Fall 1987

A proposed C language binding for the Graphical Kernel System-3D

Mark Gerard Bolten

Department:

A PROPOSED C LANGUAGE BINDING FOR

THE GRAPHICAL KERNEL SYSTEM-3D

BY

MARK GERARD BOLTEN, 1960 -

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

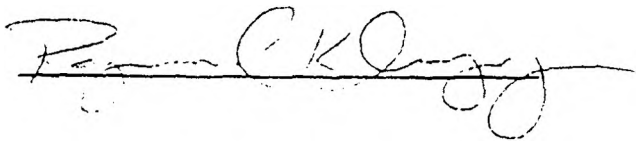
MASTER OF SCIENCE

in

COMPUTER SCIENCE

1987

Advisor



ABSTRACT

This thesis introduces a proposed C language binding definition for the International Standards Organization's draft international standard of the Graphical Kernel System-3D. This work augments the earlier C language binding of the two dimensional version of the Graphical Kernel System commonly known as GKS. The proposed function interface will provide a basis for, if not a final, C language binding for the three dimensional version of the Graphical Kernel System.

ACKNOWLEDGEMENT

The author wishes to thank the members of the faculty of the Department of Computer Science at the University of Missouri-Rolla. Their dedication to science and education has produced graduates who are confident of their abilities to approach and solve the problems which will present themselves in industry. It is to those men who have trained, educated, and counselled me that I am professionally indebted to. In particular, I thank the members of the thesis committee which has guided me in my pursuit of the Master of Science in Computer Science: Dr. C. Y. Ho, the thesis committee chairman, and Dr. Arlan DeKock, both of the Department of Computer Science; and Dr. Raymond Kluczny of the Department of Engineering Management. Finally, the author wishes to thank his wife, Mary-Claire, for her understanding and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

The Graphical Kernel System (GKS) is a two dimensional graphics standard that was approved by the International Standards Organization (ISO) [1] in 1985. Since that time, efforts have been devoted to extending this definition to a three dimensional graphics standard. The GKS-3D [2] standard is nearing completion and no changes are anticipated at this time. The current need is to create language bindings for the GKS-3D standard. This thesis proposes a C language binding that builds upon the work of the C/GKS-2D language binding [3].

A. THE EVOLUTION OF COMPUTER GRAPHICS

The application of computer graphics has grown dramatically since its inception in 1950. In the first demonstration of computer graphics, a CRT was attached to MIT's Whirlwind I [4, 5, 6] computer to facilitate the display of simple line drawings. This primitive demonstration led researchers of the 1950's to the development of more advanced CRT displays, pen plotters and light pens. Unfortunately the primary function of the mainframe computing environment in this era was restricted to providing numerical evaluating processes for military and advanced university research. This activity did not lend itself to the use of graphical peripheral devices.

During the 1960's the mainframe computing environment spread into U.S. industry, which found many applications in business, engineering, and science. In 1962, Ivan Sutherland demonstrated useful interactive computer graphics with his doctoral dissertation, "Sketchpad: A Man-Machine Graphical Communication System", at the Massachusetts Institute of Technology. His implementation spurred a revival of interest in computer graphics. New graphics hardware was developed and the computer industry saw the introduction of digitizing tablets and color displays. These graphics devices were regarded as being useful but prohibitively expensive, and were generally reserved for critical applications in the military, aeronautic, and automobile industries. Corporations with large cash reserves, such as Bell Telephone Laboratories, conducted research to verify the usefulness of interactive computer graphics in applications in their field.

The latter half of the 1970's saw the creation of mainframe environments that supported applications with user interactive response. Computer graphics hardware became so cost effective as to facilitate its use in nearly every computer application. These computer applications spanned a wide range of operating systems and hardware environments, each requiring unique graphics hardware device interfaces. Software vendors began encountering customer requirements for interactive computer graphics in their applications.

Several ad hoc vendor graphics standards arose by virtue of the fact that the manufacturer of those products had entered the field early. PLOT-10 from Tektronics and the CalComp Graphics Interface Package from CalComp were the first products to achieve widespread use. This was fine initially, but these interfaces were limited to the capabilities of the original products and actually hampered the use of more advanced graphics features needed by computer applications.

The graphics system interfaces for applications are extremely time consuming for application developers, and thus very expensive. This is a significant problem for application vendors trying to provide a product with a graphical interface for customers on numerous system configurations. The fastest way to reduce application/system interface development time across multiple system configurations is to have a consistent interface definition that is understood by both application and system developers.

To address the issue of consistency, tremendous computer industry support was given to the ANSI and ISO organizations in the late 1970's and early 1980's to create a standard application programming interface for computer graphics systems. The ISO organization approved the GKS standard for two dimensional graphics in 1985 [1], and since that time efforts have been devoted to creating language bindings for that standard [3, 7, 8, 9] as well as

extending the two dimensional definition to three dimensions [2].

B. COMPUTER GRAPHICS STANDARDS

The American National Standards Institute (ANSI) was founded in 1918 as a privately funded, non-profit organization. Its function is to coordinate and manage the development of American standards by both the private sector and government agencies. The goal of the organization is to eliminate duplication of effort and confusion within the U.S. for both users and developers of various national standards [10].

ANSI has also developed a strong profile in international standards development through its active membership in the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC). ANSI is a founding member of the ISO which was created in 1946. Prior to that, ANSI joined the IEC in the early 1930's to participate in the development of electrotechnical standards.

ANSI membership in international organizations enables U.S. interests to influence the contents of international standards. This was not considered to be important prior to the 1960's while U.S. industries held worldwide export strength and a significant technical competitive edge. During this time, U.S. standards were accepted worldwide on

a de facto basis, especially after World War II. This scenario has changed, and most U.S. industries are required to negotiate and comply with international standards regulations.

The International Standards Organization (ISO) was formed in 1946 by national organizations from 25 countries, and is currently composed of a membership of more than 75 national standards organizations. ISO coordinates the development of standards in every area except electrotechnical standards which are handled by the International Electrotechnical Commission (IEC) [10].

Delegates to ISO committees are concerned primarily with standards that enable world trade for their respective national economic positions. However, ISO is also mandated with the responsibility to promote and provide for regulations with regard to the safety and health of the public, and to protect the world environmental interests.

C. THE GRAPHICAL KERNEL SYSTEM

The Graphical Kernel System (GKS) is the first graphics applications interface definition to be approved as a national and international standard. It has been registered as an American National Standard (ANSI/X3.124-1985) [11] and as an International Standard (ISO 7942-1985) [1]. It is a two dimensional graphics standard that can be

used by most applications requiring computer generated pictures.

The use of the GKS enables applications to be portable across system configurations. The interface definition shields applications from the graphics hardware differences found in each system configuration. This is due to the uniform output primitives and input classes defined in the GKS. The GKS functionality is divided across twelve levels to allow each system configuration to be tailored to its technical capabilities. At least one of the twelve levels, and up to the total of twelve levels, must be implemented for a system to state that it can provide support for GKS applications [12].

The central concept that separates the GKS from all previous informal graphics standards is its definition of workstations. A GKS workstation is a unit composed of zero or more display devices and zero or more input devices, such as a mouse, keyboard, light pen, or data tablet. Each hardware input or output device is abstracted into a logical input or output device, shielding the application code from the particular hardware device being used. The GKS can manage multiple workstations, which enables simultaneous input and output from graphics system configurations. Thus, the workstation concept creates a flexible virtual graphics system which is mapped to a physical system of input and output graphics devices.

Six types of workstations are defined in the GKS; each workstation must be defined as one of the six types described in Table I. The first three workstation types describe physical characteristics of the workstation. The next three workstation types are actually special control workstations used for storage of graphical and non-graphical data.

Use of a GKS workstation by the application program is achieved by first opening the workstation, then activating the same workstation. At this point any and all logical output and input devices become available for the application program. Approximately 280 functions are now at the programmer's disposal for manipulation of the GKS workstation(s).

TABLE I

WORKSTATION TYPES

Category	Description
OUTPUT	only output devices are attached
INPUT	only input devices are attached
OUTIN	output and input devices are attached
WISS	Workstation Independent Segment Storage
MO	Metafile Output
MI	Metafile Input

II. REVIEW OF LITERATURE

A. COMPUTER GRAPHICS

Academic interest in the field of computer graphics has generated a number of texts covering the topic in both general and specific areas. Many consider the first notable text to be Principles of Interactive Computer Graphics [5] by Newman and Sproull, published in 1979. The reader will find an excellent introduction to the concepts of computer graphics. The use of segments is covered which will aid comprehension of the segment functions in the GKS standard. A thorough discussion of transformation calculations for windows, viewports, rotation, translation, and scaling is also provided.

Another introductory text for computer graphics is the Fundamentals of Interactive Computer Graphics [4] by Foley and Van Dam, published in 1982 as part of the IBM Systems Programming Series. This book is useful for its discussion of the Core Graphics System which was undergoing evaluation during the late 1970's by both ANSI and ISO standards committees. It provides most of its examples through the use of a subset of the Core, and mentions the then developing GKS standard as an advancement of concepts in the Core Graphics System.

A variety of texts have been written regarding specific topics in computer graphics, image processing, and pattern recognition. One in particular approaches these

topics by discussing the strictly mathematical tools used in pictorial information processing; Algorithms for Graphics and Image Processing [13] by Theo Pavlidis. Published in 1982, this book is a useful reference for studying two and three dimensional graphical mathematics calculations.

B. ORIGINS OF THE GRAPHICAL KERNEL SYSTEM

The Graphical Kernel System has a significant position in the history of computer science as the first internationally approved standard for two dimensional computer graphics. It replaces numerous regional graphics interfaces in use in the U.S. and other countries. The GKS developed as a result of a desire in the computer industry to agree on a common graphics system functionality to facilitate graphical application communication and portability between completely different computer systems.

The single event that many agree catalyzed the computer industry to support the specific development of a graphics standard was a meeting conducted by the International Federation for Information Processing (IFIP). The IFIP held a workshop in Seillac, France in 1976 at which about 25 graphics experts from Europe and the U.S. created the first working set of computer graphics principles [6, 12]. These principles were utilized by the ISO when it established a Computer Graphics Working Group

(WG2) in 1977, and again by the ANSI when it established an American Computer Graphics Committee (X3) in 1979.

Prior to these events, some very significant graphics standards development was occurring in Germany. The Deutsches Institut fur Normung (DIN) had created a subcommittee for computer graphics (DIN-NI/UA-5.9) in 1975. DIN presented its work as GKS Version 1 to international graphics experts in July 1977 at one of the first meetings of ISO WG2. This work was considered to be technically superior to that of the Core Graphics System being advanced in 1977 by the Association for Computing Machinery/Special Interest Group on Computer Graphics (ACM/Siggraph); although some features of the Core were incorporated into later versions of GKS.

During the years between 1977 and 1985, meetings were held by ISO WG2 to process proposals and comments from national standards committees. These resulted in revisions of the GKS until Version 7.0 was considered mature and stable for submission as an international standard to ISO. An interesting item to note is that a French translation of the GKS was required for submissal along with the English version. This is due to rules established by the United Nations when the ISO was created in 1946. Every standard is required to be translated into both English and French. Additionally, all minutes of meetings of the ISO WG2 had to be recorded in both English and French. This requirement delayed development of the standard to some extent.

C. THE GRAPHICAL KERNEL SYSTEM

Two books describing the Graphical Kernel System have been written by graphics experts closely involved with the development of the GKS standard. Both books describe the background of the people and events that developed the GKS standard. Introduction to the Graphical Kernel System 'G.K.S.' [6] by Hopgood, et al, provides an excellent programming introduction to the use of the GKS. It is intended to provide an informal reference manual for the graphics programmer of GKS applications.

Computer Graphics Programming - GKS - The Graphics Standard [12] by Enderle, Kansy, and Pfaff is a detailed discussion and reference of the GKS. This book also serves as an introduction to the methods and functions of GKS for computer graphics users, programmers, experts, and managers who want to understand the concepts of the GKS standard. More importantly, it provides valuable information for graphics implementors of the GKS standard regarding system configurations. An explanation of the rational behind many of the design considerations of the standard is presented which enables the implementor to determine how the GKS is to be implemented.

The complete references to the Graphical Kernel System are to be found in the American and International standards. The American National Standard for Information

Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description is catalogued as Reference Number ANSI X3.124-1985. The International Standard for Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description is catalogued as Reference Number ISO 7942-1985.

As noted in the introduction of this thesis, the Graphical Kernel System-2D is composed of approximately 280 functions. It is best to explain these functions by grouping them into the ten functional categories listed in the ISO 7942-1985 Functional Description of GKS [1]. These are summarized in Table II and described in the following text.

Control functions provide control over the allocation and deallocation of various state lists and description tables. These data structures are central to the operations of the GKS because most of the functions act by modifying information in some portion of a state list or a description table. The availability of these state lists and description tables affect whether a function is able to carry out its responsibilities. Error codes are issued by a function if the necessary data structures have not been allocated by a control function.

Output functions are restricted to a mere set of six primitives that provide building blocks for the application to create its own sophisticated drawing functions. The polyline function draws connected line segments. The

TABLE II

GKS FUNCTIONAL CATEGORIES

Control Functions
Output Functions
Output Attribute Functions
Transformation Functions
Segment Functions
Input Functions
Metafile Functions
Inquiry Functions
Utility Functions
Error Functions

polymarker draws symbols at specified x,y coordinates, such as points, plus signs, asterisks, x crosses, or small circles. The fill area function draws a closed polygon which may be hollow or filled with a pattern. The text function displays a character string, beginning at a specified x,y coordinate. The cell array paints a rectangular area which may be transformed into a parallelogram. The generalized drawing primitive is the means by which the application programmer uses an application specific primitive that has been provided by the GKS implementation. These frequently include primitives such as circles, circular arcs, ellipses, elliptic arcs, and interpolating curves such as splines.

Output attribute functions are more numerous; approximately 31 functions are provided to control the appearance of the six output functions. The attributes of each output primitive may be specified with one of two methods, either individually or bundled. The concept of bundled attributes was not found in most informal graphics standards used in the U.S., however, European experience with this concept provided the impetus for inclusion of this feature in the GKS standard. A bundled attribute provides a method of defining several individual attributes at once. One function call to a bundled attribute may replace six function calls to individual attributes. Each GKS implementation is required to provide a default set of bundled attributes, and the application programmer is given

the capability to create his own bundled attribute definitions.

