPC
PARAMETER (IBMPC=171)

C ** VERSATEC Plotters **
C (Available via Metafile Interpretation Service)
C
C Versatec 11 inch roll (V80)
INTEGER VE80
PARAMETER (VE80=16001)
Appendix E. GKSGRAL Workstation Types

C
C Versatec 22 inch roll
INTEGER VE8222
PARAMETER (VE8222=16002)
C
C Versatec 36 inch roll
INTEGER VE8236
PARAMETER (VE8236=16003)
C
C Versatec A3 Colour cut sheet (VE2700)
INTEGER VE27A3
PARAMETER (VE27A3=16602)
C
C Versatec A4 Colour cut sheet (VE2700)
INTEGER VE27A4
PARAMETER (VE27A4=16601)
C
C ** XEROX Laser Printers **
C (Available via Metafile Interpretation Service)
C
C XEROX 87/9700 and 4050 at 300 Points Per Inch
INTEGER X9730
PARAMETER (X9730=16010)
C
C XEROX 87/9700 and 4050 at 150 Points Per Inch
INTEGER X9715
PARAMETER (X9715=16011)
C
C XEROX 2700 300 Points Per Inch
INTEGER X2730
PARAMETER (X2730=16012)
C
C XEROX 2700 150 Points Per Inch
INTEGER X2715
PARAMETER (X2715=16013)
C
C XEROX 2700 75 Points Per Inch
INTEGER X2775
PARAMETER (X2775=16014)
C
C XEROX 4045 300 Points Per Inch
INTEGER X4030
PARAMETER (X4030=16015)
C
C XEROX 4045 150 Points Per Inch
INTEGER X4015
PARAMETER (X4015=16016)
C
C ** IBM APA6670 and 3812 Laser Printers **
C (Available via Metafile Interpretation Service)
C
INTEGER A6670
PARAMETER (A6670=16017)
C
C ** HEWLETT-PACKARD Plotters **
C (Replace leading '7' by 'H' or 'R' for Roll Paper)
C (Add 'L'/ 'P' for Landscape/Portrait)
C (Add paper size index: A0, A1, A2, A3, A4)
C
C Model 7475 cut paper, Portrait, A4
INTEGER H475P4
PARAMETER (H475P4=4011)
C Model 7475 cut paper, Landscape, A4
INTEGER H475L4
PARAMETER (H475L4=4012)
C Model 7475 cut paper, Portrait, A3
INTEGER H475P3
PARAMETER (H475P3=4013)
C Model 7475 cut paper, Landscape, A3
INTEGER H475L3
PARAMETER (H475L3=4014)
C Model 7550 cut paper, Portrait, A4
INTEGER H550P4
PARAMETER (H550P4=4121)
C Model 7550 cut paper, Landscape, A4
INTEGER H550L4
PARAMETER (H550L4=4122)
C Model 7550 cut paper, Portrait, A3
INTEGER H550P3
PARAMETER (H550P3=4123)
C Model 7550 cut paper, Landscape, A3
INTEGER H550L3
PARAMETER (H550L3=4124)
C Model 7585 cut paper, Portrait, A4
INTEGER H585P4
PARAMETER (H585P4=4221)
C Model 7585 cut paper, Landscape, A4
INTEGER H585L4
PARAMETER (H585L4=4222)
C Model 7585 cut paper, Portrait, A3
INTEGER H585P3
PARAMETER (H585P3=4223)
C Model 7585 cut paper, Landscape, A3
INTEGER H585L3
PARAMETER (H585L3=4224)
C Model 7585 cut paper, Portrait, A2
INTEGER H585P2
PARAMETER (H585P2=4225)
C Model 7585 cut paper, Landscape, A2
INTEGER H585L2
PARAMETER (H585L2=4226)
C Model 7585 cut paper, Portrait, A1
INTEGER H585P1
PARAMETER (H585P1=4227)
C Model 7585 cut paper, Landscape, A1
INTEGER H585L1
PARAMETER (H585L1=4228)
C Model 7585 cut paper, Portrait, A0
INTEGER H585P0
PARAMETER (H585P0=4229)
C Model 7585 cut paper, Landscape, A0
INTEGER H585L0
PARAMETER (H585L0=4230)
C Model 7586 cut paper, Portrait, A4
Appendix E. GKSGRAL Workstation Types

INTEGER H586P4
PARAMETER (H586P4=14401)

C Model 7586 cut paper, Landscape, A4
INTEGER H586L4
PARAMETER (H586L4=14402)

C Model 7586 cut paper, Portrait, A3
INTEGER H586P3
PARAMETER (H586P3=14403)

C Model 7586 cut paper, Landscape, A3
INTEGER H586L3
PARAMETER (H586L3=14404)

C Model 7586 cut paper, Portrait, A2
INTEGER H586P2
PARAMETER (H586P2=14405)

C Model 7586 cut paper, Landscape, A2
INTEGER H586L2
PARAMETER (H586L2=14406)

C Model 7586 cut paper, Portrait, A1
INTEGER H586P1
PARAMETER (H586P1=14407)

C Model 7586 cut paper, Landscape, A1
INTEGER H586L1
PARAMETER (H586L1=14408)

C Model 7586 cut paper, Portrait, A0
INTEGER H586P0
PARAMETER (H586P0=14409)

C Model 7586 cut paper, Landscape, A0
INTEGER H586L0
PARAMETER (H586L0=14410)

C Model 7586 Roll paper, Portrait, A4
INTEGER R586P4
PARAMETER (R586P4=14411)

C Model 7586 Roll paper, Landscape, A4
INTEGER R586L4
PARAMETER (R586L4=14412)

C Model 7586 Roll paper, Portrait, A3
INTEGER R586P3
PARAMETER (R586P3=14413)

C Model 7586 Roll paper, Landscape, A3
INTEGER R586L3
PARAMETER (R586L3=14414)

C Model 7586 Roll paper, Portrait, A2
INTEGER R586P2
PARAMETER (R586P2=14415)

C Model 7586 Roll paper, Landscape, A2
INTEGER R586L2
PARAMETER (R586L2=14416)

C Model 7586 Roll paper, Portrait, A1
INTEGER R586P1
PARAMETER (R586P1=14417)

C Model 7586 Roll paper, Landscape, A1
INTEGER R586L1
PARAMETER (R586L1=14418)

C Model 7586 Roll paper, Portrait, A0
INTEGER R586P0
PARAMETER (R586P0=14419)

C Model 7586 Roll paper, Landscape, A0
INTEGER R586LO
PARAMETER (R586LO=14420)

C
Appendix F: GKSGRAL Workstation Description Tables

The following consists of abridged versions of the most commonly used Workstation Description Tables for the GTS-GRAL workstation drivers. A copy of the complete set is kept in the UCO, and up-to-date machine-readable versions are available via the FIND_GKS command.

Note that the include file GTSDEV (see Appendix on Page 104) defines mnemonic names for the workstation type codes, and users are advised to use these rather than the integer codes themselves. All workstations support the following software fonts, examples of which are illustrated in 8.1:

- proportional spaced: font -1 to -11, precision 2
- proportional spaced italics: font -101 to -111, precision 2
- mono-spaced: font -201 to -211, precision 2
- mono-spaced italics: font -301 to -311, precision 2
- Greek: font -13, -113, -213, -313
- solid filled font: font -51, -151, -251, -351

In addition, the CERN version of GKSGRAL has defined 24 hatch styles which are identical on all workstations (indices -101 to -124); see 8.2.

F.1 APOLLO

workstation type: Both GPR and GSR interfaces
- GPR: 10002/10003/10004
- GSR: 9701-9708 (allows multiple workstations)
- See Appendix on Page 104

GKS Level: GPR: 2b and GSR: 2b (with sample; 2c for next release)

max. display space: 0.275*0.263 metres

device specific line types: none

user definable line types: none

device specific marker types: none

hardware characters: font 1, precision 0/1, character height 7mm

The GSR driver has access to all the hardware fonts listed in the file 'gks_fonts.config'. They are accessed via font indices -1, -2...

DIN 66003: font 1, precision 2

software characters: see introduction to this Appendix

CERN-defined hatch styles: -101 to -124

device specific hatch styles: -1 to -8

pattern: none

Colour Table (10001/2): Monochrome
 (10003): 256 colours, indices 240-255 fixed
(10004): 16 colours, indices 14,15 fixed

Pre-defined indices (0-7): background (White), foreground (Black), Red, Green, Blue, Yellow, Magenta, Cyan

Colour Table (97xx): up to 256 depending on hardware configuration, of which 16 are pre-defined.

segment priority: supported

segment highlighting: not supported

locator device 1: mouse

locator Prompt/Echo 1: tracking cross

locator Prompt/Echo 2: cross-hair

locator Prompt/Echo 3: tracking cross

locator Prompt/Echo 4: rubber band

locator Prompt/Echo 5: rectangle

locator trigger: left button

locator break: right button

stroke device 1: mouse

stroke Prompt/Echo 1,3: marker at each point

stroke max. buffer size: 64

stroke trigger: middle button

stroke break: right button

stroke enter new point: left button

stroke skip backward: cursor left

stroke skip forward: -

stroke delete last point: <BACKSPACE>

stroke delete current point: <DELETE>

stroke toggle insert: <INSERT MARK>

valuator Prompt/Echo 1,3: digital representation

valuator def. initial value: 0.0

valuator def. limits: 0., 1.

valuator trigger: <CR>

valuator break: <ESC>

valuator skip backward: -

valuator skip forward: -

valuator delete previous character: <BACKSPACE>

valuator delete current character: <DELETE>

valuator toggle insert: <INSERT MARK>
choice Prompt/Echo 1,3: menu
choice trigger: numerical character <CR> mouse
choice break: <CR>
pick device: mouse
pick Prompt/Echo 1
pick trigger: left button
pick break: right button
string Prompt/Echo 1,2: display current string
string max. buffer size: 72
string trigger: <CR>
string break: <ESC>
string enter new character: alphanumeric character
string skip backward: <-
string skip forward: -¿
string delete previous character: <BACKSPACE>
string delete current character: <CHAR DELETE>
string toggle insert: <INSERT MARK>
window size: The size of the actual window is looked for in a descriptor file gks_characteristic (for GPR) and gks_workstations.config (for GSR). Private versions of these files may be installed in the user's naming directory, otherwise default versions are taken. The format of gks_characteristic is 5I5 and that of gks_workstations.config is 7I5. For gks_characteristic the first two numbers are the position in raster units of the top left-hand corner of the window, and the second two numbers are the width and height. The file contains six lines corresponding to the workstation types 10001...10006. For gks_workstations.config the first number is the workstation type, and the next four numbers are the same as above. The seventh number is a switch (0/1) to control whether or not the graphics windows have a border.