Transformation functions in GKS provide a mechanism for translating between three coordinate space definitions. The GKS uses a cartesian coordinate system throughout the standard, however, the units along each axis vary according to the current space definition. The application space definition uses World Coordinates (WC) to describe the units used in the application program, these coordinates are completely device independent. The virtual space definition uses Normalized Device Coordinates (NDC) to provide an intermediate coordinate space for virtual devices, these coordinates are also completely device independent and range between 0.0 and 1.0. The device dependent space definition uses Device Coordinates (DC) to describe the actual hardware device coordinates.

Segment functions allow the application programmer to modularize a computer generated picture into smaller logical pictures. The segments can be created, deleted, and renamed. Segment attribute functions are provided to control the appearance of each segment. These attribute functions affect segment visibility, highlighting, priority, detectability, and transformations on the segment.

Input functions interact with the six logical input device classes created by the GKS. Table III presents each logical input device class along with the logical input

TABLE III

GKS LOGICAL INPUT CLASSES

Device	Input Values Returned to Application
LOCATOR	an x,y value in World Coordinates
STROKE	a set of x,y values in World Coordinates
VALUATOR	a real number
CHOICE	a non-negative integer
PICK	pick status, pick id, segment name
STRING	a character string

values they return to the application. GKS has three operating modes for each logical input device class: REQUEST, SAMPLE, and EVENT. During the request operating mode, the application suspends all activity until a logical input device returns data. Sample mode allows the application to continue processing while the GKS returns the status of the logical input device. Event mode is possible when an input event queue has been allocated. The application issues an event function which samples the logical input from the logical device and places the result in an event queue, for later processing by the application.

Metafile functions provide a facility for temporary and long-term storage of graphical information [14]. This is critical because segment information lasts only as long as the GKS application session. Both graphical and non-graphical information can be placed in metafiles. Metafiles may be transferred between GKS applications on one system, or between systems. This is accomplished through data transfer facilities such as networks, magnetic tapes, or diskettes.

Inquiry functions are extremely useful for finding the contents of the state lists and description tables. More specifically, one may inquire about the GKS general state, the GKS description table, the GKS state list, the workstation description table, the workstation state list, the segment state list, the pixel arrays, and the input queue. Inquiry functions do not generate error conditions

if a problem is encountered while searching for the requested information. These functions will successfully return information about the problem to the application.

Utility functions are provided for the evaluation and accumulation of transform matrixes. The application program can provide a fixed point, shift vector, rotation angle, and scale factor to the Evaluate Transformation Matrix function, and a segment transformation matrix for use by the segment functions will be returned. The Accumulate Transformation Matrix function will combine an existing segment transformation matrix with a new fixed point, shift vector, rotation angle, and scale factor to create a new segment transformation matrix. These are provided as a convenience to the application programmer.

Error functions are used to handle errors that may arise during processing. One error function is called by the GKS to execute an emergency close if a fatal error occurs. An error logging facility prints GKS error messages and the accompanying GKS function identifier to an application defined error file, then returns to the application. An application defined error handler may be installed to allow specific reactions to some error situations.

D. GKS LANGUAGE BINDINGS

Bindings to the computer programming languages most widespread in use were naturally encouraged first due to a desire to spread usage of the GKS standard. ANSI provided a Fortran binding to GKS-2D at the same time that the ANSI Functional Description of GKS-2D was released. The reference number for the Fortran binding is ANSI X3.124.1-1985 [15, 16], and this document is usually included in the appendix of the GKS-2D Functional Description (ANSI X3.124-1985) [11].

C, Pascal, and ADA bindings to the GKS-2D Functional Description are being completed at this time and should become ANSI [17, 18, 19] and ISO [3, 8, 9] standards in the very near future. It is these documents, the ISO definition of the three dimensional extensions to ISO 7942-1985, and the need for three dimensional bindings for ISO/DIS 8805 that serve as the impetus for this paper. The efforts of this work is displayed in the following major section entitled III. RESULTS.

III. RESULTS

The primary contribution of this thesis is the formulation and translation of each GKS-3D functional description into a specific C subroutine definition [20, 21, 22]. The proposed C subroutine definitions are exhibited in this section. A discussion of the methodology used to obtain these results is presented in Section IV of this thesis.

A. C/GKS-3D NOTATIONAL CONVENTIONS

The following format will be used to display the proposed C/GKS-3D binding definitions:

```
GKS Function Name                                                    nx
     gfunction(arg0, arg1, arg2)
     Gtype0    arg0;    /* IN/OUT argument 0 explanation */
     Gtype1    arg1;    /* IN/OUT argument 1 explanation */
     Gtype2    arg2;    /* IN/OUT argument 2 explanation */
```

'GKS Function Name' is the name of the function as listed in the GKS-3D definition standard. 'nx' states the minimum level at which the function is available.

The C function name bound to the GKS functions is 'gfunction'. 'arg0', 'arg1' and 'arg2' are symbolic C arguments corresponding to the GKS parameters. 'Gtype0', 'Gtype1' and 'Gtype2' are the predefined C data types of the symbolic arguments. The predefined C data types are fully described in Appendix B and C of this thesis.

To the right of each argument declaration is a C comment field which contains a brief explanation of each argument. This field indicates the correspondence between the C arguments and the input and output parameters listed in the GKS-3D standard. The parameters passed to the function in the argument are typically input parameters. These input parameters have a comment field which begins with 'IN'. In cases where GKS output parameters are passed through the argument list, the comment field begins with the word 'OUT'.

This explanation also provides information indicating whether the arguments are passed by reference or by value. The explanation of those arguments passed by reference will contain one of the words: pointer, matrix, array, structure or string. 'Pointer' indicates that the address of a single value or structure of the given type is being passed. 'Matrix' (in the context transformation matrix) means that the address of a two by three array of floats is being passed. An 'array' parameter is usually accompanied by a parameter indicating the length of the array. 'Structure' implies that the address of a structure is being passed. 'String' indicates that the address of the first of a list of characters (Gchar) is being passed. The end of a string is indicated by a NUL character.

All C/GKS functions return as their function return value the status of the error log. The return type is

omitted in the specification of all functions indicating that they return the type Gint.

B. C/GKS-3D SUBROUTINE DEFINITIONS

The Graphical Kernel System-3D is an extension of the Graphical Kernel System-2D to three dimensions. However, a basic design principle was made early in the development of the GKS-3D standard to allow all GKS-2D application programs to run unchanged on GKS-3D implementations. This means that the GKS-3D contains all the functions found in the GKS-2D standard, with additional functionality for three dimensional activities. Some functional areas of the GKS did not require additional functions, but required enhancement of the existing functions. Other functional areas did not enhance existing functions, but added new functions. Subsections 1 through 10 are used to display the proposed C/GKS-3D subroutines according to the functional categories used by the ISO 7942-1985 GKS Functional Description [1] and the ISO/DIS 8805 GKS-3D Functional Description [2].

1. GKS-3D Control Functions

No additional control functions were created by ISO DIS 8805, the existing control functions were enhanced to allocate the additional data structures necessary for three dimensional processing.

2. GKS-3D Output Functions Eight output primitives were added to allow three dimensional drawing functionality. These are similar to the original two dimensional output functions, but with z-axis addressability.

```
Polyline 3                                                    L0a
     gpolyline3(numpt, pt)
     Gint       numpt;     /* IN number of points */
     Gpt3       *pt;       /* IN points */

Polymarker 3                                                  L0a
     gpolymarker3(numpt, pt)
     Gint       numpt;     /* IN number of points */
     Gpt3       *pt;       /* IN points */

Text 3                                                        L0a
     gtext3(txpos, txdir, str)
     Gpt3       txpos;          /* IN text position */
     Gfloat     txdir[2][3];    /* IN text direction */
     Gchar      *str;           /* IN character string */

Fill Area 3                                                   L0a
     gfillarea3(numpt, pt)
     Gint       numpt;     /* IN number of points */
     Gpt3       *pt;       /* IN points */

Fill Area Set                                                 L0a
     gfillareaset(numptls, ptind, pt)
     Gint       numptls;   /* IN number of point lists */
     Gint       *ptind;    /* IN indexes to point lists */
     Gpoint     *pt;       /* IN points */

Fill Area Set 3                                               L0a
     gfillareaset3(numptls, ptind, pt)
     Gint       numptls;   /* IN number of point lists */
     Gint       *ptind;    /* IN indexes to point lists */
     Gpt3       *pt;       /* IN points */
```

```
Cell Array 3                                                    L0a
     gcellarray3(par, dim, coind)
     Gpar3      par;      /* IN cell parallelogram */
     Gidim      dim;      /* IN cell array dimensions */
     Gint       *coind;   /* IN colour index array */

Generalized Drawing Primitive 3                                 L0a
     ggdp3(numpt, pt, gdpid, gdprec)
     Gint       numpt;    /* IN number of points */
     Gpt3       *pt;      /* IN points */
     Gint       gdpid;    /* IN GDP3 identifier */
     Ggdprec    *gdprec;  /* IN GDP3 data record */
```

3. GKS-3D Output Attribute Functions Many of the output attributes functions were independent of the two or three dimensional functionality of the output primitives. These functions were enhanced to apply to the three dimensional output primitives. An additional eight output attribute functions were required to process new three dimensional functionality introduced by the edge functions.

```
Set Pattern Reference Point and Vector                          L0a
     gsetpatrefptvec(pt, vec)
     Gpt3       pt;       /* IN pattern reference point */
     Gfloat     vec[2][3];/* IN pattern reference vector */

Set Edge Index                                                  L0a
     gsetedgeind(edind)
     Gint       edind;    /* IN edge index */

Set Edge Flag                                                   L0a
     gsetedgeflag(edfl)
     Giflag     edfl;     /* IN edge flag */

Set Edge Type                                                   L0a
     gsetedgetype(edtp)
     Gint       edtp;     /* IN edge type */
```

```
Set Edge Width Scale Factor                                    L0a
     gsetedgewidth(edwd)
     Gfloat    edwd;     /* IN edge width scale factor */

Set Edge Colour Index                                          L0a
     gsetedgecolourind(edcoind)
     Gint      edcoind;  /* IN edge colour index */

Set Aspect Source Flags 3                                      L0a
     gsetasfs3(lsasfs)
     Gasfs3    *lsasfs;/* IN list of aspect source flags */

Set Edge Representation                                        L1a
     gsetedgerep(ws, edind, edrep)
     Gint      ws;       /* IN workstation identifier */
     Gint      edind;    /* IN edge index */
     Gedbundl  *edrep;   /* IN edge representation */
```

4. GKS-3D Transformation Functions Nine new transformation functions were added to assist in three dimensional transformations.

```
Set Window 3                                                   L0a
     gsetwindow3(trnum, wn)
     Gint      trnum;    /* IN transformation number */
     Glimit3   wn;       /* IN window limits */

Set Viewport 3                                                 L0a
     gsetviewport3(trnum, vp)
     Gint      trnum;    /* IN transformation number */
     Glimit3   vp;       /* IN viewport limits */

Set View Index                                                 L0a
     gsetviewind(vwind)
     Gint      vwind;    /* IN view index */

Set View Representation 3                                      L0a
     gsetviewrep(ws, vwind, vwrep)
     Gint      ws;       /* IN workstation identifier */
     Gint      vwind;    /* IN view index */
     Gvwbundl  vwrep;    /* IN view representation */
```

```
Set View Transformation Input Priority                          L0a
     gsetviewtraninputpri(ws, vwind, refvwind, relpri)
     Gint      ws;       /* IN workstation identifier */
     Gint      vwind;    /* IN view index */
     Gint      refvwind; /* IN reference view index */
     Grelpri   relpri;   /* IN relative priority */


Set HLHSR Identifier                                            L1a
     gsethlhsrid(hlid)
     Gint      hlid;     /* IN HLHSR identifier */


Set HLHSR Mode                                                  L1a
     gsethlhsrid(ws, hlmode)
     Gint      ws;       /* IN workstation identifier */
     Gint      hlmode;   /* IN HLHSR mode */


Set Workstation Window 3                                        L0a
     gsetwswindow3(ws, wswn)
     Gint      ws;       /* IN workstation identifier */
     Glimit3   wswn;     /* IN workstation window limits */


Set Workstation Viewport 3                                      L0a
     gsetwsviewport3(ws, wsvp)
     Gint      ws;     /* IN workstation identifier */
     Glimit3   wsvp;   /* IN workstation viewport limits */
```

5. GKS-3D Segment Functions Two new segment functions were needed for three dimensional segment transformations, and inserting three dimensional segments.

```
Insert Segment 3                                                L2a
     ginsertseg3(seg, tr)
     Gint      seg;      /* IN segment name */
     Gfloat    tr[3][4]; /* IN transformation matrix */


Set Segment Transformation 3                                    L1a
     gsetsegtran3(seg, tr)
     Gint      seg;      /* IN segment name */
     Gfloat    tr[3][4]; /* IN transformation matrix */
```

6. GKS-3D Input Functions No new logical input devices were added. Yet, each of certain input classes were enhanced to include three dimensional input.

Initialise Locator 3 L0b

```
ginitloc3(ws, locdev, loc)
Gint      ws;       /* IN workstation identifier */
Gint      locdev;   /* IN locator device number */
Giloc3    loc;      /* IN initial locator data */
```

Initialise Stroke 3 L0b

```
ginitstroke3(ws, stkdev, stk)
Gint      ws;       /* IN workstation identifier */
Gint      stkdev;   /* IN stroke device number */
Gistk3    stk;      /* IN initial stroke data */
```

Initialise Valuator 3 L0b

```
ginitval3(ws, valdev, val)
Gint      ws;       /* IN workstation identifier */
Gint      valdev;   /* IN valuator device number */
Gival3    val;      /* IN initial valuator data  */
```

Initialise Choice 3 L0b

```
ginitchoice3(ws, chodev, cho)
Gint      ws;       /* IN workstation identifier */
Gint      chodev;   /* IN choice device number */
Gicho3    cho;      /* IN initial choice data */
```

Initialise Pick 3 L1b

```
ginitpick3(ws, pkdev, pk)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN pick device number */
Gipk3     pk;       /* IN initial pick data */
```

Initialise String 3 L0b

```
ginitstring3(ws, strdev, str)
Gint      ws;       /* IN workstation identifier */
Gint      strdev;   /* IN string device number */
Gistr3    str;      /* IN initial string data */
```

```
Request Locator 3                                              L0b
     greqloc3(ws, locdev, st, loc)
     Gint      ws;       /* IN workstation identifier */
     Gint      locdev;   /* IN locator device number */
     Gistat    *st;      /* OUT status */
     Gloc3     *loc;     /* OUT locator data */


Request Stroke 3                                               L0b
     greqstroke3(ws, stkdev, st, stk)
     Gint      ws;       /* IN workstation identifier */
     Gint      stkdev;   /* IN stroke device number */
     Gistat    *st;      /* OUT status */
     Gstk3     *stk;     /* OUT stroke data */


Sample Locator 3                                               L0c
     gsampleloc3(ws, locdev, loc)
     Gint      ws;       /* IN workstation identifier */
     Gint      locdev:   /* IN locator device number */
     Gloc3     *loc;     /* OUT locator data */


Sample Stroke 3                                                L0c
     gsamplestroke3(ws, dev, stk)
     Gint      ws;       /* IN workstation identifier */
     Gint      dev:      /* IN stroke device number */
     Gstk3     *stk;     /* OUT stroke data */


Get Locator 3                                                  L0c
     ggetloc3(loc)
     Gloc3     *loc;     /* OUT locator data */


Get Stroke 3                                                   L0c
     ggetstroke3(stk)
     Gstk3     *stk;     /* OUT stroke data */
```

7. GKS-3D Metafile Functions No additional metafile functions were created by ISO DIS 8805, the existing metafile functions were enhanced to manage the additional data structures necessary for three dimensional processing.