F.2 HP PLOTTERS

workstation type: See Appendix on Page 104
GKS Level 2a
max. display space: see GTS-GRAL manual
device specific line types: -1 to -13
user definable line types: none
line-width: 0.3 mm to ymax
special marker types: -1 to -96 except -27 (printable characters)
user definable marker types: none
minimum marker size: 0.3 mm
nominal marker size: 5 mm
maximum marker size: max. y dimension
hardware characters: font 1, precision 0/1, character height 1. to ymax
HP fonts: negative value of HP-font - 20
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
device dep. hatch styles: -1 to -36
pattern: none

Connection id. On-line: 1 - 99
Off-line: 101 - 199 (Generates a file with the name: PL<WTYP>.PLT)

Plotter Settings

The device driver expects that the plotter is switched to EXPAND mode. In this mode only the pen can be moved as close as possible to the physical edges of the paper. If you are using RS232 serial interface, be sure that the HARDWARE/MONITOR switch at the rear is set to 'HARDWARE'. If the handshake between host and plotter does not work correctly, data can be lost.

F.3 MEGATEK 7xxx/33xx series

workstation type: 63000; Changes have been made to the driver so that the workstation type 63000 may be used for any Megatek hardware configuration (except in the case where two displays are to be run simultaneously on a single controller). This is performed by reading information stored in files that are created by the WANDGEN utility, and which are accessed via the logical names WAND$CONFIG and WAND$SYSTEM. Information on these files will be found in the WAND Installation Guide, Section 4.

GKS Level 2c (This is a 3D driver for VMS only)
max. display space: 0.34*0.34 metres (4096*4096 pixels)
device specific line types: none
Nominal/min/max line width: 0.5/0.5/0.5 mm
GTS-GRAL specific marker types: none
Nominal/min/max marker size: 6.0/1.5/12.0 mm (8 sizes)
hardware characters: font 1, precision 0, 1
hardware character heights: 8 (1.5 to 12 mm)
hatch styles: 0=hollow, 1=solid; pattern and hatch not supported.
pattern: not supported
number of colours: 16 simultaneous out of 4096
- Default: background, foreground, red, green, blue, yellow, magenta, cyan, dark red, orange, brown, light green, olive, light violet, rose.
Appendix F. GKS/GRAL Workstation Description Tables

segment priority: not supported
segment highlighting: supported
locator device 1, 2: tablet puck
locator device 3: joystick
- prompt/echo 1, 2 by crosshair, 3 by cross, 4 by rubber band, 5 by rectangle.
- -11..-15 provide closed loop control of viewing transformations (see below).
locator trigger: puck keys 1..4; joystick button
locator break: <ESC>
stroke device 1..3: tablet puck
stroke max buffer size: 64
stroke Prompt/Echo 1, 3: marker at each point
stroke trigger: <BLANK>; lower puck key
stroke break: <ESC>
stroke enter new point: <CR>; upper puck key
stroke skip backward: <CTRL>L; left puck key
stroke skip forward: <CTRL>R; right puck key
stroke delete last point: <CTRL>H
stroke delete current point: <DEL>
stroke toggle insert: <CTRL>A
valuator device 1..3: numeric keys
valuator Prompt/Echo 1, 3: Digital representation
valuator def. initial value: 0.5
valuator def. limits: 0.0, 1.0
valuator trigger: <CR>
valuator break: <ESC>
valuator skip backward: <CTRL>L
valuator skip forward: <CTRL>R
valuator delete last character: <CTRL>H
valuator delete current character: <DEL>
valuator toggle insert: <CTRL>A
choice device 1, 2, 3: Function keys; Puck; Joystick
choice Prompt/Echo 1, 3: numbered menu (1); menu (3)
choice trigger: Function Keys; Puck keys; Joystick button
choice break: <ESC>
pick device 1..3: Tablet Puck
F.3. MEGATEK 7xxx/33xx series

pick Prompt/Echo 1: crosshair
pick trigger: puck keys
pick break: <ESC>
string device 1..3: keyboard
string Prompt/Echo 1: display current string
string max. buffer size: 80
string trigger: <CR>
string break: <ESC>
string enter new character: alphanumeric character
string skip backward: <CTRL>L
string skip forward: <CTRL>R
string delete last character: <CTRL>H
string delete current character: <DEL>
string toggle insert: <CTRL>A

Setup: The hardware configuration is assumed to include a tablet and/or joystick plus a HCRST3 3D hardware transformation unit. Parameters may be set in the routine GKM9ZZ specifying the following defaults: driver interface (Parallel=GX or DR11W=MG), Raster or Stroke device, Monochrome or Colour, Display List size, availability of 3D surface processor, screen size, allocation of Display List memory if several screens are available. Note that if the interface card is specified as being MG, then one can use a GX interface simply by creating Logical Assignments of MG0..8 to GX0..8 (or vice versa).

F.3.1 Control of Viewing Transformations

It is possible to produce closed loop control by the user of the viewing transformations without passing through the application program. This is achieved via use of locator prompt/echo types and the use joystick or puck as follows:

- Locator Prompt/Echo types -11, -12, and -13 produce rotations matrices corresponding to rotation about two axes which are applied in VRC before the view mapping matrix. The axes are $y+x$ (-11), $x+z$ (-12), and $y+z$ (-13).
- Locator Prompt/Echo -14 enlarges or reduces ('scales' or 'zooms') the view window in VRC symetrically by up to $\pm10$ or $1/10$ times.
- Locator Prompt/Echo -15 shifts (pans) the view window in VRC by up to $\pm1.0$.

To start closed loop interaction the application sets the mode of the locator device via GSLCM and then issues a request locator (GRQLC3), or sets the device into event mode. Dynamic modification is then enabled by pressing the joystick button or puck key and moving the joystick or puck, depending on which physical device was selected. Interaction is terminated by releasing the key. Only one viewing transformation can be modified at a time, and the view index is specified by calling GINLC3. The BREAK action <ESC> works normally, and resets the viewing transformation to the value before closed
loop control began. As closed loop control is performed in the driver, the application program must call GQVWR to inquire what is the current state of the viewing transformation. Closed loop control implies the following restrictions:

- Locator devices used in this mode may only be enabled in Request and Event mode. No entries are placed in the event queue.
- Whilst a viewing transformation is being modified it should not be set via GSVWR, nor should output be produced using it.
- Whilst a viewing transformation is being modified the workstation window and viewport should not be modified, nor should GIVW be called.
- A viewing transformation cannot be modified by more than one locator device at a time.

### F.4 PERICOM MONTEREY MG600, GRAPH PAC, and FALCO

workstation type: See Appendix on Page 104
- 7800 for MG600, MG200
- FALCO for Falco terminals
- 7878 for Pericom Graph Pac (old Pericom)
- For Macintosh (Versaterm) use the workstation type for Pericom Graph Pac (old Pericom).

**GKS Level 2b**
max. display space: 0.28*0.20 metres

device specific line types: -1, to -5. Note that -1 to -4 are software emulations of types 1 to 4. They are slower but give better quality.

user definable line types: -1001 to -n
device specific marker types: none

hardware characters: font 1, precision 0/1, character height 4-8mm
DIN 66003: font 1, precision 2

software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124

pattern: none
number of colours: 2
segment priority: supported on MG600
segment highlighting: not supported
locator device 1: arrow keys or mouse
locator Prompt/Echo 1, 2: cross-hair
locator Prompt/Echo 4: rubber band (MG600)
locator trigger: <CR> A-Z, a-z, 0-9
locator break: <LF>, <CTRL>E, <CTRL>e
stroke device 1: arrow keys or mouse
stroke Prompt/Echo 1: cross-hair, marker at each point
stroke max. buffer size: 64
stroke trigger: <BLANK>
stroke break: <LF>, <CTRL>E, <CTRL>c
stroke enter new point: <CR>, A-Z, a-z, 0-9
stroke skip backward: <CTRL>L, <CTRL>l
stroke skip forward: <CTRL>R, <CTRL>r
stroke delete last point: <CTRL>D, <CTRL>d
stroke delete current point: <DELETE>, <BS>
stroke toggle insert: <CTRL>I, <CTRL>i
valuator device: keyboard
valuator Prompt/Echo 1
valuator def. initial value: 0.5
valuator def. limits: 0., 1.
valuator trigger: <CR>
valuator break: <LF>, <CTRL>E, <CTRL>c
valuator skip backward: <CTRL>L, <CTRL>l
valuator skip forward: <CTRL>R, <CTRL>r
valuator delete last character: <CTRL>D, <CTRL>d
valuator delete current character: <DELETE>, <BS>
valuator toggle insert: <CTRL>I, <CTRL>i
choice device: keyboard
choice Prompt/Echo 1, 3: menu
choice trigger: numerical character <CR>
choice break: <LF>, <CTRL>E, <CTRL>c
pick device: mouse or arrow keys
pick Prompt/Echo 1: crosshair
pick trigger: A-Z, a-z, 0-9
pick break: <LF>, <CTRL>E, <CTRL>c
string device: keyboard
string Prompt/Echo 1: display current string
string max. buffer size: 72
string trigger: <CR>
string break: <LF>, <CTRL>E, <CTRL>c
string enter new character: alphanumeric character
string skip backward:<CTRL>L,<CTRL>l
string skip forward:<CTRL>R,<CTRL>r
string delete last character:<CTRL>D,<CTRL>d
string delete current character:<DELETE>,<BS>
string toggle insert:<CTRL>I,<CTRL>i

SETUP for MGxx graphics directory: H) GS/CAN sets terminal+display
- graphics mode: D) graphics state III (set by GKS automatically)
- graphics general P) normal text drawing
- graphics general Q) normal block fill
- graphics general M) CR Status Term

SETUP for Falco: a guide is available from the UCO

**F.5 POSTSCRIPT LASER PRINTERS**

workstation type: See Appendix on Page [104]
- 12201 Portrait -Colour
- 12202 Landscape -Colour
- 12203 Portrait -Monochrome
- 12204 Landscape -Monochrome

Connection id: If conid>200 then an output file will be written in Encapsulated PostScript format. See section on Page [39]

GKS Level 2a
max. display space: 0.1894*0.2794 metres (portrait)
max. display space: 0.2794*0.1894 metres (Landscape)
device specific line types: none
user definable line types: none
line-width scale factor: 1.
special marker types: none
user definable marker types: none
minimum marker size: 1.0 mm
nominal marker size: 2.0 mm
maximum marker size: max. y dimension
hardware characters: font 1 (Times-Roman), precision 0.1
- font -1 (Times-Roman)
- font -2 (Times-Italic)
- font -3 (Times-Bold)
- font -4 (Times-BoldItalic)
- font -5 (Helvetica)
- font -6 (Helvetica-Oblique)
- font -7 (Helvetica-Bold)
- font -8 (Helvetica-BoldOblique)
- font -9 (Courier)
- font -10 (Courier-Oblique)
- font -11 (Courier-Bold)
- font -12 (Courier-BoldOblique)
- font -13 (Greek and Symbols)

DIN 66003: font 1, precision 2

Software characters: see introduction to Appendix

Escape Character: The back-slash ('\') character within a string is used as an 'escape' character with the following effect:

\n    linefeed (newline)
\r    carriage return
\t    horizontal tab
\b    backspace
\f    form feed
\    back-slash
\(    left parenthesis
\)    right parenthesis
\ddd octal character code ddd

If the character following the back-slash is not one of the above then it is ignored. For more details see the PostScript Language Reference Manual.

CERN-defined hatch styles:-101 to -124

device dep. hatch styles:-1 to -13

pattern:none

available colours or intensities:16 (via shading)

pre-defined colour indices:8

Connection id. On-line:1 - 99
- Off-line:101 - 199 (Generates a file on FORTRAN logical unit (conid - 100))

VAX OPEN statement use CARRIAGECONTROL='LIST'
F.6 QMS Lasergrapfix 800, 1200

workstation type: See Appendix on Page 104
- 13001 Portrait -Upper Box
- 13002 Landscape -Upper Box
- 13003 Portrait -Lower Box
- 13004 Landscape -Lower Box

max. display space: 0.2000*0.2870 metres (13001)
- 0.2870*0.2000 metres (13002)
- 0.2700*0.4100 metres (13003)
- 0.4100*0.2700 metres (13004)
- 07874*11299 device units (13001)
- 11299*07874 device units (13002)
- 10629*16141 device units (13003)
- 16141*10629 device units (13004)

GKS Level 2a

device specific line types: -1 to -12
user definable line types: -1001 to -n
line-width scale factor: 1. to 15.
minimum line width factor: 0.08 mm
nominal line width factor: 0.08 mm
maximum line width factor: 2.62 mm

device specific marker types: not implemented
GTS-GRAL specific marker types: -101 to -114
user definable marker types: -1001 to -n
minimum marker size: 1.00 mm
nominal marker size: 5.00 mm
maximum marker size: 287, 200, 410, 270 mm (13001, 2, 3, 4)

hardware characters: font 1, precision 0/1 character, height 1. to 32767.
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix

CERN-defined hatch styles: -101 to -124

device dep. hatch styles: none
pattern: none

Connection id.
On-line: 1 - 99
- Off-line: 101 - 199 (Generates a file on FORTRAN logical unit (conid - 100))
F.7. TEKTRONIX 4010, 4012, 4014 and Compatible Terminals.

Device SETUP: see full WTD
VAX OPEN statement use CARRIAGECONTROL='NONE'

F.7 TEKTRONIX 4010, 4012, 4014 and Compatible Terminals.

workstation type: See Appendix on Page 104
- 101 for Tektronix 4010, 4014
- 103 for 4014 with enhanced graphics option

GKS Level 1.2b
max. display space: 0.36355*0.27359 metres
device specific line types: -5
user definable line types: -1001 to -n
special marker types: -101 to -114
user definable marker types: -1001 to -n
hardware characters: font 1, precision 0/1 character, height 4-8mm
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
pattern: none
locator device 1: thumbwheels
locator Prompt/Echo 1, 2: cross-hair
locator trigger: <CR>, A-Z, a-z, 0-9
locator break: <LF>, <CTRL>E, <CTRL>c
stroke device 1: thumbwheels
stroke Prompt/Echo 1: cross-hair
stroke max. buffer size: 64
stroke trigger: <BLANK>
stroke break: <LF>, <CTRL>E, <CTRL>c
stroke enter new point: <CR>, A-Z, a-z, 0-9
stroke skip backward: <CTRL>L, <CTRL>I
stroke skip forward: <CTRL>R, <CTRL>r
stroke delete last point: <CTRL>D, <CTRL>d
stroke delete current point: <DELETE>, <BS>
stroke toggle insert: <CTRL>I, <CTRL>i
valuator device: keyboard
valuator Prompt/Echo 1, 3: digital rep.
valuator def. initial value: 0.5
valuator def. limits: 0., 1.
valuator trigger: <CR>
valuator break: <LF>, <CTRL>E, <CTRL>c
valuator skip backward: <CTRL>L, <CTRL>I
valuator skip forward: <CTRL>R, <CTRL>r
valuator delete last character: <CTRL>D, <CTRL>d
valuator delete current character: <DELETE>, <BS>
valuator toggle insert: <CTRL>I, <CTRL>i

choice device 1: keyboard
choice Prompt/Echo 1,3
choice trigger: numerical character, <CR>
choice break: <LF>, <CTRL>E, <CTRL>c

pick device 1: thumbwheels
pick Prompt/Echo 1,3
pick trigger: A-Z,a-z,0-9
pick break: <LF>, <CTRL>E, <CTRL>c

string device: keyboard
string Prompt/Echo 1,2: display current string
string max. buffer size: 72
string trigger: <CR>
string break: <LF>, <CTRL>E, <CTRL>c
string enter new character: alphanumeric character
string skip backward: <CTRL>L, <CTRL>I
string skip forward: <CTRL>R, <CTRL>r
string delete last character: <CTRL>D, <CTRL>d
string delete current character: <DELETE>, <BS>
string toggle insert: <CTRL>I, <CTRL>i

Connection id. 1 - 99
- 101 - 199 (Generates a file on FORTRAN logical unit (conid - 100))

VAX OPEN statement use CARRIAGECONTROL='NONE'
F.8  TEKTRONIX 4100/4200 and PERICOM MX2000/7000/8000 series

workstation type: See Appendix on Page 104
- 121 for Tektronix 4107 or 4207
- 122 for Tektronix 4109
- 123 for Tektronix 4111
- 125 for Tektronix 4113
- 127 for Tektronix 4115
- 221 for Pericom MX2000 or MX7000
- 227 for Pericom MX8000

GKS Level 2b on VM/CMS, 2c on VMS
max. display space: 0.24*0.18 metres (4107, MX2000)
- 0.3573*0.268 metres (4109)
- 0.3854*0.2878 metres (4111)
- 0.3560*0.2670 metres (4113)
- 0.3430*0.2700 metres (4115)

device specific line types: -4 to -7
user definable line types: -1001 to -n
device specific marker types: -1, -6 to -10

hardware characters: font 1, precision 0/1 character, height 4-8mm
DIN 66003: font 1, precision 2

software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
pattern: textured -1 to -16, dithered -50 to -174

Colours: 16 colours out of 256, 8 pre-defined
Pre-defined indices (0-7): background (Black), foreground (White), Red, Green, Blue, Yellow, Magenta, Cyan.

segment priority: supported
segment highlighting: supported

locator device 1: joystick or arrow keys
- (mouse if SETUP "mousemap yes")

locator device 2, 3: VAX only: mouse or tablet with SETUP "mousemap no"

locator Prompt/Echo 1, 2: cross-hair
locator trigger: <CR>, A-Z, a-z, 0-9
locator break: <LF>, <CTRL>E, <CTRL>e

stroke device 1: joystick or arrow keys
Appendix F. GKSGRAL Workstation Description Tables

- (mouse if SETUP "mousemap yes")

stroke device 2, 3: VAX only: mouse or tablet with SETUP "mousemap no"

stroke Prompt/Echo 1: cross-hair, marker at each point
stroke max. buffer size: 64
stroke trigger: <BLANK>
stroke break: <LF>, <CTRL>E, <CTRL>c
stroke enter new point: <CR>, A-Z, a-z, 0-9
stroke skip backward: <CTRL>L, <CTRL>I
stroke skip forward: <CTRL>R, <CTRL>r
stroke delete last point: <CTRL>D, <CTRL>d
stroke delete current point: <DELETE>, <BS>
stroke toggle insert: <CTRL>I, <CTRL>i

valuator device: keyboard
valuator Prompt/Echo 1
valuator def. initial value: 0.5
valuator def. limits: 0., 1.
valuator trigger: <CR>
valuator break: <LF>, <CTRL>E, <CTRL>c
valuator skip backward: <CTRL>L, <CTRL>I
valuator skip forward: <CTRL>R, <CTRL>r
valuator delete last character: <CTRL>D, <CTRL>d
valuator delete current character: <DELETE>, <BS>
valuator toggle insert: <CTRL>I, <CTRL>i

choice device 1: keyboard
choice Prompt/Echo 1, 3: menu
choice trigger: numerical character <CR>
choice break: <LF>, <CTRL>E, <CTRL>c
pick device 1: joystick or arrow keys
- (mouse if SETUP "mousemap yes")
pick device 2, 3: VAX only: mouse or tablet with SETUP "mousemap no"
pick Prompt/Echo: 1
pick trigger: A-Z, a-z, 0-9
pick break: <LF>, <CTRL>E, <CTRL>c

string device: keyboard
string Prompt/Echo 1: display current string
string max. buffer size: 72
string trigger: <CR>
string break: LF, <CTRL>E, <CTRL>c
string enter new character: alphanumeric character
string skip backward: <CTRL>L, <CTRL>I
string skip forward: <CTRL>R, <CTRL>r
string delete last character: <CTRL>D, <CTRL>d
string delete current character: <DELETE>, <BS>
string toggle insert: <CTRL>I, <CTRL>i