8. GKS-3D Inquiry Functions Numerous additional inquiry functions were added to allow the application to

access the new three dimensional information held in the GKS system tables.

```
Inquire Current Primitive Attribute Values 3                    L0a
     ginqcurprimattr3(patrefpt, patrefvec, vwind, error)
     Gpt3       *patrefpt;          /* OUT current pattern
                                       reference point */
     Gfloat     *patrefvec[2][3];   /* OUT current pattern
                                       reference vector */
     Gint       *vwind;             /* OUT current view
                                       index */
     Gint       *error;             /* OUT error indicator*/


Inquire Current HLHSR Identifier Value                          L1a
     ginqhlhsrid(hlid, error)
     Gint       *hlid;    /* OUT current HLHSR identifier */
     Gint       *error;   /* OUT error indicator */


Inquire Current Individual Attribute Values 3                   L0a
     ginqcurindattr3(edind, edbundl, asfs, error)
     Gint       *edind;   /* OUT current edge index */
     Gedbundl   *edbundl; /* OUT current edge bundle */
     Gasfs3     *asfs;    /* OUT current list of aspect
                             source flags 3 */
     Gint       *error;   /* OUT error indicator */


Inquire Normalization Transformation 3                          L0a
     ginqntran3(ntrnum, wn, vp, error)
     Gint       *ntrnum;  /* IN norm. trans. number */
     Glimit3    *wn;      /* OUT window limits */
     Glimit3    *vp;      /* OUT viewport limits */
     Gint       *error;   /* OUT error indicator */


Inquire Clipping 3                                              L0a
     ginqclip3(clipind, clipvol, error)
     Gclip      *clipind; /* OUT clipping indicator */
     Glimit3    *clipvol; /* OUT clipping volume */
     Gint       *error;   /* OUT error indicator */
```

Inquire Text Extent 3 L0a

```
ginqtextextent3(ws, pos, dir, str, concpt, par, error)
Gint      ws;       /* IN workstation identifier */
Gpt3      pos;      /* IN text position */
Gfloat    dir[2][3];/* IN text direction vector */
Gchar     *str;     /* IN character string */
Gpt3      *concpt;  /* OUT concatenation point */
Gpar3     *par;     /* OUT text extent parallelogram*/
Gint      *error;   /* OUT error indicator */
```

Inquire List of Edge Indices L1a

```
ginqedgeindices(ws, numed, lsed, error)
Gint      ws;       /* IN workstation identifier */
Gint      *numed;   /* OUT num of edge tables */
Gint      *lsed;    /* OUT list of edge indices */
Gint      *error;   /* OUT error indicator */
```

Inquire Edge Representation L1a

```
ginqedgerep(ws, edind, tp, edbundl, error)
Gint      ws;       /* IN workstation identifier */
Gint      edind;    /* IN edge index */
Gitp      tp;       /* IN type of returned values */
Gedbundl  *edbundl; /* OUT edge representation */
Gint      *error;   /* OUT error indicator */
```

Inquire List of View Indices L0a

```
ginqviewindices(ws, numvw, lsvw, error)
Gint      ws;       /* IN workstation identifier */
Gint      *numvw;   /* OUT num of view tables */
Gint      *lsvw;    /* OUT list of view indices */
Gint      *error;   /* OUT error indicator */
```

Inquire View Representation L0a

```
ginqviewrep(ws, vwind, vwrep, error)
Gint      ws;       /* IN workstation identifier */
Gint      vwind;    /* IN view index */
Ginqvw3   *vwrep;   /* OUT view representation */
Gint      *error;   /* OUT error indicator */
```

Inquire HLHSR Mode L1a

```
ginqhlhsrmode(ws, hlupst, reqmode, curmode, error)
Gint      ws;       /* IN workstation identifier */
Gwstus    *hlupst;  /* OUT HLHSR update state */
Gint      *reqmode; /* OUT requested HLHSR mode */
Gint      *curmode; /* OUT current HLHSR mode */
Gint      *error;   /* OUT error indicator */
```

Inquire Workstation Transformation 3 L0a

```
ginqwstran3(ws, wstr, error)
Gint      ws;       /* IN workstation identifier */
Gtr3      *wstr;    /* OUT workstation tran. */
Gint      *error;   /* OUT error indicator */
```

Inquire Locator Device State 3 L0b

```
ginqlocst3(ws, dev, tp, opmode, esw, loc, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN locator device number */
Gitp      tp;       /* IN type of returned values */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Giloc3    *loc;     /* OUT locator data */
Gint      *error;   /* OUT error indicator */
```

Inquire Stroke Device State 3 L0b

```
ginqstrokest3(ws, dev, tp, opmode, esw, stk, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN stroke device number */
Gitp      tp;       /* IN type of returned values */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Gistk3    *stk;     /* OUT stroke data */
Gint      *error;   /* OUT error indicator */
```

Inquire Valuator Device State 3 L0b

```
ginqvalst3(ws, dev, opmode, esw, val, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN valuator device number */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Gival3    *val;     /* OUT valuator data */
Gint      *error;   /* OUT error indicator */
```

Inquire Choice Device State 3 L0b

```
ginqchoicest3(ws, dev, opmode, esw, cho, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN choice device number */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Gicho3    *cho;     /* OUT choice data */
Gint      *error;   /* OUT error indicator */
```

Inquire Pick Device State 3 L1b

```
ginqpickst3(ws, dev, tp, opmode, esw, pk, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN pick device number */
Gitp      tp;       /* IN type of returned values */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Gipk3     *pk;      /* OUT pick data */
Gint      *error;   /* OUT error indicator */
```

Inquire String Device State 3 L0b

```
ginqstringst3(ws, dev, opmode, esw, str, error)
Gint      ws;       /* IN workstation identifier */
Gint      dev;      /* IN string device identifier */
Gimode    *opmode;  /* OUT operating mode */
Gesw      *esw;     /* OUT echo switch */
Gistr3    *str;     /* OUT string data */
Gint      *error;   /* OUT error indicator */
```

Inquire Display Space Size 3 L0a

```
ginqdisplayvolsize(wstp, disp, error)
Gint      wstp;     /* IN workstation type */
Ginqdsp3  *disp;    /* OUT display space size */
Gint      *error;   /* OUT error indicator */
```

Inquire Dynamic Modification of Workstation Attributes 3 L0a

```
ginqdynamicmodwsattr3(wstp, wsattr, error)
Gint      wstp;     /* IN workstation type */
Ginqattr3 *wsattr;  /* OUT workstation attributes */
Gint      *error;   /* OUT error indicator */
```

Inquire Edge Facilities L0a

```
ginqedgefacil(wstp, edfac, error)
Gint      wstp;     /* IN workstation type */
Ginqedfac *edfac;   /* OUT edge facilities */
Gint      *error;   /* OUT error indicator */
```

Inquire Predefined Edge Representation L0a

```
ginqprededgerep(wstp, edind, edbundl, error)
Gint      wstp;     /* IN workstation type */
Gint      edind;    /* IN predefined edge index */
Gedbundl  *edbundl; /* OUT edge bundle */
Gint      *error;   /* OUT error indicator */
```

Inquire View Facilities L0a

```
ginqviewfacil(wstp, numprvw, error)
Gint      wstp;     /* IN workstation type */
Gint      *numprvw; /* OUT num predefined view ind */
Gint      *error;   /* OUT error indicator */
```

Inquire Predefined View Representation L0a

```
ginqpredviewrep(wstp, prvwind, vwbundl, error)
Gint      wstp;     /* IN workstation type */
Gint      prvwind;  /* IN predefined view index */
Gvwbundl  *vwbundl; /* OUT view bundle */
Gint      *error;   /* OUT error indicator */
```

Inquire HLHSR Facilities L0a

```
ginqhlhsrfacil(wstp, hlfac, error)
Gint      wstype;   /* IN workstation type */
Ginqhlfac *hlfac;   /* OUT HLHSR facilities */
Gint      *error;   /* OUT error indicator */
```

Inquire Generalized Drawing Primitive 3 L0a

```
ginqgdp3(wstp, gdpid, numattr, lsattr, error)
Gint      wstp;     /* IN workstation type */
Gint      gdpid;    /* IN gdp identifier */
Gint      *numattr; /* OUT num attribute sets used */
Gint      *lsattr;  /* OUT list of attribute sets */
Gint      *error;   /* OUT error indicator */
```

Inquire Maximum Length of Workstation State Tables 3 L0a

```
ginqmaxlengthwssttable3(wstp, maxvw, maxed, error)
Gint      wstp;     /* IN workstation type */
Gint      *maxvw;   /* OUT max num view tables  */
Gint      *maxed;   /* OUT max num edge bundles */
Gint      *error;   /* OUT error indicator */
```

Inquire Default Locator Device Data 3 L0b

```
ginqdefloc3(wstp, dev, loc, error)
Gint      wstp;     /* IN workstation type */
Gint      dev;      /* IN locator device number */
Gdefloc3  *loc;     /* OUT locator data */
Gint      *error;   /* OUT error indicator */
```

Inquire Default Stroke Device Data 3 L0b

```
ginqdefstroke3(wstp, dev, stk, error)
Gint      wstp;     /* IN workstation type */
Gint      dev;      /* IN stroke device number */
Gdefstk3  *stk;     /* OUT stroke data */
Gint      *error;   /* OUT error indicator */
```

```
Inquire Default Valuator Device Data 3                          L0b
     ginqdefval3(wstp, dev, val, error)
     Gint       wstp;     /* IN workstation type */
     Gint       dev;      /* IN valuator device number */
     Gdefval3   *val;     /* OUT valuator data */
     Gint       *error;   /* OUT error indicator */


Inquire Default Choice Device Data 3                            L0b
     ginqdefchoice3(ws, dev, cho, error)
     Gint       wstp;     /* IN workstation type */
     Gint       dev;      /* IN choice device number */
     Gdefcho3   *cho;     /* OUT choice data */
     Gint       *error;   /* OUT error indicator */


Inquire Default Pick Device Data 3                              L1b
     ginqdefpick3(wstp, dev, pk, error)
     Gint       wstp;     /* IN workstation type */
     Gint       dev;      /* IN pick device number */
     Gdefpk3    *pk;      /* OUT pick data */
     Gint       *error;   /* OUT error indicator */


Inquire Default String Device Data 3                            L0b
     ginqdefstring3(wstp, dev, str, error)
     Gint       wstp;     /* IN workstation type */
     Gint       dev;      /* IN string device number */
     Gdefstr3   *str;     /* OUT string data */
     Gint       *error;   /* OUT error indicator */


Inquire Segment Attributes 3                                    L1a
     ginqsegattr3(seg, segattr, error)
     Gint       seg;      /* IN segment name */
     Gsegattr3  *segattr; /* OUT segment attributes */
     Gint       *error;   /* OUT error indicator */


Inquire Pixel Array Dimensions 3                                L0a
     ginqpixelarraydim3(ws, par, vwind, dim, error)
     Gint       ws;       /* IN workstation identifier */
     Gpar3      par;      /* IN parallelogram */
     Gint       vwind;    /* IN view index */
     Gidim      *dim;     /* OUT pixel array dimensions */
     Gint       *error;   /* OUT error indicator */
```

```
Inquire Pixel Array 3                                          L0a
     ginqpixelarray3(ws, pt, vwind, dim, valid,coind,error)
     Gint      ws;       /* IN workstation identifier */
     Gpt3      pt;       /* IN point */
     Gint      vwind;    /* IN view index */
     Gidim     dim;      /* IN pixel array dimensions */
     Gcovalid  *valid;   /* OUT invalid values present */
     Gint      *coind;   /* OUT colour index array */
     Gint      *error;   /* OUT error indicator */


Inquire Pixel 3                                                L0a
     ginqpixel3(ws, pt, vwind, coind, error)
     Gint      ws;       /* IN workstation identifier */
     Gpt3      pt;       /* IN point */
     Gint      vwind;    /* IN view index */
     Gint      *coind;   /* OUT colour index */
     Gint      *error;   /* OUT error indicator */
```

9. GKS-3D Utility Functions The GKS-2D evaluate and accumulate transformation functions were retained in their original form, and 3D evaluate and accumulate transformation functions were added. View orientation and mapping functionality were created to extend 3D functionality.

```
Evaluate View Orientation Matrix 3                             L0a
     gvieworient3(vw, or, error)
     Gevvwor3  *vw;       /* IN view evaluation data */
     Gfloat    *or[4][4];/* OUT view orientation matrix */
     Gint      *error;    /* OUT error indicator */


Evaluate View Mapping Matrix 3                                 L0a
     gviewmap3(vw, map, error)
     Gevvwmap3 *vw;           /* IN view evaluation data */
     Gfloat    *map[4][4];    /* OUT view mapping matrix */
     Gint      *error;        /* OUT error indicator */


Evaluate Transformation Matrix 3                               L1a
     gevaltran3(tr, segtr, error)
     Gevtr3    *tr;           /* IN evaluation data */
     Gfloat    *segtr[3][4];  /* OUT segment
                                 transformation matrix */
     Gint      *error;        /* OUT error indicator */
```

```
Accumulate Transformation Matrix 3                          L1a
     gaccumtran3(osegtr, tr, nsegtr, error)
     Gfloat    *osegtr[3][4]; /* IN old segment
                                 transformation matrix */
     Gevtr3    *tr;           /* IN evaluation data */
     Gfloat    *nsegtr[3][4]; /* OUT new segment
                                 transformation matrix */
     Gint      *error;        /* OUT error indicator */
```

10. GKS-3D Error Functions No additional error functions were created by ISO DIS 8805, the existing error functions were enhanced to handle the additional data structures necessary for three dimensional error processing.