SETUP
factory
- baud 4800 4800
- code ansi
- tabtype mxmouse (if you have a mouse instead of a tablet)
- mousemap yes
- nvsave

F.9 VAXstation GPX

workstation type: See Appendix on Page 104
- (6 open workstations are allowed: 8601-6)

GKS Level 2c
max. display space: 0.3358*0.2834 metres
device specific line types: none
user definable line types: none
device specific line widths: 15
device specific marker types: none
device specific marker sizes: 50
hardware characters: font 1, precision 0, character, height 4.5 to 20mm
- font -1 to -23, precision 0 correspond to UIS fonts 1 to 26 (1, 8, 14 excluded)
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
UIS hatch styles: -1 to -58
pattern: none
Colour Table: 8 index entries
Pre-defined indices (0-7): background (Black), foreground (White), Red, Green, Blue, Yellow, Magenta, Cyan

segment priority: supported
segment highlighting: supported (but not on monochrome screens)
locator device 1: mouse
locator Prompt/Echo 1,3: tracking cross
locator trigger: left mouse button
locator break: right mouse button
stroke device 1: mouse
stroke Prompt/Echo 1,3: cursor, marker at each point
stroke max. buffer size: 64
stroke trigger: middle button
stroke break: right button
stroke enter new point: left button
stroke skip backward: cursor left <-
stroke skip forward: cursor right ->
stroke delete last point: <BACKSPACE>
stroke delete current point: . (full stop)
stroke toggle insert: <INSERT>
valuator device: keyboard
valuator Prompt/Echo 1,3: digital representation
valuator def. initial value: 0.0
valuator def. limits: 0., 1.
valuator trigger: <CR>
valuator break: <ESC>
valuator skip backward: <-
valuator skip forward: ->
valuator delete previous character: <BACKSPACE>
valuator delete current character: .
valuator toggle insert: <INSERT>
choice device: keyboard or mouse
choice Prompt/Echo 1: numbered menu
choice Prompt/Echo 3: menu
choice trigger: numerical character <CR>
choice break: <ESC>
pick device: mouse
pick Prompt/Echo 1
pick trigger: left button
pick break: right button
string device: keyboard
string Prompt/Echo 1,2: display current string
string max. buffer size: 72
string trigger: <CR>
string break: <ESC>
string enter new character: alphanumeric character
string skip backward: <-
string skip forward: -¿
string delete previous character: <BACKSPACE>
string delete current character: <DELETE>
string toggle insert: <INSERT>

Cursor visibility: The cursor foreground must be set to non-black (i.e. not background) in the workstation set-up on colour stations in order to ensure visibility inside the GKS window.

Window size: The size of the actual window is looked for in a descriptor file WINDOW.DAT under the current working directory or, if the file is not found, in the directory defined by the logical name GKS_WINDOW.
The format of the file is 4F5.3, A20 as follows:

1. window size X in metres
2. window size Y in metres
3. window position X on screen in metres
4. window position Y on screen in metres
5. window title

For example: 0.15 0.15 0.20 0.15 GKS_WINDOW_1. The first line in the file corresponds to workstation type 8601, the second to 8602 etc.
See also the utility routines GCCDWN and GCSDWN for information on how to set the window size and title from within an application program (on Page 107).

F.10 VERSATEC

Workstation type: See Appendix on Page 104

GKS Level 2a

Max. display space: 5.0*0.265 metres (V80)
- 5.0*0.535 metres (8222)
- 5.0*0.580 metres (8224) !Not available at CERN!
Appendix F. GKS/GRAL Workstation Description Tables

- 5.0*0.893 metres (8236)
- 5.0*1.030 metres (8242) !Not available at CERN!
- 0.24*0.20 metres (Versacolor)

device specific line types: none
user definable line types: none
line-width scale factor: 1. to 5.
special marker types: -1 to -9
user definable marker types: none
minimum marker size: 0.127 mm
nominal marker size: 2 mm
maximum marker size: max. y dimension

hardware characters: font 1, precision 0/1 character, height 1. to ymax
- No lower case hardware characters

DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
device dep. hatch styles: -1 to -13
pattern: none
colours (Versacolor): 8 line colours
- 256 solid fill area colours (dithering)

F.11 VT240

workstation type: 1020, (1021 for Monochrome) see Appendix on Page 104

GKS Level 2b (VMS only)
max. display space: 0.218*0.138 metres (800*240 pix)
device specific line types: -4, -5, -7, -8
user definable line types: -1001 to -n
Nominal/min/max line width: 0.3/0.3/0.3 mm
GTS-GRAL specific marker types: -101 to -114
Nominal/min/max marker size: 5.0/0.3/138.0 mm (240 sizes)
hardware characters: font 1, precision 0/1
hardware character heights: 3 (3 to 9 mm)
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
pattern: not supported

number of colours: 4 simultaneous out of 64
- Default: background (Black), foreground (White), Red, Green

segment priority: supported

segment highlighting: not supported

locator device 1: arrow keys
- prompt/echo 1 and 2 by crosshair

locator device 2 and 3: arrow keys
- prompt/echo 4/5 by rubber band/rectangle

locator trigger: <CR>, <LF>, <ENTER>
locator break: <ESC>, x, X

stroke device 1, 2, 3: arrow keys
stroke max buffer size: 64

stroke Prompt/Echo 1: marker at each point
stroke trigger: <BLANK>
stroke break: <ESC>, x, X
stroke enter new point: <CR>, O, <LF>
stroke skip backward: <org
stroke skip forward: < or .
stroke delete last point: <BS>, <DEL>, b, B
stroke delete current point: d or D
stroke toggle insert: i, I

valuator device: keyboard (0-9, +, -, .., E, c)
valuator Prompt/Echo 1, 2, 3: Digital representation
valuator def. initial value: 0.0
valuator def. limits: 0.0, 1.0
valuator trigger: <CR>, <LF>, <ENTER>
valuator break: <ESC>, keypad -
valuator skip backward: cursor keys, keypad 4
valuator skip forward: cursor keys, keypad 6
valuator delete last character: <BS>, <DEL>
valuator delete current character: keypad .
valuator toggle insert: <PF4>, keypad 0

choice device 1: keyboard

choice Prompt/Echo 1, 3: menu
choice trigger: numerical character <CR>
choice break: <ESC>, keypad-
pick device 1, 2, 3: arrow keys
pick Prompt/Echo 1: crosshair
pick trigger: <CR>, <LF>, O
pick break: <ESC>, x, X
string device 1: keyboard
string Prompt/Echo 1: display current string
string max. buffer size: 72
string trigger: <CR>, <LF>, <ENTER>
string break: <ESC>, keypad-
string enter new character: alphanumeric character
string skip backward: cursor key, keypad 4
string skip forward: cursor key, keypad 6
string delete last character: <BS>, <DEL>
string delete current character: keypad.
string toggle insert: PF4, keypad 0

SETUP  The routine GK10AZ may be used to define the default configuration.
Current parameters are set to:
Local echo: off
Keypad mode: application keypad active on close workstation
Break: 100 milli-seconds
Echo text for Message and Choice: ANSI text type (not Graphic)

F.12  VT340

workstation type: 1030; see Appendix on Page 104
GKS Level 2b (VMS only)
max. display space: 0.240*0.145 metres (800*480 pix)
device specific line types: -1 to -5
user definable line types: -1001 to -n
Nominal/min/max line width: 0.3/0.3/0.3 mm
GTS-GRAL specific marker types: -101 to -114
Nominal/min/max marker size: 5.0/0.3/145.0 mm (480 sizes)
hardware characters: font 1, precision 0/1
hardware character heights: 17 (3 to 72.5 mm)
DIN 66003: font 1, precision 2

Software characters: see introduction to Appendix

CERN-defined hatch styles: -101 to -124

Pattern: not supported

Number of colours: 16 simultaneous out of 4096

- Default: background (black), foreground (white), red, green, blue, yellow, magenta, cyan, green, blue, yellow, magenta, cyan, foreground, red.

Segment priority: supported

Segment highlighting: not supported

Locator device 1: mouse, keys A1..A4

Locator device 2, 3: keys A1..A4

- Prompt/echo 1, 2 by crosshair, 3 by cross, 4 by rubber band, 5 by rectangle

Locator trigger: <CR>, <LF>, <ENTER>; Mouse B1

Locator break: <ESC>, x, X; keypad -; Mouse B3

Stroke device 1: keys A1..A4; mouse

Stroke device 2, 3: keys A1..A4

Stroke max buffer size: 64

Stroke Prompt/Echo 1, 3: marker at each point

Stroke trigger: <BLANK>; mouse B2

Stroke break: <ESC>, x, X; mouse B3 (2,3 also keypad -)

Stroke enter new point: <CR>, O, <LF>; mouse B1

Stroke skip backward: < or,

Stroke skip forward: > or.

Stroke delete last point: <BS>, <DEL>, (1 also b, B)

Stroke delete current point: d or D (2,3 also keypad .)

Stroke toggle insert: i, I (2,3 also keypad 0)

Valuator device 1, 2, 3: numeric keys

Valuator Prompt/Echo 1, 2, 3: Digital representation

Valuator def. initial value: 0.5

Valuator def. limits: 0.0, 1.0

Valuator trigger: <CR>, <LF>, <ENTER>

Valuator break: <ESC>, keypad -

Valuator skip backward: A3, keypad 4

Valuator skip forward: A4, keypad 6

Valuator delete last character: <BS>, <DEL>
valuator delete current character: keypad.
valuator toggle insert: <PF4>, keypad 0
choice device 1, 2: keyboard
choice device 3: mouse
choice Prompt/Echo 1, 3: numbered menu (1); menu (3)
choice trigger: numerical character; mouse B1
choice break: <ESC>, keypad -; mouse B3
pick device 1: keys A1..A4; mouse
pick device 2, 3: keys A1..A4
pick Prompt/Echo 1: crosshair
pick trigger: <CR>, <LF>, O; mouse B1 (2, 3 <ENTER>)
pick break: <ESC>, x, X; mouse B3 (2, 3 keypad -)
string device 1: keyboard
string Prompt/Echo 1: display current string
string max. buffer size: 72
string trigger: <CR>, <LF>, <ENTER>
string break: <ESC>, keypad -
string enter new character: alphanumeric character
string skip backward: cursor key, keypad 4
string skip forward: cursor key, keypad 6
string delete last character: <BS>, <DEL>
string delete current character: keypad.
string toggle insert: PF4, keypad 0

SETUP  The routine GK06AZ may be used to define the default configuration.
Current parameters are set to:
Local echo off
Keypad mode: application keypad active on close workstation
Break: 100 milli-seconds
Echo text for Message and Choice: ANSI text type (not Graphic)

If arrow keypad is not available then keys A1..A4 are defined as:

A1: Keypad 8
A2: Keypad 2
A3: Keypad 4
A4: Keypad 6
F.13  XEROX, APA 6670 and 3812 LASER PRINTERS

workstation type: only accessible via metafile interpreters

GKS Level 2a

max. display space:
- 0.2073*0.2946 metres (XEROX 8700 300 ppi)
- 0.2059*0.2926 metres (XEROX 8700 150 ppi)
- 0.0840*0.1192 metres (XEROX 2700 300 ppi)
- 0.1504*0.2127 metres (XEROX 2700 150 ppi)
- 0.2032*0.2899 metres (XEROX 2700 75 ppi)
- 0.1295*0.1822 metres (XEROX 4045 300 ppi)
- 0.2059*0.2926 metres (XEROX 4045 150 ppi)
- 0.1465*0.2142 metres (APA 6670 and IBM 3812)
- ppi = pixels per inch

device specific line types: none
user definable line types: none
line-width scale factor: 1. to 5.
special marker types: none
user definable marker types: none
minimum marker size: 0.127 mm
nominal marker size: 5 mm
maximum marker size: max. y dimension
hardware characters: none
DIN 66003: font 1, precision 2
software characters: see introduction to Appendix
CERN-defined hatch styles: -101 to -124
device dep. hatch styles: none
pattern: none
Appendix G: GKSGRAL Software Installation

Appendix G: GKSGRAL Software Installation

This appendix contains information on the installation of GKSGRAL on machines not supported by the CERN computer centre. Further details may be found in the installation guide for each system available in the full GKSGRAL manual [10].

Note

GKSGRAL and GKSGRAL-3D are proprietary software products for which CERN holds a site licence. Thus, they may be installed and used at CERN on any machines running one of the following systems: VM/CMS, UNIX, UNICOS, VMS, AEGIS, SINTRAN-3. Information on how to obtain a licence for use outside CERN may be found in Appendix on Page 144.

It is illegal to take copies of the software off the CERN site.

G.1 Library Configuration

Although most users will simply link to one or other of the GTS-GRAL libraries, it is perhaps as well to mention that there is a fairly straight-forward procedure to re-configure these libraries to add or omit drivers. In both GKSGRAL and GKSGRAL-3D all driver code is accessed via the routine GKDDLK. Thus, in order to change the configuration, it is only necessary to edit and re-compile this routine. Of course, the corresponding driver routines must be added to/deleted from the library. Where access to GKDDLK is not available, it is possible to use the non-shared library version and to link to dummy versions of the un-used drivers.

Some of the drivers may make use of low-level proprietary software, such as VERSAPLOT, which cannot be included in the library as a separate licence is required. Installations which configure drivers of this type, but which do not have the low level software, may wish to provide a dummy library in order to avoid a long list of unresolved external references.

G.2 VAX Installation

VAX implementations can be distributed within CERN without going through the Program Library if the target machine is connected to DECNET, and the following automated procedures have been written to carry out the installation:

VXCERN::CERN:[GKS.PRO.MGR]INSTALL_GKS.COM
VXCERN::CERN:[GKS.PRO.MGR]INSTALL_GKS_GPX.COM
VXCERN::CERN:[GKS.PRO.MGR]INSTALL_GKS_MEGATEK.COM
VXCERN::CERN:[GKS.PRO.MGR]INSTALL_GKS_SRC.COM
VXCERN::CERN:[GKS.PRO.MGR]INSTALL_GR.COM

The first one should be used for VAXes which support time-sharing graphics terminals, and the second for VAXstations booted with the UIS graphics interface. INSTALL_GKS_MEGATEK.COM deals with the special case of MEGATEK support, INSTALL_GKS_SRC.COM may be used to copy the source code for particular drivers, and INSTALL_GR.COM should be used to install the utilities GRVIEW, GRCONV and GRPLOT.

In all cases it is necessary to answer various questions during the installation. For example, these include the name of the root directory on the target node in which GKS is to be stored and the version (OLD,
PRO, or NEW) which is to be copied. To make GKS-related commands and Logical Names known at the system level system privileges are required to run the procedures. Further command files will be made available in the future to install various graphics utilities individually.

G.3 IBM Installation

Distribution for IBM VM/CMS is done on magnetic tape in CMS TAPE DUMP format and can be read using the REXX exec:

```rexx
/* REXX */
/*BATCH TAPES 1 */
/*BATCH PUN 400 */
/*BATCH TIME 4 */
TRACE C
'TAPE LOAD * * A ( EOT '
```

The first file on the tape, ’README INFO’, contains information on the tape’s contents, as well as installation execs and a guide to modifying drivers. New CERN products and example programs are continually being added as they are developed.

G.4 UNIX Installation

G.4.1 Sites with hardware for which object code is not available.

The following describes the first version of the installation procedure for sites starting from the GKS source code. This should be the case only for those institutes which are the first to order a licence to run the GTS-GRAL software under UNIX on a machine type which is not available at CERN, and for which no other institute has yet installed the software.

For CERN-supported systems other than UNIX, affiliated institutes are able legally to receive only the GTS-GRAL kernel and driver libraries, and the driver source, but NOT the kernel source. This requirement has been relaxed for UNIX in order to support machines not available at CERN, but in this case the institute must sign a source licence with GTS-GRAL, for which there will be no charge. (If compiled libraries are already available, then the distribution is as described in section on Page 141).

Once the software has been delivered and compiled, the institute is requested to act as a subsidiary distribution centre to CERN, so that the libraries may be made available to institutes which later request support for the same machine type. However, they should only distribute libraries if authorized to do so. CERN can make available example scripts to be used in order to build a tar distribution file.

The source code is made available in a tree stored in a tar file called `unix_src.tar`, which can be distributed either on tape or via a network (eg via FTP). Later, there may be a distinction made between the OLD, PROduction, and NEW versions.

The first thing to do is to create a root directory, for example, `/user/gts-root`, and set this as current. The tar file should then be copied to this root from tape or via a network, and then be unpacked:

```
mkdir /user/gts-root
(cp-or-ftp-or-mv unix_src.tar /user/gts-root/unix_src.tar)
cd /user/gts-root
tar -xf unix_src.tar
```
Appendix G. GKSGRAL Software Installation

So far the development has been done at CERN on an Apollo DN3000 under SR 10.2 using SYSTYPE 'sys5.3', and the tar file has been moved to CRAY X/MP and Y/MP machines, SEL Gould, and a DECstation RISC under Ultrix. In all cases both GKS and GKS-3D have been compiled successfully. The drivers tested have been the ones for PostScript, the 2D metafile driver, the Apollo GPR and GSR drivers, and the Tektronix 4107 (on the DECstation).

Any problems and/or modifications made to code or scripts should be carefully commented and sent to GRAPHICS@CERNVM.CERN.CH in order that they may be incorporated in later versions.

G.4.1.1 Tar file contents.

The root contains the directories bin, dmo, doc, gks, gks3d, and utl, plus the files:

- .login C shell login script
- .cshrc C shell new process script

bin contains command files needed to compile the libraries, etc. All scripts starting with 'c_' were written at CERN, and most of the others have been modified:

- startgks Script to define symbolic names (Bourne or C shell)
- startgks.apollo Apollo development version of startgks
- startgks.cray CRAY version of startgks
- startgks.ux Ultrix version of startgks
- c_system_rebuild Re-build GKS, GKS-3D, and driver libraries
- c_gkslib_create Create GKS library
- c_gksdriv_create Create Driver library
- c_gkslib3d_create Create GKS-3D library
- compcc Compile all C routines (*.c) in a directory and insert them into a library. (Uses the symbol '$ccomp')
- compf Compile all Fortran routines (*.f) in a directory and insert them into a library. (Uses the symbol '$fortran')
- c_fort Apollo only; this uses the Aegis Fortran compiler
- c_fix_include Utility to convert format of INCLUDE statement
- c_gks_comp_single Compile routine and replace in a library
- c_gkzCompile Utility to test compile gkz routines only
- lkgks2d Example script to link a GKS application
- lkgks3d Example script to link a GKS-3D application
- rencmn Utility to convert .cmn file names to upper case
- reninc Utility to convert .inc file names to upper case
- renfor Utility to convert .for file names .f file names
- renftn Utility to convert .ftn file names .f file names
G.4. UNIX Installation

doc contains the files:

help.unix UNIX help file
help.changes The list of updates made to date
help.driver Some notes on how to write a driver
help.errors List of GKS Error codes

gks contains:

demo GTS-GRAL demo programs
drivers driver source
kernel kernel source with CERN mods
kernel.original original routines modified by CERN
fonts data files for software fonts
libs (empty) directory for libraries

gks3d contains:

demo GTS-GRAL demo programs
kernel kernel source with CERN mods
kernel.original original routines modified by CERN
exe data files for demos
libs (empty) directory for libraries

gks/drivers contains:

driver.list file defining which drivers to compile
gkx gkx utility routines with CERN mods
gkx.original original routines modified by CERN
gkz copy of relevant gkz.xxx directory
gkz.standard gkz machine-dependent routines with CERN mods
gkz.original original routines modified by CERN
gkz.apollo gkz routines for Apollo
gkz.cray gkz routines for CRAY
metafile 2D metafile driver for GKS-3D
tek4014 Tektronix 4014 driver
tek4107 Tektronix 4107 driver
vev9c Versatec driver
xxpscrip PostScript driver
Appendix G. GKSGRAL Software Installation

utl contains the files:

- gks_enum: Include file for GKS enumerated types
- gks_gtsdev: Include file for GTS-GRAL workstation type codes
- gks_implem.gts: Data file used by various GKSPACK routines
- gkspack.build: Script to unpack gkspack.car
- gkspack.car: GKSPACK 'CAR' file
- gkspack.f: GKSPACK Fortran source for UNIX systems
- gkspack_test.f: Simple GKSPACK test program
- grconv: Script used for metafile conversion
- gredit.f: Source of editor part of GRVIEW program
- grview: Script used for metafile viewing and editing
- grview.comp: Script used to build GRVIEW
- grview.f: Source of GRVIEW program

G.4.1.2 Procedure

1. Edit the .login and .cshrc to set the local value for $gkshome, etc. If the C shell is being used, and make sure they are executed. (If using the Bourne or Korn shells, then modify the files as necessary.)