IV. DISCUSSION

The two dimensional specification of the Graphical Kernel System (GKS) is defined in ISO 7942-1985. As explained in that document, the standard is specified in a language independent manner and is embedded in language dependent layers (language bindings) for use with particular programming languages [16, 20, 23, 24]. A full review of this document is provided in the section entitled II. REVIEW OF LITERATURE.

A. SCOPE AND FIELD OF APPLICATION

GKS is extended to GKS-3D (the Graphical Kernel System for Three Dimensions) by the document ISO/DIS 8805. In a similar way, the language bindings [3, 7, 8, 9, 15, 17, 18, 19] for GKS are extended to form language bindings for GKS-3D. The purpose of this thesis is to define a standard binding for the C computer programming language.

The Graphical Kernel System (GKS) ISO/DIS 8805 specifies a language independent nucleus of a system for computer graphics in three dimensions. For integration into a programming language, GKS-3D is embedded in a language dependent layer obeying the particular conventions of that language. This thesis proposes a language dependent layer for the C language as delineated in the section entitled III. RESULTS.

B. PRINCIPLES AND FEATURES OF THIS BINDING

1. Mapping of GKS Function Names The function names of GKS are all mapped to C function names which start with the letter 'g'. This mapping is generally done in a one-to-one correspondence to the ISO/DIS 8805 GKS-3D standard. The remaining letters after the first one are obtained by deriving a unique acronym from the words of the function name; e.g., activate becomes activate, workstation becomes ws. Hence, the C language function name of the GKS function Activate Workstation is gactivatews. For information about abbreviations, see 'Abbreviations Used in Formulating Function Names' in Appendix A of this thesis. Names used internally which may be known outside GKS, e.g., during linking, start with some easily recognized and documented form. Therefore, applications should not choose external names starting with this construct when using GKS, in order to avoid name conflicts. All C language function names of GKS-2D functions and the proposed GKS-3D functions are available in the tables of Appendix E.

2. Parameters In general, the order of GKS function parameters is preserved. For some functions, however, there are additional parameters which have been inserted in the normal parameter sequence (e.g., array length for arrays which are output parameters).

In order that the application program may inquire any element of a list (member of a set), such as the set of

segment names, in this binding the inquiry functions return only a single element of a list (member of a set). In addition, the total number of elements of the list (members of the set) is always returned. The elements (members) are numbered starting from 1; each invocation of the inquiry function requires the desired element (member) number as an input parameter and returns the corresponding element (member). When the list (set) is empty, a zero is returned as the number of elements (members) and the parameter representing the single element in the list is undefined.

Certain inquiry functions in GKS-3D have current and requested values. For INQUIRE VIEW REPRESENTATION, either the current or the requested values may be inquired and the input value of an additional argument selects which is required.

3. Error Handling There are three error routines in every GKS system, named 'EMERGENCY CLOSE GKS', 'ERROR LOGGING' and 'ERROR HANDLER' (full length C function binding names gemergencyclosegks, gerrorlog, and gerrorhand, respectively). The application programmer may replace the default implementation-provided gerrorhand with an application specific function using the same name, gerrorhand. Furthermore, this user-defined error routine may call the implementation-provided error logging routine gerrorlog. This allows the application to manage certain types of error situations.

4. Non-Standard C Compiler Support Many existing C compilers do not implement the complete C language as defined in reference [17]. The current shortcomings of these compilers should not be allowed to limit the capabilities of this C/GKS-3D language binding, especially as most of the C vendors have announced plans to upgrade their compiler products. Therefore, this binding employs the full functionality of the language, as opposed to a restricted subset. The primary limitation of the current C compilers resides in their inability to process long external names. To assist implementers whose compilers conform to the previous de facto standard (usually called K & R, as described in bibliography reference [21]), this binding proposes short function identifiers such that implementations using non-conforming compilers could easily be developed. A list of short function identifiers has been created and are available for review in Appendix D. They are distinct in the first six characters independent of upper or lower case. Short function identifiers are shown in Appendix D for both the two and three dimensional versions of GKS.

5. Functions versus Macros An implementation may substitute C macros for C functions. However, the macros must be designed so that side-effects work properly. This is due to the fact that applications must be compiled to include the GKS C macro source definitions rather than simply linked to the GKS C subroutine definitions. If a

GKS application is developed with a GKS implementation that uses only functions, and then ported to a GKS implementation that uses macros, it must be completely recompiled. This maintenance expense should be minimized as much as possible by the GKS implementor by providing full documentation on any side-effects of the GKS C macros.

6. Character Strings The C language uses NUL terminated arrays of characters to represent character strings. This is significantly more efficient than the method of providing two arguments for every character string: length of character string, and character string. The application programmer must note that NUL will not be usable as a printable character.

7. Function Identifiers The C language standard requires that compilers recognize internal identifiers to be distinct in at least 31 characters. The standard requires that external identifiers (those seen by the linker) be recognized to a minimum of six characters, independent of case. Implementations which must run in environments which honor less than 31 characters in external identifiers must include a set of C definitions in the header file which equate the long names to equivalent short names. These short names can be found in Appendix C where short function identifiers for GKS-2D and GKS-3D subroutines are listed in tables by function area.

8. Registration The GKS standard reserves certain numerical value ranges for registration as graphical items.

The registered graphical items will be bound to the C programming language (and other programming languages). The registered item binding will be consistent with the binding present in this document.

9. Generalized Drawing Primitives Generalized drawing primitives are separate functions. This binding does not specify the registration of the function identifiers. There are two types of function identifiers for gdps: the registered function identifiers and the unregistered function identifiers.

The identifiers for the registered gdps are:

```
#define   ggdpnnnn        nnnn /* where 'nnnn' is the
                                  registered gdp
                                  identifier          */
```

The identifiers for the unregistered gdps are:

```
#define   gugdpnnnn       nnnn /* where 'nnnn' is
                                  implementation
                                  dependent           */
```

10. Escapes Escapes are separate functions. This binding does not specify the registration of the function identifiers. There are two types of function identifiers for escapes; the registered function identifiers and the unregistered function identifiers.

The names for the registered escapes are:

```
#define   gescapennnn     nnnn /* where 'nnnn' is the
                                  registered escape
                                  identifier          */
```

The identifiers for the unregistered escapes are:

```
#define   guescapennnn nnnn   /* where 'nnnn' is
                                  implementation
                                  dependent           */
```

11. Header Files C provides an include mechanism to allow external files to be included during compilation. Section 4 of this binding describes the datatypes that are to be defined in the file 'gks.h'. This file must be included in any application program that intends to use GKS. Additional implementation dependent definitions may be added to this file, but it is highly recommended that implementation dependent definitions be stored in a separate include file. In light of the fact that the C system file definition 'FILE' is used in gks.h, the C standard i/o header file 'stdio.h' must appear before the appearance of 'gks.h' within the application's program.

12. Error Codes Hard coding numbers into a program decreases its maintainability and is considered by the computer industry to be poor programming practice. This binding defines a set of constants for the GKS error numbers. Each error code is eight characters long and begins with an 'E'. This is consistent with UNIX operating

system conventions [25, 26, 27, 28] which is a primary environment for C compilers.

13. Return Values All of the C functions return the status of the error log as a function value. Since the type of each GKS function is uniformly Gint, the return types of the function are not explicitly stated in the binding. Applications which need to examine the returned error log status may take advantage of the implicit typing of all C functions to Gint and simply use the function invocation in the appropriate expression.

The manner in which the status of the error log is reported is implementation dependent. Two mechanisms are suggested, either return ERRLOGMT status to indicate an empty error log or an implementation might return the number of errors logged. By returning a count of errors logged, the application can determine if an error has occurred since the previous GKS function was invoked.

14. Buffer Management The application program must provide the memory buffer required by the inquiry and input functions. The buffer must be sufficiently large to contain the requested returned data. If the binding cannot fit all the data into the user's buffer, an error condition and the size of the buffer required to contain the entire structure are returned. Inquiry and input functions use buffers for three primary data types:

a. Fixed Structures Functions that return structures of known size. These functions assume the application has

a buffer of adequate size for the returned data. Functions which use this mechanism for returning data are inquiries to a single object of fixed size. The size of the object is determined by the user through the use of the standard C sizeof function. Get, sample and request input functions also fall into this category. The size of the buffer required for the returned data is known to the application program either through inquiry or by setting the buffer size when the device is initialized.

b. Variable Structures Functions which return a structure which contain one or more variable size members. For these inquiries, the application program provides the buffer and its size in units as returned by the C sizeof function. If the buffer size is sufficient, the inquiry proceeds normally, otherwise the inquiry fails. At this point the application must take appropriate action to provide a buffer of sufficient size and retry the inquiry.

c. List of Objects Functions which return lists of objects. Since many of the lists or arrays of objects that may be queried for in GKS are virtually unbounded, a mechanism is setup to inquire a finite subset of the objects, starting at a particular object within the list. Two input parameters are required: a count of the objects which can fit into the user's buffer and an object starting index. One output parameter is required: the actual number of objects which GKS has available. If the start

parameter is out of the range of the list then the error EOUTRANG is returned. Otherwise, a normal non-error condition is generated and returned by the routine.

V. CONCLUSION

This thesis has presented a C language binding for GKS-3D. This work builds upon the previously completed draft of the C language binding for GKS-2D. This fulfills in part the current need in the computer industry for language bindings to the GKS-3D international standard. The C language was selected for a binding of the GKS-3D definition due to the widespread use of the C language that has developed in the last ten years.

The influence of the C programming language [21], and the growing use of the UNIX operating system [25, 26, 27, 28] environment by the computer industry is significant. Numerous engineering workstations have been developed during the 1980's that utilize the UNIX operating system. These workstations are usually designed to provide a very high performance/price ratio to support user interactive-intensive applications. C is included with each UNIX operating system and is often used as the language of choice for the application.

A C language binding for GKS-3D is especially useful for vendors of these systems because they encounter customer requirements for interactive computer graphics in customer applications. The time required for application/system development across multiple system configurations is greatly reduced with a consistent interface definition that is understood by both application and system developers. An additional advantage of the

proposed C graphics interface is that it is easily extensible to the incorporation of advanced features provided by special purpose workstations for applications. However, an application that does not need the extensions, but simply utilizes the standard GKS functionality will be portable across multiple system configurations.

To conclude, the work in the very late 1970's and early 1980's to create a standard application programming interface for graphics systems has achieved reality with the development of the language bindings such as C/GKS-3D.

BIBLIOGRAPHY

1. ISO 7942-1985, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description. Switzerland: ISO. (1985).
2. ISO/DIS 8805, Information Processing Systems - Computer Graphics - Graphical Kernel System for Three Dimensions (GKS-3D) Functional Description. Switzerland: ISO. (1987).
3. ISO/WD 8439, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) C. Switzerland: ISO. (1986).
4. Foley, J. D., and Van Dam, A. Fundamentals of Interactive Computer Graphics. Reading, MA: Addison-Wesley. (1982).
5. Newman, William M., and Sproull, Robert F. Principles of Interactive Computer Graphics. New York, NY: McGraw-Hill. (1979).
6. Hopgood, F.R.A., et al. Introduction to the Graphical Kernel System 'G.K.S.'. London, England: Academic Press. (1983).
7. ISO 7943-1985, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) FORTRAN. Switzerland: ISO. (1985).
8. ISO/WD 8420, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) PASCAL. Switzerland: ISO. (1986).

9. ISO/DIS 8651/3, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) ADA. Switzerland: ISO. (1987).

10. National Computer Graphics Association. Standards in the Computer Graphics Industry. Fairfax, VA: NCGA. (1986).

11. ANSI X3.124-1985, Information Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description. New York, NY: ANSI. (1985).

12. Enderle, G., Kansy, K., and Pfaff, G. Computer Graphics Programming - GKS - The Graphics Standard. Berlin, Germany: Springer-Verlag. (1984).

13. Pavlidis, Theo. Algorithms for Graphics and Image Processing. Rockville, MD: Computer Science Press. (1982).

14. ISO/WD 8632, Information Processing Systems - Computer Graphics - Metafile for Transfer and Storage of Picture Description Information. Switzerland: ISO. (1987).

15. ANSI X3.124.1-1985, Information Systems - Computer Graphics - Graphical Kernel System (GKS) FORTRAN Binding. New York, NY: ANSI. (1985).

16. ANSI X3.9-1978, Information Systems - Computer Languages - FORTRAN Language Specification. New York, NY: ANSI. (1978).

17. ANSI X3H34/87-12R5, Information Systems - Computer Graphics - Graphical Kernel System (GKS) C Binding. New York, NY: ANSI. (1987).

18. ANSI X3H3/86-67R1, Information Systems - Computer Graphics - Graphical Kernel System (GKS) PASCAL Binding. New York, NY: ANSI. (1987).

19. ANSI X3H3/86-18, Information Systems - Computer Graphics - Graphical Kernel System (GKS) ADA Binding. New York, NY: ANSI. (1986).

20. ANSI X3J11/85-044, C Programming Language Standard Technical Report. New York, NY: ANSI. (1985).

21. Kernighan, Brian W., and Ritchie, Dennis M. The C Programming Language. London, England: Prentice-Hall. (1978).

22. Digital Equipment Corporation. Programming in Vax-11 C. Nashua, NH: Digital Equipment Press. (1982).

23. ANSI X3.97-1983, Information Systems - Computer Languages - PASCAL Language Specification. New York, NY: ANSI. (1983).

24. ISO 7185, Information Processing Systems - Programming Languages - PASCAL. Switzerland: ISO. (1987).

25. AT&T. System V Interface Definition Volume I. Indianapolis, IN: AT&T Press. (1986).

26. AT&T. System V Interface Definition Volume II. Indianapolis, IN: AT&T Press. (1986).

27. AT&T. UNIX System V Programmer's Guide. London, England: Prentice-Hall. (1987).

28. Bach, Maurice J. The Design of the UNIX Operating System. Englewood Cliffs, NJ: Prentice-Hall. (1986).

VITA

Mark Gerard Bolten was born in Jefferson City, Missouri on July 24, 1960. He received his primary and secondary education in Jefferson City, Missouri from the parochial school system of the Diocese of Jefferson City. In December of 1981, he received his Bachelor of Science degree from the University of Missouri-Rolla. He attended classes in the Graduate School of the University of Missouri-Rolla during the period January 1982 to May 1983.

While at the University of Missouri-Rolla Mr. Bolten was a CAD/CAM Graduate Research Assistant for the Computer Center and a Graduate Research Assistant for the Department of Computer Science. After leaving the University of Missouri-Rolla Mr. Bolten was employed as an systems applications analyst by a major American aerospace firm. In this capacity he was involved in the initial design phase of a new management science software system. He was also involved in a commercial telecommunications design effort with the same firm. Since that time, Mr. Bolten has pursued a number of consulting assignments that have ranged over a variety of computer-related fields.