2. Rename the most appropriate file to $gkshome/bin/startgks and edit it to set up local conditions. Then execute it, for example by typing 'source startgks' under the C shell. The points to watch for include:
   - Use of Bourne Shell or C Shell
   - Use of UNIX System V or BSD 4.2,
   - Local home directory definition (if not set in .login file)
   - Local Fortran and C command syntax
   - Local librarian. ie bld not ar, or special ar syntax. Note that if ar must be used under BSD without ranlib, then the use of ranlib must be uncommented in the files c_gkslib_create, c_gksdrv_create, and c_gkslib3d_create.
   - Ensure definitions of gkschXX are correct.

3. If necessary (eg under Ultrix, SEL-Gould, or perhaps other BSD systems) edit the files c_gks*create and c_gks_comp_single in order to remove the comment character in front of the ranlib command.

4. Process all Fortran files to change the format of the INCLUDE statement to whatever works locally. (They are delivered using VMS format.) This may be done with the script $gkstools/c_fix_include after setting the target working directory as current. Modify c_fix_include as necessary.

5. Replace the files in $gkshome/gks/drivers/gkz with the set most appropriate to the local system. It may be necessary to edit all the .c files in order to change the subroutine names to/from upper/lower case with/without a trailing underscore, etc. There exist already directories gkz.cray,
G.4. UNIX Installation

`gkz.apollo` and `gkz.standard` all under `$gkshome/gks/drivers` and, as delivered, `gkz` contains the files from `gkz.standard`. Make sure that you start with the most appropriate set of `gkz` files. Copy the tree using (eg) the command:

```
$gkstools/c_cpt $gkshome/gks/drivers/gkz.cray $gkshome/gks/drivers/gkz
```

Note that C routines for different machines have the following characteristics:

- **STANDARD**: Global symbolic names are lower case with trailing underscore.
- **CRAY**: Global symbolic names are upper case with no underscore.
- **APOLLO**: Global symbolic names are lower case with no underscore if Aegis Fortran compiler is used. (This is the CERN recommendation, and is backwards compatible from SR 10 to SR 9.7)

6 Edit the file `$gksdrivlist` to specify the drivers you wish to include. The file has one line per driver, with the format `dn name`. The first character is a literal `d` and is followed by an integer which numbers the lines. The second argument is the name of the sub-directory below `$gkshome/gks/drivers` containing the driver which is to be added.

7 Edit the routines `$gkshome/gks/kernel/gkddlk.f` and `$gkshome/gks3d/kernel/gkddlk.f` (they are not identical) to ensure that the drivers included in `$gksdrivlist` are known to GKS and GKS-3D.

8 Execute `$gkstools/c_system_rebuild` (and cross your fingers).

9 Check that the environment variable `GKS_FONTS` is defined correctly and is available to all users.

G.4.1.3 Problem areas

There may be problems in the following areas:

1 Although most compilers accept the INCLUDE statement, this is not defined in the Fortran 77 standard, and so its syntax may be vary on different machines.

2 The format of External Global Symbol Names is not defined anywhere to be a standard, even though most UNIX systems use lower case symbol names with a trailing underscore. This is crucial for the linking of routines written in different languages. (GKSGRAL uses the languages Fortran *and* C.)

3 Machine dependencies in some of the `gkz` routines. GTS-GRAL have tried to keep all system dependencies within these routines. If UNIX were really a standard then there should be nothing to change. Nevertheless, there are variations, (eg `gkzof` for opening files, etc).

4 Special features of local compilers.

5 Some BSD implementations seem to use ar without ranlib. In this case use of ranlib must be uncommented in the files `c_gkslib_create`, `c_gksdriv_create` and `c_gkslib3d_create`.

G.4.2 Sites with hardware for which object code is available.

If compiled libraries are already available, then the distribution is made via a tar file called `unix_lib.tar`. As this file contains binary libraries it must be compatible with the target machine. It may also be distributed by an institute affiliated to CERN, rather than by the CERN Program Library, if the target machine is not available on the CERN site.

The first thing to do is to create a root directory, for example, `/user/gts-root`, and set this as current. The tar file should then be copied to this root from tape or via a network, and then be unpacked:
Appendix G. GKSGRAL Software Installation

mkdir /user/gts-root
(cp-or-ftp-or-mv unix_lib.tar /user/gts-root/unix_lib.tar)
cd /user/gts-root
tar -xf unix_lib.tar .

If the driver library contains the required drivers, then the only thing that remains to be done is to define the environment variables gkshome, gkslib, gksdriv, gks3dlib, GKS_FONTS, etc. This is best done by modifying the file startgks, as described above. If there is a requirement to use a driver which is not included in the the driver library, then the mechanism to add a driver is described below.

G.4.2.1 Tar file contents.

The tar file contains the same directory structure described in section on Page 138, but the contents are modified slightly:

- bin may not contain those scripts used to compile the kernel.
- gks/kernel contains only gkddlk.f
- gks/libs contains the kernel and driver libraries
- gks3d/kernel contains only gkddlk.f
- gks3d/libs contains the 3D kernel library

G.4.2.2 Procedure to add drivers.

To add drivers to the driver library one follows a procedure similar to that given in section on Page 140 which describes how to build the libraries from a source distribution.

1 Edit the file $gksdrivlist to specify the drivers you wish to include. The file has one line per driver, with the format ‘dn name’. The first character is a literal ‘d’ and is followed by an integer which numbers the lines. The second argument is the name of the sub-directory below $gkshome/gks/drivers containing the driver which is to be added.

2 Execute $gkstools/c_gksdriv_create, which will compile the whole driver library. Alternatively, one can simply add the driver to the existing library via the commands:

```
cd $gkshome/gks/drivers/new_driver
$compile $gksdriv
$ccompile $gksdriv
```

3 Edit the routines $gkshome/gks/kernel/gkddlk.f and $gkshome/gks3d/kernel/gkddlk.f (they are not identical) to ensure that the drivers compiled above are known to GKS and GKS-3D. The routines must then be compiled and replaced in their respective libraries using the tool $gkstools/c_gks_comp_single:

```
c_gks_comp_single $gkshome/gks/kernel/gkddlk.f $gkslib
```

Those people who wish to write new drivers are basically on their own. They are advised to start from the source of an existing driver with features similar to the one they wish to write. However, a few points of interest are to be found in the file ‘help.drivers’.
G.5 Apollo Installation

No installation procedure is necessary for GKSGRAL or GKSGRAL-3D on APOLLO nodes connected to the standard DOMAIN network at CERN. For nodes on other DOMAINS the procedure is to obtain a cassette from the Program Library Office which contains the following trees written with wbak:

/cern/gks/pro
/cern/gks/new
/cern/gks/distribution

They can be restored with the command:

rbak -dev ct -f 1 -int -ld -all -as /cern/gks/pro
rbak -dev ct -f 2 -int -ld -all -as /cern/gks/new
rbak -dev ct -f 3 -int -ld -all -as /user/gts-root

For those institutes that wish to use the GKS libraries without change, it should be sufficient simply to restore either /cern/gks/pro or /cern/gks/new/. For those institutes which want to add drivers, or get access to the machine readable documentation, etc., it will be necessary to restore /cern/gks/distribution. This is equivalent to the root /user/gts-root referred to in the section on Page 137 and so should preferably be restored with ‘-as /user/gts-root’.
Appendix H: Distribution of GKSGRAL to Affiliated Institutes

As part of the agreement signed between CERN and GTS-GRAL, GTS-GRAL contracted to allow CERN to distribute the GKSGRAL and GKSGRAL-3D packages to affiliated institutes (in Europe and worldwide) on payment by the institute of a nominal embedded-licence fee. This licence allows the software to be used only in experiments conducted in collaboration with CERN. This means that all members of a CERN collaboration should be able to use a common implementation of GKS/GKS-3D.

Since signing the original contract, GTS-GRAL have agreed to extend the scope of the embedded licence to include the use of the software for any HEP experiments, not only those performed at CERN, on payment of an additional sum. However, the very low fee agreed with GTS-GRAL does not entitle the packages to be made available for general applications outside of HEP, and institutes are advised that they have both legal and moral obligations to take a standard licence (less 25% if they wish to do this.

In addition, a further extension to the original contract now provides CERN with the right to print and distribute GTS-GRAL User Manuals to affiliated institutes holding GTS-GRAL licences. This means that both software distribution and manuals may be obtained from CERN. CERN will charge the institute for printing plus a small handling charge.

H.1 Summary of Offer for an Embedded Licence

With the conditions of use specified above, an embedded licence entitles the holder to receive the following software:

1. The kernel of GKSGRAL and GKSGRAL-3D at the current revision level in object-code format for one of the following operating systems: MVS, VM/CMS, VMS, AEGIS, UNIX[1] or SINTRAN-3.

2. The drivers available at CERN in both object-code and source-code format. As of January 1989, CERN holds licences to distribute the following drivers for use on the systems listed in parentheses:

   - TEKTRONIX 4014 (MVS, VM/CMS, VMS, AEGIS, UNIX, SINTRAN-3)
   - TEKTRONIX 4107/4109 (MVS, VM/CMS, VMS, AEGIS, UNIX, SINTRAN-3)
   - TEKTRONIX 4235/6 (VMS) 3D driver[1] (in preparation)
   - Hewlett-Packard Plotters (MVS, VM/CMS, VMS, AEGIS, UNIX, SINTRAN-3)
   - VERSATEC Plotters via VERSAPLOT (MVS, VM/CMS, VMS, AEGIS, UNIX, SINTRAN-3)
   - PERICOM MG600 (VM/CMS, UNIX, VMS)
   - PERICOM MX2000/7000/8000 (VM/CMS, UNIX, VMS)
   - VAXstation (VMS-UIS)
   - APOLLO GSR and GSR (AEGIS/UNIX)
   - MEGATEK 33xx (VMS) 3D driver[1]
   - PostScript (VM/CMS, VMS, AEGIS, UNIX)

[1] The director responsible for computing, Dr. J. Thresher, has decided that it will be necessary to ask the initial users of the software on UNIX to contribute 3000.– CHF to the cost of extending the CERN licence. This is independent of the licencing arrangements made between the institute and GTS-GRAL.
## H.2 Pricing

- QMS LG(VMS)
- VT240 and VT340 (VMS)
- XEROX printers via EPIC (VM/CMS)
- IBM APA 6670 and 3812

3 The skeleton drivers in source-code format.
4 The source-code of the link routines necessary to re-configure the system with extra drivers.

CERN is entitled to distribute updates and additional drivers\(^3\) to institutes holding such a licence. However, additional software not available at CERN, may be purchased directly from GTS-GRAL with the following conditions:

1 A 35other than those already licenced at CERN.
2 A 50for drivers not already licenced at CERN.

A current list of products available from GTS-GRAL may be obtained by writing to the address given below. Over 50 computers are supported and a similar number of drivers exist.

### H.2 Pricing

The base prices for embedded licences are:

A: on IBM DM 28.000,—
B: on VAX DM 28.000,—
C: on APOLLO or Microvax DM 14.000,—
D: on UNIX for multi-user system DM 28.000,—
E: on UNIX for single-user system DM 14.000,—

of which the following percentages are payable per institute and operating system:

<table>
<thead>
<tr>
<th>Licences</th>
<th>Percentage of A, B...E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 49</td>
<td>10.0%</td>
</tr>
<tr>
<td>50 - 100</td>
<td>5.0%</td>
</tr>
<tr>
<td>more than 100</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

This implies that for the first 49 licences\(^3\) to be signed *world-wide*, the licencee will pay DM 2.800,— for a multi-user computer, and DM 1’400,— for single-user machines. In the case that the affiliated institute wishes to use the software to work on HEP experiments not performed at CERN, the licence fee is surcharged by 50to analyse data from a LEP experiment and an SLD or Gran Sasso experiment would cost DM 2.100,— instead of DM 1.400,— once the limit of 49 licences has been passed.

The prices for a full licence which would permit the software to be used for applications outside the domain of HEP are:

\(^2\)Institutes wishing to purchase extra drivers are advised to take a licence including access to the source in case modification are required, for example to enable the use of GCGTOA and GCATOG.

\(^3\)Since end 1990 the total number of licences has passed 100.
Appendix H. Distribution of GKSGRAL to Affiliated Institutes

<table>
<thead>
<tr>
<th>Software</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>GKSGRAL-3D for IBM</td>
<td>DM 28.000,–</td>
</tr>
<tr>
<td>GKSGRAL-3D for DEC/VMS</td>
<td>DM 28.000,–</td>
</tr>
<tr>
<td>GKSGRAL-3D for APOLLO</td>
<td>DM 14.500,–</td>
</tr>
<tr>
<td>GKSGRAL-2D for IBM</td>
<td>DM 19.250,–</td>
</tr>
<tr>
<td>GKSGRAL-2D for VAX</td>
<td>DM 19.250,–</td>
</tr>
<tr>
<td>GKSGRAL-2D for APOLLO</td>
<td>DM 9.950,–</td>
</tr>
</tbody>
</table>

Institutes are entitled to receive a 25% discount on these figures. (The pricing for MicroVAX is the same as for APOLLO.) If the institute wishes to have the software installed by GTS-GRAL, then the installation costs would be:

1. IBM (MVS,VM/CMS) DM 5.000,–
2. DEC/VAX (VMS) DM 5.000,–
3. APOLLO DM 2.500,–

H.3 Procedure

An institute wishing to obtain an embedded licence for the use of GTS-GRAL software must place an order with the company directly, who will deliver the manual and give CERN permission for distribution when the licence has been signed and on payment of the fee. The address to contact is:

GTS-GRAL GmbH
Friedberger Strasse 25
6100 Darmstadt
West Germany.

Tel. +49.6151.73090
Fax. +49.6151.77718

It is possible that in some countries it may be necessary to deal with a local GTS-GRAL distributor, although one may still contact GTS-GRAL in the first instance.

Once the licence has been signed and payed for, and CERN has received confirmation of this from GTS-GRAL, then it will be possible to obtain the software via the CERN Program Library Office. To order additional copies of GKSGRAL or GKSGRAL-3D manuals contact the Program Library.

The Electronic Mail address for the program library is: CERNLIB@CERNVM.CERN.CH

An attempt is being made to maintain a list of embedded-licence holders. Thus, the following information should be provided:

1. The address of the institute.
2. The date the contract was signed.
3. The name of the contact person(s). (Not the person who signed the contract, but the person responsible for installation and maintenance.)
4. The Electronic Mail address and telephone number by which they can be contacted.
5. The operating system for which the software is licenced.
6. The distribution medium required.
Note that for VM/CMS sites it will be necessary to discuss the installation procedure in more detail in order to resolve possible problems concerning different FORTRAN compilers, ASCII-to-EBCDIC conversion tables, and so forth. For UNIX sites installation may also be a problem if the target machine is not available at CERN for compiled libraries to be built.

Affiliated institutes should be aware that there is little spare effort available at CERN to provide software support, although a member of the Graphics Section has been designated to coordinate the interface between CERN and external graphics users, and may be contacted via GRAPHICS@CERNVM.CERN.CH. The graphics contact people at affiliated institutes will also receive a Newsletter which is distributed via E-Mail. In addition, some first-line support is available from ‘volunteers’ in various CERN member states. The up-to-date list of names of these people will be sent to those institutes holding licences.
Bibliography


Index

APA 6670, 34, 128
APOLLO, 109
Apollo
   GKSGRAL installation, 135
   GKSGRAL on, 70
   GPR, 70
   GSR, 70
   inlib, 72
Appendix E metafile, 3, 30, 50
   CGM, 3
   Computer Graphics Metafile, 3
arc GDP, 10
ASFs, 11
aspect ratio, 18, 19
aspect source flags, 11
attribute source flags, 43
attributes, 10
   3D, 43
   list of, 12
   view index, 43, 46
Bezier curve GDP, 10
BookMaster, 35
bundle index, 11
bundled attributes, 11
cell array, 10
CGM, 50
character
   expansion factor, 15
   height, 16
   spacing, 15
   up-vector, 15
character precision text, 15
choice input, 26
circle GDP, 10
clipping, 19
colour
   index, 16
   look-up table, 17
conid, 30, 62
connection identifier, 6, 30, 62
conversion, 4
converting metafiles, 32
converting numeric variables, 14
coordinate systems, 17
Cray
   GKSGRAL on, 72
cubic curve GDP, 10
current point, 2
Debugging on IBM, 66
deferral states, 7, 19
device coordinates, 17
device drivers, 62, 101
DIAL, 66
distribution, 136
documentation, 5
driver list, 136
echo area, 28
editor, 4, 31
ellipse GDP, 10
Emulators, 5
Encapsulated PostScript, 39
enumerated types, 97
error codes, 86
error file, 6
event input, 26
examples
   3D, segments and viewing, 82
   input initialization, 80
   polylines and fill areas, 76
   request input, 79
   text fonts, 78
   viewports, polymarkers, and text, 76
execution
   on APOLLO, 72
   on IBM, 65
   on UNIX, 70
   on VAX, 67
exref, 65
FALCO, 114
fill area, 9, 13, 14
fill area limitation, 63
fill area set, 42, 43
fill area styles, 63
FIND, 5, 32, 33, 35
fonts, 63, 109
FORTRAN binding, 1, 62
<table>
<thead>
<tr>
<th>FORTRAN Logical Unit Numbers, 63</th>
<th>GIITM, 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCNAME, 38</td>
<td>GINLC, 28</td>
</tr>
<tr>
<td>GDP, 10</td>
<td>GINPK, 28</td>
</tr>
<tr>
<td>generalised drawing primitive, 10</td>
<td>GINSG3, 49</td>
</tr>
<tr>
<td>GGRCONV, 37</td>
<td>GINSG, 22</td>
</tr>
<tr>
<td>GKS routine</td>
<td>GINSK, 28</td>
</tr>
<tr>
<td>GACTM3, 49</td>
<td>GINST, 28</td>
</tr>
<tr>
<td>GACTM, 23</td>
<td>GINVLC, 28</td>
</tr>
<tr>
<td>GACWK, 7, 30</td>
<td>GMSG, 8</td>
</tr>
<tr>
<td>GASF3, 43</td>
<td>GOPKS, 6</td>
</tr>
<tr>
<td>GASGWK, 22</td>
<td>GOPWK, 6, 30, 62</td>
</tr>
<tr>
<td>GCATOG, 54</td>
<td>GPL3, 41</td>
</tr>
<tr>
<td>GCA, 10</td>
<td>GPL, 9</td>
</tr>
<tr>
<td>GCCDWN, 55, 68</td>
<td>GPM3, 41</td>
</tr>
<tr>
<td>GCGIMP, 56</td>
<td>GPM, 9</td>
</tr>
<tr>
<td>GCGTOA, 54</td>
<td>GPRMPT, 57</td>
</tr>
<tr>
<td>GCGWTC, 56</td>
<td>GQCF, 17</td>
</tr>
<tr>
<td>GCIMPL, 56</td>
<td>GQDSP, 19</td>
</tr>
<tr>
<td>GCINIT, 55</td>
<td>GQDVL, 48</td>
</tr>
<tr>
<td>GCLKS, 6</td>
<td>GQLWK3, 51, 63</td>
</tr>
<tr>
<td>GCLRWK, 7</td>
<td>GQLWK, 51, 63</td>
</tr>
<tr>
<td>GCLSG, 21</td>
<td>GQMDS (see GQDSP), 19</td>
</tr>
<tr>
<td>GCLWK, 6, 30</td>
<td>GQMNTN, 51, 63</td>
</tr>
<tr>
<td>GCNAME, 38, 56</td>
<td>GQSGP, 24, 51, 63</td>
</tr>
<tr>
<td>GCQWKN, 56</td>
<td>GQSGS, 51, 63</td>
</tr>
<tr>
<td>GCRSG, 20</td>
<td>GQWKM, 51, 63</td>
</tr>
<tr>
<td>GCSDWN, 57, 67</td>
<td>GRDITM, 31</td>
</tr>
<tr>
<td>GCSDKW, 22</td>
<td>GRENSG, 21</td>
</tr>
<tr>
<td>GCSTOP, 57</td>
<td>GRQCH, 26</td>
</tr>
<tr>
<td>GCWTPC, 57</td>
<td>GRQLC, 26</td>
</tr>
<tr>
<td>GCWTP1, 57</td>
<td>GRQPK, 26</td>
</tr>
<tr>
<td>GDWK, 7, 30</td>
<td>GRQSK, 26, 58</td>
</tr>
<tr>
<td>GDSGWK, 21</td>
<td>GRQST, 26</td>
</tr>
<tr>
<td>GDSG, 21</td>
<td>GRQVL, 26</td>
</tr>
<tr>
<td>GEVPJM, 47</td>
<td>GRSGWK, 25</td>
</tr>
<tr>
<td>GEVTM3, 49</td>
<td>GSASF, 11</td>
</tr>
<tr>
<td>GEVTM, 23</td>
<td>GSCHH, 13, 16</td>
</tr>
<tr>
<td>GEVWMM, 47</td>
<td>GSCHSP, 12, 15</td>
</tr>
<tr>
<td>GFA3, 41</td>
<td>GSCHUP, 12, 15</td>
</tr>
<tr>
<td>GFAS3, 42</td>
<td>GSCHXP, 12, 15</td>
</tr>
<tr>
<td>GFAS, 42</td>
<td>GSCLIP, 19</td>
</tr>
<tr>
<td>GFA, 9</td>
<td>GSCR, 17</td>
</tr>
<tr>
<td>GGDP3, 41</td>
<td>GSDS, 7</td>
</tr>
<tr>
<td>GGDP, 10</td>
<td>GSDTEC, 24</td>
</tr>
<tr>
<td>GGTITM, 31</td>
<td>GSEDCI, 43</td>
</tr>
</tbody>
</table>
INDEX