Mr. Bolten is currently a member of the Association for Computing Machinery, the Institute of Electrical and Electronics Engineers, the Society of Manufacturing Engineers, and the Association of Information and Image Management.

APPENDIX A

ABBREVIATIONS USED IN FUNCTION NAMES

TABLE IV

ABBREVIATIONS USED IN GKS-3D FUNCTION NAMES[1]

Word or Phrase	Abbreviation
cell	FULL
edge	FULL
edgetype	FULL
edgewidth	FULL
flag	FULL
HLHSR	FULL
set	FULL
view	FULL
volume	vol

1 In this table FULL denotes words or phrases taken in their entirety without abbreviation; NULL denotes those which are deleted completely when forming function names.

TABLE V

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES[2]

Word or Phrase	Abbreviation
accumulate	accum
activate	FULL
active	FULL
alignment	align
all	NULL
and	NULL
array	FULL
aspect source flags	asf
associate	assoc
associated	assoc
attribute	attr
attributes	attr
available	avail
await	FULL
base	FULL
category	FULL
character	char
choice	FULL
classification	class
clear	FULL
clipping	clip
close	FULL

2 In this table FULL denotes words or phrases taken in their entirety without abbreviation; NULL denotes those which are deleted completely when forming function names.

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
colour	FULL
connection	conn
copy	FULL
create	FULL
current	cur
data	NULL
deactivate	FULL
default	def
deferral	defer
delete	del
detectability	det
device	NULL
devices	NULL
dimensions	dim
display	FULL
dynamic	NULL
emergency	FULL
error	FULL
escape	FULL
evaluate	eval
expansion	expan
extent	FULL
facilities	facil

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
factor	NULL
fill area	fillarea
flush	FULL
font	FULL
from	NULL
generalized drawing primitive	gdp
generalized drawing primitives	gdp
get	FULL
gks	FULL
gksm	FULL
handling	hand
height	FULL
highlighting	highlight
identifier	id
in use	NULL
index	ind
indicator	NULL
indices	FULL
individual	indiv
initialize	init
input	FULL
insert	FULL
inquire	inq

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
interior	int
item	NULL
length	NULL
linetype	FULL
linewidth	FULL
list of	NULL
locator	loc
logging	log
logical	NULL
marker	FULL
matrix	NULL
maximum	max
message	FULL
mode	FULL
modification	mod
more	FULL
name	FULL
names	name
normalization	n
number	num
numbers	num
of	NULL
on	NULL

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
open	FULL
operating	op
overflow	FULL
path	FULL
pattern	pat
pick	FULL
pixel	FULL
point	pt
polyline	FULL
polymarker	FULL
precision	prec
predefined	pred
priorities	pri
priority	pri
primitive	prim
queue	NULL
read	FULL
redraw	FULL
reference	ref
rename	FULL
representation	rep
request	req
sample	FULL

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
scale	NULL
select	sel
segment	seg
segments	segs
set	FULL
set of	NULL
simultaneous	NULL
size	FULL
spacing	space
state	st
states	st
string	FULL
stroke	FULL
style	FULL
supported	NULL
surface	NULL
table	FULL
text	FULL
to	NULL
transformation	tran
type	FULL
types	type
up	FULL

TABLE V (continued)

ABBREVIATIONS USED IN GKS-2D FUNCTION NAMES

Word or Phrase	Abbreviation
update	FULL
valuator	val
value	NULL
values	NULL
vector	NULL
viewport	FULL
visibility	vis
width	FULL
window	FULL
with	NULL
workstation	ws
workstations	ws
write	FULL

APPENDIX B

ABBREVIATIONS USED IN DATA TYPE NAMES

TABLE VI

ABBREVIATIONS USED IN GKS DATA TYPE NAMES[3]

Word or Phrase	Abbreviation
attribute control flag	acf
alignment	align
array	NULL
aspect source flag	asf
aspect source flags	asfs
attribute	attr
attributes	attrs
availability	avail
back	bk
buffer	buf
bundle	bundl
category	cat
classification	class
clipping indicator	clip
clear	clr
colour	co
choice	cho

3 The type definition names are not abbreviated using the same rules as the function identifiers. This would have generated many type definition names which differed from function identifiers only in the case of the sentinel character.

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
concatenation	conc
connection and type	ct
coordinate	c
current	cur
data	NULL
default	def
deferral	defer
deferral mode	defmode
detectability	det
device	dev
dimension	dim
direction	dir
display	dsp
distance	dist
deferral and update state	dus
echo	e
edge	ed
evaluate	ev
event	event
expansion	exp
extent	NULL
facilities	fac
factor	NULL

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
fill area	fill
fixed	fx
flag	fl
font and precision	fp
front	fr
generalized drawing primitive	gdp
GKS metafile	gksm
highlighting	hi
HLHSR	hl
horizontal	hor
height	ht
identifier	id
immediate regeneration	imm
implicit regeneration	irg
index(es)	ind
indice	ind
individual	indiv
initial	i
input	in
inquire	inq
integer dimension	idim
interior	inter
(coordinate) limits	limit

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
(poly)line	ln
list	ls
locator	loc
map	map
(poly)marker	mk
matrix	NULL
maximum	max
minimum	min
mode	mode
modification	mod
name	NULL
nominal	nom
normal	n
normalization	n
number	num
old	o
open	op
operating	op
orientation	or
pattern	pat
parallelogram	par
pick	pk
plane	pl

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
(coordinate) point	pt
polyline/fill area control flag	pfcf
position	pos
precision	prec
predefined	pr
primitive	prim
priority	pri
projection	proj
prompt and echo type	pet
queue	queue
raster	ras
record	rec
rectangle	rect
reference	ref
regeneration	regen
relative	rel
representation	rep
request	req
response	res
rotation	rot
scale	NULL
segment(s)	seg
shift	sh

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
spacing	sp
state	st
status	st
string	str
stroke	stk
surface	surf
switch	sw
text	tx
transformation	tr
transformation update state	tus
type	tp
unit(s)	un
update	up
valid	valid
valuator	val
value	value
vector	vec
vertical	ver
view	vw
viewport	vp
visibility	vis
volume	vol
width	wd

TABLE VI (continued)

ABBREVIATIONS USED IN GKS DATA TYPE NAMES

Word or Phrase	Abbreviation
window	wn
workstation	ws
workstation identifier	ws

APPENDIX C

DATA TYPES USED IN GKS-3D FUNCTIONS

A. TYPE DEFINITIONS

The data types defined in this section allow for the ease of porting GKS implementation between different environments. These types are used as the basis for all the other data types.

1. Gchar - character This datatype must be defined by the implementation as a type suitable for containing the character set used by the GKS implementation. Suggest:

```
typedef   char          Gchar;
```

2. Gfloat - floating point number This datatype must be defined by the implementation as a type suitable for containing the real values used by the GKS implementation. Suggest:

```
typedef   float         Gfloat;
```

3. Gint - integer This datatype must be defined by the implementation as a type suitable for containing the integer values used by the GKS implementation. Suggest:

```
typedef   int           Gint;
```

4. Gfile - file object This datatype must be defined by the implementation as a type suitable for containing the file object type used by the GKS implementation. Suggest:

```
typedef   FILE          Gfile;
```

5. Glong - long integer This datatype must be defined by the implementation as a type suitable for containing an integer of twoice the size of Gint. Suggest:

typedef long Glong;

6. Guint - unsigned integer This datatype must be defined by the implementation as a type suitable for containing the unsigned integer values used by the GKS implementation. Suggest:

typedef unsigned int Gint;

B. DATA TYPE FORMATS

The following format is used to display the proposed C/GKS-3D data types:

```
GKS Data Type Name
typedef struct {
     Gtype0    data0;     /* data type 0 explanation */
     Gtype1    data1;     /* data type 1 explanation */
     Gtype2    data2;     /* data type 2 explanation */
} 'GKS Data Type Name';
```

'GKS Data Type Name' is the name of the C data type as used in the C/GKS-3D definition. Each data type name is displayed in alphabetical order for reference.

To the right of each data type definition is a C comment field which contains a brief explanation of each data type. This explanation also provides information indicating whether the arguments are passed by reference or

by value. The explanation of those data types passed by reference will contain one of the words: pointer, matrix, array, structure or string. 'Pointer' indicates that the address of a single value or structure of the given type is being passed. 'Matrix' (in the context transformation matrix) means that the address of a two by three array of floats is being passed. An 'array' parameter is usually accompanied by a parameter indicating the length of the array. 'Structure' implies that the address of a structure is being passed. 'String' indicates that the address of the first of a list of characters (Gchar) is being passed. The end of a string is indicated by a NUL character.

C. C/GKS-3D DATA TYPE DEFINITIONS

```
Gasf                        /* aspect source flag */
typedef enum {
     GBUNDLED,              /* bundled attribute */
     GINDIVIDUAL            /* individual attribute */
} Gasf;


Gasfs3                      /* 3D aspect source flags */
typedef struct {
     Gasf       edfl;       /* edge flag */
     Gasf       edtp;       /* edge type */
     Gasf       edwd;       /* edge width scale factor */
     Gasf       edcoind;    /* edge colour index */
} Gasfs3;


Gclip                       /* clipping indicator */
typedef enum {
     GCLIP,                 /* clip */
     GNOCLIP                /* no clip */
} Gclip;
```

```
Gcovalid                    /* valid colour values */
typedef enum {
     GABSENT,               /* colour value absent */
     GPRESENT               /* colour value present */
} Gcovalid;


Gcstat                      /* choice status */
typedef enum {
     GOK,                   /* choice */
     GNOCHOICE              /* no choice */
} Gcstat;


Gcsw                        /* coordinate switch */
typedef enum {
     GWC,                   /* world coordinate */
     GNDC                   /* normalized device coordinate */
} Gcsw;


Gdefcho3                    /* inquire default choice 3 */
typedef struct {
     Gint       maxcho;     /* maximum number of choices */
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Gchorec    *chorec;    /* default choice data record */
} Gdefcho3;


Gdefloc3                    /* inquire default locator 3 */
typedef struct {
     Gpt3       locpos;     /* default locator position */
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Glocrec    *locrec;    /* default locator data record */
} Gdefloc3;


Gdefpk3                     /* inquire default pick 3 */
typedef struct {
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Gpkrec     *pkrec;     /* default pick data record */
} Gdefpk3;
```

```
Gdefstk3                    /* inquire default stroke 3 */
typedef struct {
     Gint       stkbuf;     /* maximum stroke buffer size */
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Gstkrec    *stkrec;    /* default stroke data record */
} Gdefstk3;


Gdefstr3                    /* inquire default string 3 */
typedef struct {
     Gint       strbuf;     /* maximum string buffer size */
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Gstrrec    *strrec;    /* default string data record */
} Gdefstk3;


Gdefval3                    /* inquire default valuator 3 */
typedef struct {
     Gfloat     ivalue;     /* default initial value */
     Gint       numpet;     /* num of prompt and echo types */
     Gint       *lspet;     /* list of prompt and echo types*/
     Glimit3    evol;       /* default echo volume */
     Gvalrec    *valrec;    /* default valuator data record */
} Gdefval3;


Gdevun                      /* device coordinate unit */
typedef enum {
     GMETRES,               /* metric */
     GOTHER                 /* other */
} Gdevun;


Gedbundl                    /* edge bundle */
typedef struct {
     Giflag     edfl;       /* edge flag */
     Gint       edtp;       /* edge type */
     Gfloat     edwd;       /* edge width scale factor */
     Gint       edcoind;    /* edge colour index */
} Gedbundl;


Gesw                        /* echo switch */
typedef enum {
     GECHO,                 /* echo */
     GNOECHO                /* no echo */
} Gesw;
```

```
Gevtr3                      /* evaluate transformation 3 */
typedef struct {
     Gpt3       fxpt;       /* fixed point */
     Gfloat     shvec;      /* shift vector */
     Gfloat     xrot;       /* rotation angle in x */
     Gfloat     yrot;       /* rotation angle in y */
     Gfloat     zrot;       /* rotation angle in z */
     Gfloat     sc[3];      /* scale factors */
     Gcsw       csw;        /* coordinate switch */
} Gevtr3;


Gevvwmap3                   /* evaluate view map 3 */
typedef struct {
     Glimit     wn;         /* window limits */
     Glimit3    vp;         /* viewport limits */
     Gprojtp    projtp;     /* projection type */
     Gpt3       projrefpt;/* projection reference point */
     Gfloat     vwpldist;   /* view plane distance */
     Gfloat     frpldist;   /* front plane distance */
     Gfloat     bkpldist;   /* back plane distance */
} Gevvwmap3;


Gevvwor3                    /* evaluate view orientation 3 */
typedef struct {
     Gpt3       vwrefpt;    /* view reference point */
     Gfloat     vwpln[3];   /* view plane normal */
     Gfloat     vwvec[3];   /* view up vector */
     Gcsw       csw;        /* coordinate switch */
} Gevvwor3;


Gicho3                      /* initialise choice 3 */
typedef struct {
     Gcstat     ist;        /* initial status */
     Gint       ichonum;    /* initial choice number */
     Gint       pet;        /* prompt and echo type */
     Glimit3    evol;       /* echo volume */
     Gchorec    *chorec;    /* choice data record */
} Gicho3;


Gidim                       /* dimensions */
typedef struct {
     Guint      x;          /* x dimension */
     Guint      y;          /* y dimension */
} Gidim;
```

```
Giflag                          /* initial status */
typedef enum {
     GOFF,                      /* off status */
     GON                        /* on status */
} Giflag;


Giloc3                          /* initialise locator 3 */
typedef struct {
     Gint       intrnum;        /* initial normalization
                                   transformation number */
     Gint       ivwind;         /* initial view index */
     Gpt3       ilocpos;        /* initial locator position */
     Gint       pet;            /* prompt and echo type */
     Glimit3    evol;           /* echo volume */
     Glocrec    *locrec;        /* locator data record */
} Giloc3;


Gimode                          /* input mode */
typdef enum {
     GREQUEST,                  /* request mode */
     GSAMPLE,                   /* sample mode */
     GEVENT                     /* event mode */
} Gimode;


Ginqattr3                       /* inquire change attributes 3 */
typedef struct {
     Gmodtp     vwtr;           /* view transformation */
     Gmodtp     edbundl;        /* edge bundle representation */
     Gmodtp     hlmode;         /* HLHSR mode */
} Ginqattr3;


Ginqdsp3                        /* inquire displace space 3 */
typedef struct {
     Gdevun     un;             /* device coordinate units */
     Gfloat     devun[3];       /* display space size in
                                   device coordinate units */
     Gint       rasun[3];       /* display space size in
                                   raster coordinate units */
} Ginqdsp3;
```

```
Ginqedfac                   /* inquire edge facilities */
typedef struct {
     Gint       numedtp;    /* number of edge types */
     Gint       *lsedtp;    /* list of edge types */
     Gint       numedwd;    /* number of edge widths */
     Gfloat     nomedwd;    /* nominal edge width */
     Gfloat     minedwd;    /* minimum edge width */
     Gfloat     maxedwd;    /* maximum edge width */
     Gint       numpred;    /* number of predefined edge
                               indices */
} Ginqedfac;


Ginqgdp3                    /* inquire generalized drawing
                               primitives 3 */
typedef struct {
     Gint       numattr;    /* number of sets of attributes */
     Gint       *lsattr;    /* list of sets of attributes */
} Ginqgdp3;


Ginqhlfac                   /* inquire HLHSR facilities */
typedef struct {
     Gint       numhlid;    /* number of HLHSR identifiers */
     Gint       *lshlid;    /* list of HLHSR identifiers */
     Gint       numhlmode;/* number of HLHSR modes */
     Gint       *lshlmode;/* list of HLHSR modes */
} Ginqhlfac;


Ginqhlmode                  /* inquire HLHSR mode */
typedef struct {
     Gwstus     hlupst;     /* HLHSR update status */
     Gint       reqmode;    /* requested HLHSR mode */
     Gint       curmode;    /* current HLHSR mode */
} Ginqhlmode;


Ginqtx3                     /* inquire text representation 3*/
typedef struct {
     Gpt3       concpt;     /* concatenation point */
     Gpar3      txpar;      /* text extent parallelogram */
} Ginqtx3;
```

```
Ginqvw3                    /* inquire view representation */
typedef struct {
     Gwstus     vwtrupst;       /* view trans. update state*/
     Gfloat     reqor[4][4];    /* requested orientation */
     Gfloat     curor[4][4];    /* current orientation */
     Gfloat     reqmap[4][4];   /* requested mapping */
     Gfloat     curmap[4][4];   /* current mapping */
     Glimit3    reqclip;        /* requested clip limits */
     Glimit3    curclip;        /* current clip limits */
     Gclip      reqxy;          /* requested xy clip status*/
     Gclip      curxy;          /* current xy clip status */
     Gclip      reqbk;          /* requested back clip */
     Gclip      curbk;          /* current back clip */
     Gclip      reqfr;          /* requested front clip */
     Gclip      curfr;          /* current front clip */
} Ginqvw3;


Ginqwstr3                  /* inquire workstation
                              transformations 3 */
typedef struct {
     Gwstus     wstrupst; /* workstation transformation
                             update status */
     Glimit3    reqwswn;  /* requested workstation window */
     Glimit3    curwswn;  /* current workstation window */
     Glimit3    reqwsvp;  /* request workstation viewport */
     Glimit3    curwsvp;  /* current workstation viewport */
} Ginqwstr3;


Gipk3                      /* initialise pick 3 */
typedef struct {
     Gpstat     ist;      /* initial status */
     Gint       iseg;     /* initial segment */
     Gint       ipkid;    /* initial pick identifier */
     Gint       pet;      /* prompt and echo type */
     Glimit3    evol;     /* echo volume */
     Gpkrec     *pkrec;   /* pick data record */
} Gipic3;


Gistat                     /* initial status */
typedef enum {
     GOK,                  /* ok */
     GNONE                 /* none */
} Gistat;
```

```
Gistk3                          /* initialise stroke 3 */
typedef struct {
     Gint       intrnum;        /* initial normalization
                                   transformation number */
     Gint       ivwind;         /* initial view index */
     Gint       inumpt;         /* initial number of points */
     Gpt3       *istkpt;        /* initial points in stroke */
     Gint       pet;            /* prompt and echo type */
     Glimit3    evol;           /* echo volume */
     Gstkrec    *stkrec;        /* stroke data record */
} Gistk3;


Gistr3                          /* initialise string 3 */
typedef struct {
     Gchar      *istr;          /* initial string */
     Gint       pet;            /* prompt and echo type */
     Glimit3    evol;           /* echo volume */
     Gstrrec    *strrec;        /* string data record */
} Gistr3;


Gitp                            /* type of returned values */
typedef enum {
     GSET,                      /* set */
     GREALIZED                  /* realized */
} Gitp;


Gival3                          /* initialise valuator 3 */
typedef struct {
     Gfloat     ivalue;         /* initial value */
     Gint       pet;            /* prompt and echo type */
     Glimit3    evol;           /* echo volume */
     Gvalrec    *valrec;        /* valuator data record */
} Gival3;


Glimit                          /* 2D space limits */
typedef struct {
     Gfloat     xmin;           /* x minimum */
     Gfloat     xmax;           /* x maximum */
     Gfloat     ymin;           /* y minimum */
     Gfloat     ymax;           /* y maximum */
} Glimit;
```

```
Glimit3                     /* 3D space limits */
typedef struct {
     Gfloat     xmin;       /* x minimum */
     Gfloat     xmax;       /* x maximum */
     Gfloat     ymin;       /* y minimum */
     Gfloat     ymax;       /* y maximum */
     Gfloat     zmin;       /* z minimum */
     Gfloat     zmax;       /* z maximum */
} Glimit3;


Gloc3                       /* 3D locator */
typedef struct {
     Gint       ntrnum;     /* normalization transformation
                               number */
     Gint       vwind;      /* view index */
     Gpt3       locpos;     /* locator position */
} Gloc3;


Gmodtp                      /* dynamic modification types */
typedef enum {
     GIRG,                  /* implicit regeneration */
     GIMM                   /* immediate regeneration */
} Gmodtp;


Gpar3                       /* 3D parallelogram */
typedef struct {
     Gpt3       p;          /* first point coordinate */
     Gpt3       q;          /* second point coordinate */
     Gpt3       r;          /* third point coordinate */
} Gpar3;


Gprojtp                     /* projection type */
typdef enum {
     GPARALLEL,             /* parallel projection */
     GPERSPECTIVE           /* perspective projection */
} Gprojtp;


Gpstat                      /* pick status */
typedef enum {
     GOK,                   /* pick */
     GNOPICK                /* no pick */
} Gpstat;
```

```
Gpt3                        /* 3D point */
typedef struct {
     Gfloat    x;           /* x coordinate */
     Gfloat    y;           /* y coordinate */
     Gfloat    z;           /* z coordinate */
} Gpt3;


Gras3                       /* 3D raster data */
typedef struct {
     Gint      xmin;        /* x minimum */
     Gint      xmax;        /* x maximum */
     Gint      ymin;        /* y minimum */
     Gint      ymax;        /* y maximum */
     Gint      zmin;        /* z minimum */
     Gint      zmax;        /* z maximum */
} Gras3;


Grelpri                     /* relative priority */
typedef enum {
     GHIGHER,               /* higher priority */
     GLOWER                 /* lower priority */
} Grelpri;


Gsegattr3                   /* segment attributes 3 */
typedef struct {
     Gfloat    segtr[3][4];   /* segment transformation
                                 matrix */
     Gsegvis   segvis;        /* segment visibility */
     Gseghi    seghi;         /* segment highlighting */
     Gfloat    segpri;        /* segment priority */
     Gsegdet   segdet;        /* segment detectability */
} Gsegattr3;


Gsegdet                     /* segment detectability */
typedef enum {
     GUNDETECTABLE,         /* undetectable segment */
     GDETECTABLE            /* detectable segment */
} Gsegdet;


Gseghi                      /* segment highlighting */
typedef enum {
     GNORMAL,               /* normal segment */
     GHIGHLIGHTED           /* highlighted segment */
} Gseghi;
```

```
Gsegvis                    /* segment visibility */
typedef enum {
     GVISIBLE,             /* visible segment */
     GINVISIBLE            /* invisible segment */
} Gsegvis;


Gstk3                      /* 3D stroke */
typedef struct {
     Gint      ntrnum;     /* normalization transformation
                              number */
     Gint      vwind;      /* view index */
     Gint      numpt;      /* number of points */
     Gpt3      *stkpt;     /* stroke points */
} Gstk3;


Gtr3                       /* 3D transformations */
typedef struct {
     Gwstus    wstrupst;   /* workstation transformation
                              update state */
     Glimit3   reqwswn;    /* requested workstation window */
     Glimit3   curwswn;    /* current workstation window */
     Glimit3   reqwsvp;    /* requested workstationviewport*/
     Glimit3   curwsvp;    /* current workstation viewport */
} Gtr3;


Gvwbundl                   /* view bundle */
typedef struct {
     Gfloat    vwor[4][4];    /* view orientation matrix */
     Gfloat    vwmap[4][4];   /* view mapping matrix */
     Glimit3   vwclip;        /* view clipping limits */
     Gclip     xyclip;        /* x-y clipping indicator */
     Gclip     bkclip;        /* back clipping indicator */
     Gclip     frclip;        /* front clipping indicator*/
} Gvwbundl;


Gwstus                     /* workstation transformation
                              update status */
typedef enum {
     GNOTPENDING,          /* transformation not pending */
     GPENDING              /* transformation pending */
} Gwstus;
```

APPENDIX D

C/GKS SHORT FUNCTION IDENTIFIERS

TABLE VII

SHORT FUNCTION IDENTIFIERS FOR GKS-3D CONTROL FUNCTIONS

Function	Identifier
none	n/a

TABLE VIII

SHORT FUNCTION IDENTIFIERS FOR GKS-3D OUTPUT FUNCTIONS

Function	Identifier
gpolyline3	gpl3
gpolymarker3	gpm3
gpolytext3	gtx3
gfillarea3	gfa3
gfillareaset	gfas
gcellarray3	gca3
ggdp3	ggdp3

TABLE IX

SHORT FUNCTION IDENTIFIERS FOR GKS-3D
OUTPUT ATTRIBUTES FUNCTIONS

Function	Identifier
gsetasf3	gsasf3
gsetedgecolourind	gsedci
gsetedge	gsedfl
gsetedgeind	gsedi
gsetedgerep	gsedr
gsetedgetype	gsedt
gsetedgewidth	gsewsc
gsethlhsrid	gshrid
gsethlhsrmode	gshrm
gsetpatrefpt3	gsprp3

TABLE X

SHORT FUNCTION IDENTIFIERS FOR GKS-3D TRANSFORMATION FUNCTIONS

Function	Identifier
gsetwindow3	gsw3
gsetviewport3	gsv3
gsetviewind	gsvwi
gsetviewrep	gsvwr
gsetviewtraninputpri	gsvtip
gsethlhsrid	gshrid
gsethlhsrmode	gshrm
gsetwswindow3	gswkw3
gsetwsviewport3	gswkv3

TABLE XI

SHORT FUNCTION IDENTIFIERS FOR GKS-3D SEGMENT FUNCTIONS

Function	Identifier
gsetsegtran3	gssgt3
ginsertseg3	ginsg3

TABLE XII

SHORT FUNCTION IDENTIFIERS FOR GKS-3D INPUT FUNCTIONS

Function	Identifier
ginitchoice3	ginch3
ginitloc3	ginlc3
ginitpick3	ginpk3
ginitstring3	ginst3
ginitstroke3	ginsk3
greqloc3	grqlc3
greqstroke3	grqsk3
gsampleloc3	gsmlc3
gsamplestroke3	gsmsk3
ggetloc3	ggtlc3
ggetstroke3	ggtsk3

TABLE XIII

SHORT FUNCTION IDENTIFIERS FOR GKS-3D METAFILE FUNCTIONS

Function	Identifier
none	n/a

TABLE XIV

SHORT FUNCTION IDENTIFIERS FOR GKS-3D INQUIRY FUNCTIONS

Function	Identifier
ginqchoicest3	gqch3s
ginqclip3	gqclp3
ginqcurasf3	gqasf3
ginqcuredge	gqedfg
ginqcuredgecolourind	gqedci
ginqcuredgeind	gqedi
ginqcuredgetype	gqedt
qinqcuredgewidth	gqewsc
ginqcurhlhsrid	gqhrid
ginqcurpatrefpt3	gqprp3
ginqcurviewind	gqvwi
ginqdefchoice3	gqdch3
ginqdefloc3	gqdlc3
ginqdefpick3	gqdpk3
ginqdefstring3	gqdst3
ginqdefstroke3	gqdsk3
ginqdefval3	gqdvl3
ginqdisplayvolsize	gqdvol
ginqdynamicmodws3attr	gqdw3a
ginqedgefacil	gqedf
ginqedgeindices	gqeedi
ginqedgerep	gqedr
ginqgdp3	gqgdp3

TABLE XIV (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-3D INQUIRY FUNCTIONS

Function	Identifier
ginqavailgdp3	ggedg3
ginqhlhsrfacil	gqhrf
ginqhlhsrmode	gqhrm
ginqviewindices	gqevwi
ginqloc3st	gqlc3s
ginqmaxlengthwssttable	gqlw3
ginqntran3	gqnt3
ginqpick3st	gqpk3s
ginqpixel3	gqpx3
ginqpixelarray3	gqpxa3
ginqpixelarraydim3	gqpxd3
ginqprededgerep	gqpedr
ginqpredviewrep	gqpvwr
ginqsegattr3	gqsga3
ginqstring3st	gqst3s
ginqstroke3st	gqsk3s
ginqtextextent3	gqtx3
ginqval3st	gqvl3s
ginqviewfacil	gqvwf
ginqviewrep	gqvwr
ginqwstran3	gqwkt3

TABLE XV

SHORT FUNCTION IDENTIFIERS FOR GKS-3D UTILITY FUNCTIONS

Function	Identifier
gaccumtran3	gactm3
gevaltran3	gevtm3
gevalview	gevvwm

TABLE XVI

SHORT FUNCTION IDENTIFIERS FOR GKS-3D ERROR FUNCTIONS

Function	Identifier
none	n/a

TABLE XVII

SHORT FUNCTION IDENTIFIERS FOR GKS-2D CONTROL FUNCTIONS

Function	Identifier
gopengks	gopks
gclosegks	gclks
gopenws	gopws
gclosews	gclwk
gactivatews	gacwk
gdeactivatews	gdacws
gclearws	gclrwk
gredrawsegws	grsgwk
gupdatews	guwk
gsetdeferst	gsds
gmessage	gmsg
gescape	gesc

TABLE XVIII

SHORT FUNCTION IDENTIFIERS FOR GKS-2D OUTPUT FUNCTIONS

Function	Identifier
gpolyline	gpl
gpolymarker	gpm
gtext	gtx
gfillarea	gfa
gcellarray	gca