GSEDFG, 43
GSEDI, 43
GSEDR, 43
GSED, 43
GSELT, 18
GSEWSC, 43
GSFACI, 13, 14
GSFAIS, 13, 14
GSFAI, 13
GSFAR, 12
GSFASI, 13, 14
GSHLIT, 24
GSLN, 12, 13
GSLWSC, 12, 13
GSMKSC, 12, 13
GSMK, 12, 13
GSPARF, 13, 14
GSPAR, 12, 14
GSPA, 13, 14
GSPKID, 25
GSPLCI, 12, 16
GSLI, 13
GSPRI, 12
GSPMCI, 12
GSPMI, 13
GSPMR, 12
GSSGP, 24
GSSGT3, 49
GSSGT, 23
GSTXAL, 12, 16
GSTXCI, 12
GSTXFP, 12, 15
GSTXI, 13
GSTX, 13, 16
GSTXR, 12
GSV3, 44
GSVIS, 24
GSPV, 18
GSVIP, 26
GSVI, 46
GSWVR, 46
GSW3, 44
GSKV3, 47
GSKVP, 18, 39
GSKW3, 47
GSKWN, 18, 39
GSWN, 18
GTX3, 42
GTX, 9
GUARC, 58
GUBEZ1, 58
GUCIR1, 58
GUCIR2, 59
GUCUR1, 59
GUELL1, 59
GUELL2, 59
GUMEN2, 59
GUNERR, 59
GUSHG, 14
GUSIGD, 59
GUWK, 8
GKS-3D, 41
GKS3812, 4, 34
GKS3D
attributes, 43
drawing primitives, 41
example, 82
input, 49
metafile, 50
segments, 49
view index, 43, 46
viewing, 43
GKSCP, 4, 34
GKSED, 4
GKS3D
GKS3D-
GKS3GRL, 4, 62
distribution, 136
installation, 129
on APOLLO, 70
on CRAY, 72
on IBM, 64
on UNIX, 68
on VAX/VMS, 66
workstation types, 101
GKS3GRL-3D, 4, 62
GKSPACK, 53, 64, 66
GCATOG, 54
GCCDWN, 55
GCGIMP, 56
GCGTOA, 54
GCGWTC, 56
GCIMPL, 56
GCINIT, 55
INDEX

GCNAME, 56
GCQWKN, 56
GCSDWN, 57
GCSTOP, 57
GCWTPC, 57
GCWTP1, 57
GPRMPT, 57
GRQSK, 58
GUARC, 58
GUBEZ1, 58
GUCIR1, 58
GUCIR2, 59
GUCUR1, 59
GUELL1, 59
GUELL2, 59
GUMEN2, 59
GUNERR, 59
GUSIGD, 59
Error Messages, 60
Example Program, 60
GKSSGML, 4
GKSTV, 4
GKSSTV, 4, 34
GKSX87, 4, 34
GRAFLIB, 64, 66
GRCONV, 4, 32, 35
Greek text font, 63
GRPLOT, 4, 34
GRVIEW, 4, 31

hardcopy, 4, 34
hatch styles, 63
hatching, 14
HELP, 5, 32, 33, 35
HIGZ, 64, 66
hints, 51
HP PLOTTERS, 111
HPLOT, 64, 66

IBM
3812, 34, 128
5080, 62
debugging, 66
DIAL (use of), 66
GKSGLRAL distribution, 130
GKSGLRAL on, 64
metafile hardcopy, 34

IBM3812, 4
implementation-dependent features, 63
implicit regeneration, 8, 19
include
enum, 65, 67, 69, 71, 73
gtsdev, 65, 67, 69, 71, 73
on Apollo, 71
on CRAY, 73
on IBM, 65
on UNIX, 69
on VAX/VMS, 67
include files, 62, 97
enum, 4
gtsdev, 4
individual attributes, 11
initializing logical input devices, 27
input, 25
classes, 25
device numbers, 25
event, 26
event mode, 62
example, 79
for 3D, 49
inquiry functions, 28
modes, 26
prompt/echo type, 26
request, 26
sample, 26
sample mode, 62
trigger, 26
installation, 129
interpreting metafiles, 31
italic text, 63

I\TeX, 38
levels of GKS, 62
locator input, 26
logical input devices, 25
logical workstations, 3
Macintosh via Versaterm, 114
maximum parameter values, 63
MEGATEK, 62, 111
messages, 8
metafile
2D item numbers, 92
3D item numbers, 94
   for 3D, 50
   interpreter, 31
metafiles, 4, 30
MGKS, 4

networked graphics, 4
non-retained data, 8, 20, 25
normal
   transformation, 17, 28, 44
normalized device coordinates, 17
numeric variables, 14

output
   example, 80

pattern, 13
PCs, 5
PERICOM MG series and Graph Pac, 114
PERICOM MX2000/7000/8000 series, 119
pick identifier, 25
pick input, 25, 26
picture size, 39
PLOT10/GKS, 4
polyline, 9, 12, 13
polyline styles, 13
polymarker, 9, 12, 13
polymarker types, 13
PostScript, 31, 34, 35, 39, 116
postscript, 4
primitives
   2D, 8
   3D, 41
projection transformation, 45
prompt/echo type, 26
prompts, 8
proportionally spaced text, 63

QMS, 117

request input, 26

sample input, 26
segment
   3D, 49
   association, 22
   attributes, 22
   copying, 22
detectability, 24
highlighting, 24
insertion, 22
names, 21
priority, 24
re-drawing, 25
storage, 20
transformation, 23
visibility, 24
segments, 20
SGML, 35, 37
size, 39
string input, 26
string precision text, 15
stroke input, 26
stroke precision text, 15
tar file, 68, 130, 131, 134
TeemTalk, 5
TEKTRONIX 4010/12/14, 118
TEKTRONIX 4100/4200 series, 119
text, 9, 12–14
   3D, 42
   alignment, 16
   example, 78
   fonts, 15
   path, 16
   precision, 15
text fonts, 63, 109
TNR, 18
transformation
   3D normalization, 44
   3D segment, 49
   3D view mapping, 45
   3D view orientation, 45
   3D viewing, 43
   3D workstation, 47
   input initialization, 28
   locator input, 26
   normalization, 17
   segment, 23
   workstation, 18
transformations, 17
trigger, 26
UCO, 5
Unix
GKSGRAL installation, 130
GKSGRAL on, 68
on CRAY, 72
tar file, 68, 130, 131, 134
User Consultancy Office, 5

valuator input, 26
VAX
  GKSGRAL distribution, 129
  GKSGRAL on, 66
  metafile open, 30
  setting window size, 67
VAXstation, 67, 121
VERSATEC, 4, 34, 123
Versaterm for Macintosh, 114
view index, 43, 46
view mapping transformation, 45
view orientation transformation, 45
view representation in 3D, 46
viewing pipeline, 43
viewport, 17, 18, 26, 29, 45
VT240/241, 124
VT340, 126

WDSS, 20
window, 17, 18, 29, 45
WISS, 20
workstation, 6
  activation, 7
  connection identifier, 6
  identifier, 6
  opening/closing, 6
  transformation, 18
  transformation 3D, 47
  type, 6
  update, 7, 8
  viewport, 18
  window, 18
workstation description tables, 109
workstation transformation, 39
world coordinates, 17

X-Window, 4, 67
XEROX, 4, 34, 128