TABLE XIX

SHORT FUNCTION IDENTIFIERS FOR GKS-2D OUTPUT ATTRIBUTES FUNCTIONS

Function	Identifier
gsetlineind	gspli
gsetlinetype	gsln
gsetlinewidth	gslwsc
gsetlinecolourind	gsplci
gsetmarkerind	gspmi
gsetmarkertype	gspmt
gsetmarkersize	gsmksc
gsetmarkercolourind	gspmci
gsettextind	gstxi
gsettextfontprec	gstxfp
gsetcharexpan	gschxp
gsetcharspace	gschsp
gsettextcolourind	gstxci
gsetcharheight	gschh
gsetcharup	gschup
gsettextpath	gstxp
gsettextalign	gstxal
gsetfillind	gsfai
gsetfillintstyle	gsfais
gsetfillstyleind	gsfasi
gsetfillcolourind	gsfaci
gsetpatsize	gspa
gsetpatrefapt	gsparf

TABLE XIX (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D
OUTPUT ATTRIBUTES FUNCTIONS

Function	Identifier
gsetasf	gsasf
gsetpickid	gspkid
gsetlinerep	gsplr
gsetmarkerrep	gspmr
gsettextrep	gstxr
gsetfillrep	gsfar
gsetpatrrep	gspar
gsetcolourrep	gscr

TABLE XX

SHORT FUNCTION IDENTIFIERS FOR GKS-2D TRANSFORMATION FUNCTIONS

Function	Identifier
gsetwindow	gswn
gsetviewport	gsvp
gsetviewportinputpri	gsvpip
gselntran	gselnt
gsetclip	gsclip
gsetwswindow	gswkwn
gsetwsviewport	gwkvp

TABLE XXI

SHORT FUNCTION IDENTIFIERS FOR GKS-2D SEGMENT FUNCTIONS

Function	Identifier
gcreateseg	gcrsg
gcloseseg	gclsg
grenameseg	grensg
gdelsg	gdsg
gdelsegws	gdswk
gassocsegws	gasgwk
gcopysegws	gcsgwk
ginsertseg	ginsg
gsetsegtran	gssgt
gsetvis	gsvis
gsethighlight	gshlit
gsetsegpri	gssgp
gsetdet	gsdtec

TABLE XXII

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INPUT FUNCTIONS

Function	Identifier
ginitloc	ginlc
ginitstroke	ginsk
gintival	ginvl
ginitchoice	ginch
ginitpick	ginpk
ginitstring	ginst
gsetlocmode	gslcm
gsetstrokemode	gsskm
gsetvalmode	gsvlm
gsetchoicemode	gschm
gsetpickmode	gspkm
gsetstringmode	gsstm
greqloc	grqlc
greqstroke	grqsk
greqval	grqvl
greqchoice	grqch
greqpick	grqpk
greqstring	grqst
gsampleloc	gsmlc
gsamplestroke	gsmsk
gsampleval	gsmvl
gsamplechoice	gsmch
gsamplepick	gsmpk

TABLE XXII (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INPUT FUNCTIONS

Function	Identifier
gsamplestring	gsmst
gawaitevent	gwait
gflushevents	gflush
ggetloc	ggtlc
ggetstroke	ggtsk
ggetval	ggtvl
ggetchoice	ggtch
ggetpick	ggtpk
ggetstring	ggtst

TABLE XXIII

SHORT FUNCTION IDENTIFIERS FOR GKS-2D METAFILE FUNCTIONS

Function	Identifier
gwritegksm	gwitm
ggetttypegksm	ggtitm
greadgksm	grditm
ginterpret	giitm

TABLE XXIV

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INQUIRY FUNCTIONS

Function	Identifier
Inquiry Function for Operating State Value	
ginqopst	gqops
Inquiry Function for GKS Description Table	
ginqlevelgks	gqlvks
ginqavailwstypes	gqavwk
ginqwsmaxnum	gqwkm
ginqmaxntrannum	gqmntn
Inquiry Function for GKS State List	
ginqopenws	gqopwk
ginqactivews	gacwk
ginqlineind	gqpli
ginqmarkerind	gqpmi
ginqtextind	gqtxi
ginqcharheight	gqchh
ginqcharup	gqchup
ginqcharwidth	gqchw
ginqcharbase	gqchb
ginqtextpath	gqtxp
ginqtextalign	gqtxal
ginqfillind	gqfal
ginqpatwidth	gqpaw

TABLE XXIV (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INQUIRY FUNCTIONS

Function	Identifier
ginqpatheight	gqpah
ginqpatrefpt	gqparf
ginqpickid	gqpkid
ginqlinetype	gqln
ginqlinewidth	gqlwsc
ginqlinecolourind	gqplci
ginqmarkertype	gqmk
ginqmarkersize	gqmksc
ginqmarkercolourind	gqpmci
ginqtextfontprec	gqtxfp
ginqcharexpansion	gqchxp
ginqcharspace	gqchsp
ginqtextcolourind	gqtxcin
ginqfillintstyle	gqfais
ginqfillstyleind	gqfasi
ginqfillcolourind	gqfaci
ginqasf	gqasf
ginqcurntrannum	gqcntn
ginqntrannum	gqlntn
ginqntran	gqnt
ginqclip	gqclip
ginqnameopenseg	gqopsg
ginqsegnames	gqsgus

TABLE XXIV (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INQUIRY FUNCTIONS

Function	Identifier
ginqmoreevents	gqsim
Inquiry Functions for Workstation State List	
ginqwsconntype	gqwkc
ginqwsst	gqwks
ginqwsdeferupdatest	gqwkdu
ginqlineindices	gqlpli
ginqlinerep	gqplr
ginqmarkerindices	gqlpmi
ginqmarkerrep	gqpmr
ginqtextindices	gqltxi
ginqtextrep	gqtxr
ginqtextextent	gqtxx
ginqfillindices	gqlfai
ginqfillrep	gqfar
ginqpatindices	gqlpai
ginqpatrep	gqpar
ginqcolourindices	gqlci
ginqcolourrep	gqcr
ginqwstran	gqwkt
ginqsegnamesws	gqsgwk
ginqlocst	gqlcs
ginqstrokest	gqsks

TABLE XXIV (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D
INQUIRY FUNCTIONS

Function	Identifier
ginqvalst	gqvls
ginqchoicest	gqchs
ginqpickst	gqpks
ginqstringst	gqsts
Inquiry Functions for Workstation Description Table	
ginqwscategory	gqwkca
ginqwsclass	gqwkcl
ginqdisplaysize	gqdsp
ginqmodwsattr	gqdwka
ginqdefdeferst	gqdds
ginqlinefacil	gqplf
ginqpredlinerep	gqpplr
ginqmarkerfacil	gqpmf
ginqpredmarkerrep	gqppmr
ginqtextfacil	gqtxf
ginqpredtextrep	gqptxr
ginqfillfacil	gqfaf
ginqpredfillrep	gqpfar
ginqpatfacil	gqpaf
ginqpredpatrep	gqppar
ginqcolourfacil	gqcf
ginqpredcolourrep	gqpcr

TABLE XXIV (continued)

SHORT FUNCTION IDENTIFIERS FOR GKS-2D INQUIRY FUNCTIONS

Function	Identifier
ginqavailgdp	gqlgdp
ginqgdp	gqgdp
ginqmaxwssttables	gqlwk
ginqnumsegpri	gqsgp
ginqmodsegattr	gqdsga
ginqnumavailinput	gqli
ginqdefloc	gqdlc
ginqdefstroke	gqdsk
ginqdefval	gqdvl
ginqdefchoice	gqdch
ginqdefpick	gqdpk
ginqdefstring	gqdst
Inquiry Functions for Segment State List	
ginqassocws	gqaswk
ginqsegsttr	gqsga
ginqpixelarraydim	gqpxad
ginqpixelarray	gqpxa
ginqpixel	gqpx
ginqinputoverflow	gqiqov

TABLE XXV

SHORT FUNCTION IDENTIFIERS FOR GKS-2D UTILITY FUNCTIONS

Function	Identifier
gevaltran	gevtm
gaccumtran	gactm

TABLE XXVI

SHORT FUNCTION IDENTIFIERS FOR GKS-2D ERROR FUNCTIONS

Function	Identifier
gemergencyclosegks	geclks
gerrorhand	gerhnd
gerrorlog	gerlog

APPENDIX E

ORDERED TABLES OF GKS 3D AND 2D FUNCTIONS

TABLE XXVII

GKS-3D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-3D Function Name
gaccumtran3	Accumulate Transformation Matrix 3
gcellarray3	Cell Array 3
gevaltran3	Evaluate Transformation Matrix 3
gevalview	Evaluate View Matrix
gfillarea3	Fill Area 3
gfillareaset	Fill Area Set
gfillareaset3	Fill Area Set 3
ggdp3	Generalized Drawing Primitive 3
ggetloc3	Get Locator 3
ggetstroke3	Get Stroke 3
ginitchoice3	Initialise Choice 3
ginitloc3	Initialise Locator 3
ginitpick3	Initialise Pick 3
ginitstring3	Initialise String 3
ginitstroke3	Initialise Stroke 3
ginitval3	Initialise Valuator 3
ginqavailgdp3	Inquire List of Available Generalized Drawing Primitives 3
ginqchoicest3	Inquire Choice 3 Device State
ginqclip3	Inquire Clipping 3
ginqcurasf3	Inquire Current Aspect Source Flags 3

TABLE XXVII (continued)

GKS-3D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-3D Function Name
ginqcuredge	Inquire Current Edge Flag
ginqcuredgecolourind	Inquire Current Edge Colour Index
ginqcuredgeind	Inquire Current Edge Index
ginqcuredgetype	Inquire Current Edgetype
ginqcuredgewidth	Inquire Current Edgewidth Scale Factor
ginqcurhlhsrid	Inquire Current HLHSR Identifier
ginqcurpatrefpt3	Inquire Current Pattern Reference Points 3
ginqcurviewind	Inquire Current View Index
ginqdefchoice3	Inquire Default Choice 3 Device Data
ginqdefloc3	Inquire Default Locator 3 Device Data
ginqdefpick3	Inquire Default Pick 3 Device Data
ginqdefstring3	Inquire Default String 3 Device Data
ginqdefstroke3	Inquire Default Stroke 3 Device Data
ginqdefval3	Inquire Default Valuator 3 Device Data
ginqdisplayvolsize	Inquire Display Volume Size
ginqdynamicmodws3attr	Inquire Dynamic Modification of Workstation 3 Attributes
ginqedgefacil	Inquire Edge Facilities
ginqedgeindices	Inquire List of Edge Indices
ginqedgerep	Inquire Edge Representation

TABLE XXVII (continued)

GKS-3D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-3D Function Name
ginqgdp3	Inquire Generalized Drawing Primitive 3
ginqhlhsrfacil	Inquire HLHSR Facilities
ginqhlhsrmode	Inquire HLHSR Mode
ginqloc3st	Inquire Locator 3 Device State
ginqmaxlengthwssttable	Inquire Maximum Length of GKS-3D Workstation State Tables
ginqntran3	Inquire Normalization Transformation 3
ginqpick3st	Inquire Pick 3 Device State
ginqpixel3	Inquire Pixel 3
ginqpixelarray3	Inquire Pixel Array 3
ginqpixelarraydim3	Inquire Pixel Array Dimenstions 3
ginqprededgerep	Inquire Predefined Edge Representation
ginqpredviewrep	Inquire Predefined View Representation
ginqsegattr3	Inquire Segment Attributes 3
ginqstring3st	Inquire String 3 Device State
ginqstroke3st	Inquire Stroke 3 Device State
ginqtextextent3	Inquire Text Extent 3
ginqval3st	Inquire Valuator 3 Device State
ginqviewfacil	Inquire View Facilities
ginqviewindices	Inquire List of View Indices
ginqviewrep	Inquire View Representation

TABLE XXVII (continued)

GKS-3D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-3D Function Name
ginqwstran3	Inquire Workstation Transformation 3
ginsertseg3	Insert Segment 3
gpolyline3	Polyline 3
gpolymarker3	Polymarker 3
greqloc3	Request Locator 3
greqstroke3	Request Stroke 3
gsampleloc3	Sample Locator 3
gsamplestroke3	Sample Stroke 3
gsetasf3	Set Aspect Source Flags 3
gsetedgecolourind	Set Edge Colour Index
gsetedgeflag	Set Edge Flag
gsetedgeind	Set Edge Index
gsetedgerep	Set Edge Representation
gsetedgetype	Set Edgetype
gsetedgewidth	Set Edgewidth Scale Factor
gsethlhsrid	Set HLHSR Identifier
gsethlhsrmode	Set HLHSR Mode
gsetpatrefpt3	Set Pattern Reference Points 3
gsetsegtran3	Set Segment Transformation 3
gsetviewind	Set View Index
gsetviewport3	Set Viewport 3
gsetviewrep	Set View Representation

TABLE XXVII (continued)

GKS-3D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-3D Function Name
gsetviewtraninputpri	Set View Transformation Input Priority
gsetwindow3	Set Window 3
gsetwsviewport3	Set Workstation Viewport 3
gsetwswindow3	Set Workstation Window 3
gtext3	Text 3

TABLE XXVIII

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
gaccumtran	accumulate transformation matrix
gactivatews	activate workstation
gassocsegws	associate segment with workstation
gawaitevent	await event
gcellarray	cell array
gclearws	clear workstation
gclosegks	close gks
gcloseseg	close segment
gclosews	close workstation
gcopysegws	copy segment to workstation
gcreateseg	create segment
gdeactivatews	deactivate workstation
gdelseg	delete segment
gdelsegws	delete segment from workstation
gemergencyclosegks	emergency close gks
gerrorhand	error handling
gerrorlog	error logging
gescape	escape
gevaltran	evaluate transformation matrix
gfillarea	fill area
gflushevents	flush device events
ggdp	generalized drawing primitive
ggetchoice	get choice
ggettypegksm	get item type from gksm

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ggetloc	get locator
ggetpick	get pick
ggetstring	get string
ggetstroke	get stroke
ggetval	get valuator
ginitchoice	initialise choice
ginitloc	initialise locator
ginitpick	initialise pick
ginitstring	initialise string
ginitstroke	initialise stroke
ginitval	initialise valuator
ginqactivews	inquire set of active workstations
ginqasf	inquire aspect source flags
ginqassocws	inquire set of associate workstations
ginqavailgdp	inquire list of available generalized drawing primitives
ginqavailwstypes	inquire list of available workstation types
ginqcharbase	inquire character base vector
ginqcharexpan	inquire character expansion factor
ginqcharheight	inquire character height
ginqcharspace	inquire character spacing
ginqcharup	inquire character up vector
ginqcharwidth	inquire character width

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ginqchoicest	inquire choice device state
ginqclip	inquire clipping indicator
ginqcolourfacil	inquire colour facilities
ginqcolourindices	inquire list of colour indices
ginqcurntrannum	inquire current normalization transformation number
ginqdefchoice	inquire default choice device data
ginqdefdeferst	inquire default deferral state values
ginqdefloc	inquire default locator device data
ginqdefpick	inquire default pick device data
ginqdefstring	inquire default string device data
ginqdefstroke	inquire default stroke device data
ginqdefval	inquire default valuator device data
ginqdisplaysize	inquire display surface size
ginqfillcolourind	inquire fill area colour index
ginqfillfacil	inquire fill area facilities
ginqfillind	inquire fill area index
ginqfillindices	inquire list of fill area indices
ginqfillintstyle	inquire fill area interior style
ginqfillrep	inquire fill area representation
ginqfilstyleind	inquire fill area style index
ginqdgp	inquire generalized drawing primitive

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ginqindivattr	inquire current individual attributes
ginqinputoverflow	inquire input queue overflow
ginqlevelgks	inquire level of gks
ginqlinecolourind	inquire polyline colour index
ginqlinefacil	inquire polyline facilities
ginqlineind	inquire polyline index
ginqlineindices	inquire list of polyline indices
ginqlinerep	inquire polyline representation
ginqlinetype	inquire linetype
ginqlinewidth	inquire linewidth scale factor
ginqlocst	inquire locator device state
ginqmarkercolourind	inquire polymarker colour index
ginqmarkerfacil	inquire polymarker facilities
ginqmarkerind	inquire polymarker index
ginqmarkerindices	inquire list of polymarker indices
ginqmarkerrep	inquire polymarker representation
ginqmarkersize	inquire marker size scale factor
ginqmarkertype	inquire marker type
ginqmaxntrannum	inquire maximum normalization transformation number
ginqmaxwssttables	inquire maximum length of workstation state tables
ginqmodsegattr	inquire dynamic modification of segment attributes

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ginqmodwsattr	inquire dynamic modification of workstation attributes
ginqmoreevents	inquire more simultaneous events
ginqnameopenseg	inquire name of open segment
ginqntrannum	inquire list of normalization transformation numbers
ginqntran	inquire normalization transformation
ginqnumavailinput	inquire number of available logical input devices
ginqnumsegpri	input number of segment priorities supported
ginqopenws	inquire set of open workstations
ginqopst	inquire operating state value
ginqpatfacil	inquire pattern facilities
ginqpatindices	inquire list of pattern indices
ginqpatrefpt	inquire pattern reference point
ginqpatrep	inquire pattern representation
ginqpatheight	inquire pattern height vector
ginqpatwidth	inquire pattern width vector
ginqpickid	inquire pick identifier
ginqpickst	inquire pick device state
ginqpixel	inquire pixel
ginqpixelarray	inquire pixel array
ginqpixelarraydim	inquire pixel array dimensions

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ginqpredcolourrep	inquire predefined colour representation
ginqpredfillrep	inquire predefined fill area representation
ginqpredlinerep	inquire predefined polyline representation
ginqpredmarkerrep	inquire predefined polymarker representation
ginqpredpatrep	inquire predefined pattern representation
ginqpredtextrep	inquire predefined text representation
ginqprimattr	inquire current primitive attribute values
ginqsegattr	inquire segment attributes
ginqsegnames	inquire set of segment names in use
ginqsegnamesws	inquire set of segment names on workstation
ginqstringst	inquire string device state
ginqstokest	inquire stroke device state
ginqtextalign	inquire text alignment
ginqtextcolourind	inquire text colour index
ginqtextextent	inquire text extent
ginqtextfacil	inquire text facilities
ginqtextfontprec	inquire text font and precision
ginqtextind	inquire text index
ginqtextindices	inquire list of text indices

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
ginqtextpath	inquire text path
ginqtextrep	inquire text representation
ginqvalst	inquire valuator device state
ginqwscategory	inquire workstation category
ginqwsclass	inquire workstation classification
ginqwsconntype	inquire workstation connection and type
ginqwsdeferupdatest	inquire workstation deferral and update states
ginqwsmaxnum	inquire workstation maximum number
ginqwsst	inquire workstation state
ginqwstran	inquire workstation transformation
ginsertseg	insert segment
ginterpret	interpret item
gmessage	message
gopengks	open gks
gopenwk	open workstation
gpolyline	polyline
gpolymarker	polymarker
greadgksm	read item from gksm
gredrawsegws	redraw all segments on workstation
grenameseg	rename segment
greqchoice	request choice
greqloc	request locator

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
greqpick	request pick
greqstring	request string
greqstroke	request stroke
greqval	request valuator
gsamplechoice	sample choice
gsampleloc	sample locator
gsamplepick	sample pick
gsamplestring	sample string
gsamplestroke	sample stroke
gsampleval	sample valuator
gselntran	select normalization transformation
gsetasf	set aspect source flags
gsetcharexpan	set character expansion factor
gsetcharheight	set character height
gsetcharspace	set character spacing
gsetcharup	set character up vector
gsetchoicemode	set choice mode
gsetclip	set clipping indicator
gsetcolourrep	set colour representation
gsetdeferst	set deferral state
gsetdet	set detectability
gsetfillcolourind	set fill area colour index
gsetfillind	set fill area index

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
gsetfillintstyle	set fill area interior style
gsetfillrep	set fill area representation
gsetfillstyleind	set fill area style index
gsethighlight	set highlighting
gsetlinecolourind	set polyline colour index
gsetlineind	set polyline index
gsetlinerep	set polyline representation
gsetlinetype	set linetype
gsetlinewidth	set linewidth scale factor
gsetlocmode	set locator mode
gsetmarkercolourind	set polymarker colour index
gsetmarkerind	set polymarker index
gsetmarkerrep	set polymarker representation
gsetmarkersize	set marker size scale factor
gsetmarkertype	set marker type
gsetpatrefpt	set pattern reference point
gsetpatrep	set pattern representation
gsetpatsize	set pattern size
gsetpickid	set pick identifier
gsetpickmode	set pick mode
gsetsegpri	set segment priority
gsetsegtran	set segment transformation
gsetstringmode	set string mode

TABLE XXVIII (continued)

GKS-2D FUNCTION NAMES ORDERED BY BOUND NAME

Bound Name	GKS-2D Function Name
gsetstrokemode	set stroke mode
gsettextalign	set text alignment
gsettextcolourind	set text colour index
gsettextfontprec	set text font and precision
gsettextind	set text index
gsettextpath	set text path
gsettextrep	set text representation
gsetvalmode	set valuator mode
gsetviewport	set viewport
gsetviewportinputpri	set viewport input priority
gsetvis	set visibility
gsetwindow	set window
gsetwsviewport	set workstation viewport
gsetwswindow	set workstation window
gtext	text
gupdatews	update workstation
gwritegksm	write item to gksm

TABLE XXIX

GKS-3D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-3D Function Name
gaccumtran3	accumulate transformation matrix 3
gcellarray3	cell array 3
gevaltran3	evaluate transformation matrix 3
gevalview	evaluate view matrix
gfillarea3	fill area 3
gfillareaset	fill area set
gfillareaset3	fill area set 3
ggdp3	generalized drawing primitive 3
ggetloc3	get locator 3
ggetstroke3	get stroke 3
ginitchoice3	initialise choice 3
ginitloc3	initialise locator 3
ginitpick3	initialise pick 3
ginitstring3	initialise string 3
ginitstroke3	initialise stroke 3
ginitval3	initialise valuator 3
ginqchoice3st	inquire choice 3 device state
ginqclip3	inquire clipping 3
ginqcurasf3	inquire current aspect source flags 3
ginqcuredgecolourind	inquire current edge colour index
ginqcuredge	inquire current edge flag
ginqcuredgeind	inquire current edge index
ginqcuredgetype	inquire current edgetype

TABLE XXIX (continued)

GKS-3D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-3D Function Name
ginqcuredgewidth	inquire current edgewidth Scale factor
ginqcurhlhsrid	inquire current HLHSR identifier
ginqcurpatrefpt3	inquire current pattern reference points 3
ginqcurviewind	inquire current view index
ginqdefchoice3	inquire default choice 3 device data
ginqdefloc3	inquire default locator 3 device data
ginqdefpick3	inquire default pick 3 device data
ginqdefstring3	inquire default string 3 device data
ginqdefstroke3	inquire default stroke 3 device data
ginqdefval3	inquire default valuator 3 device data
ginqdisplayvolsize	inquire display volume size
ginqdynamicmodws3attr	inquire dynamic modification of workstation 3 attributes
ginqedgefacil	inquire edge facilities
ginqedgerep	inquire edge representation
ginqgdp3	inquire generalized drawing primitive 3
ginqhlhsrfacil	inquire HLHSR facilities
ginqhlhsrmode	inquire HLHSR mode
ginqavailgdp3	inquire list of available generalized drawing primitives 3

TABLE XXIX (continued)

GKS-3D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-3D Function Name
ginqedgeindices	inquire list of edge indices
ginqviewindices	inquire list of view indices
ginqloc3st	inquire locator 3 device state
ginqmaxlengthwssttable	inquire maximum length of GKS-3D workstation state tables
ginqntran3	inquire normalization transformation 3
ginqpick3st	inquire pick 3 device state
ginqpixel3	inquire pixel 3
ginqpixelarray3	inquire pixel array 3
ginqpixelarraydim3	inquire pixel array dimensions 3
ginqprededgerep	inquire predefined edge representation
ginqpredviewrep	inquire predefined view representation
ginqsegattr3	inquire segment attributes 3
ginqstring3st	inquire string 3 device state
ginqstroke3st	inquire stroke 3 device state
ginqtextextent3	inquire text extent 3
ginqval3st	inquire valuator 3 device state
ginqviewfacil	inquire view facilities
ginqviewrep	inquire view representation
ginqwstran3	inquire workstation transformation 3
ginsertseg	insert segment 3
gpolyline3	polyline 3

TABLE XXIX (continued)

GKS-3D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-3D Function Name
gpolymarker3	polymarker 3
greqloc3	request locator 3
greqstroke3	request stroke 3
gsampleloc3	sample locator 3
gsamplestroke3	sample stroke 3
gsetasf3	set aspect source flags 3
gsetedgecolourind	set edge colour index
gsetedgeflag	set edge flag
gsetedgeind	set edge index
gsetedgerep	set edge representation
gsetedgetype	set edgetype
gsetedgewidth	set edgewidth scale factor
gsethlhsrid	set HLHSR identifier
gsethlhsrmode	set HLHSR mode
gsetpatrefpt3	set pattern reference points 3
gsetsegtran3	set segment transformation 3
gsetviewind	set view index
gsetviewrep	set view representation
gsetviewtraninputpri	set view transformation input priority
gsetviewport3	set viewport 3
gsetwindow3	set window 3
gsetwsviewport3	set workstation viewport 3
gsetwswindow3	set workstation window 3

TABLE XXIX (continued)

GKS-3D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-3D Function Name
gtext3	text 3

TABLE XXX

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
gaccumtran	accumulate transformation matrix
gactivatews	activate workstation
gassocsegws	associate segment with workstation
gawaitevent	await event
gcellarray	cell array
gclearws	clear workstation
gclosegks	close gks
gcloseseg	close segment
gclosews	close workstation
gcopysegws	copy segment to workstation
gcreateseg	create segment
gdeactivatews	deactivate workstation
gdelseg	delete segment
gdelsegws	delete segment from workstation
gemergencyclosegks	emergency close gks
gerrorhand	error handling
gerrorlog	error logging
gescape	escape
gevaltran	evaluate transformation matrix
gfillarea	fill area
gflushevents	flush device events
ggdp	generalized drawing primitive
ggetchoice	get choice
ggettypegksm	get item type from gksm

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ggetloc	get locator
ggetpick	get pick
ggetstring	get string
ggetstroke	get stroke
ggetval	get valuator
ginitchoice	initialise choice
ginitloc	initialise locator
ginitpick	initialise pick
ginitstring	initialise string
ginitstroke	initialise stroke
ginitval	initialise valuator
ginqasf	inquire aspect source flags
ginqcharbase	inquire character base vector
ginqcharexpan	inquire character expansion factor
ginqcharheight	inquire character height
ginqcharspace	inquire character spacing
ginqcharup	inquire character up vector
ginqcharwidth	inquire character width
ginqchoicest	inquire choice device state
ginqclip	inquire clipping indicator
ginqcolourfacil	inquire colour facilities
ginqcolourrep	inquire colour representation
ginqcurntrannum	inquire current normalization transformation number

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ginqindivattr	inquire current individual attributes values
ginqprimattr	inquire current primitive attribute values
ginqdefchoice	inquire default choice device data
ginqdefdeferst	inquire default deferral state values
ginqdefloc	inquire default locator device data
ginqdefpick	inquire default pick device data
ginqdefstring	inquire default string device data
ginqdefstroke	inquire default stroke device data
ginqdefval	inquire default valuator device data
ginqdisplaysize	inquire display surface size
ginqmodsegattr	inquire dynamic modification of segment attributes
ginqmodwsattr	inquire dynamic modification of workstation attributes
ginqfillcolourind	inquire fill area colour index
ginqfillfacil	inquire fill area facilities
ginqfillind	inquire fill area index
ginqfillintstyle	inquire fill area interior style
ginqfillrep	inquire fill area representation
ginqfilstyleind	inquire fill area style index
ginqdgp	inquire generalized drawing primitive

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ginqinputoverflow	inquire input queue overflow
ginqlevelgks	inquire level of gks
ginqlinetype	inquire linetype
ginqlinewidth	inquire linewidth scale factor
ginqavailgdp	inquire list of available generalized drawing primitives
ginqavailwstypes	inquire list of available workstation types
ginqcolourindices	inquire list of colour indices
ginqfillindices	inquire list of fill area indices
ginqntrannum	inquire list of normalization transformation numbers
ginqpatindices	inquire list of pattern indices
ginqlineindices	inquire list of polyline indices
ginqmarkerindices	inquire list of polymarker indices
ginqtextindices	inquire list of text indices
ginqlocst	inquire locator device state
ginqmarkersize	inquire marker size scale factor
ginqmarkertype	inquire marker type
ginqmaxwssttables	inquire maximum length of workstation state tables
ginqmaxntrannum	inquire maximum normalization transformation number
ginqmoreevents	inquire more simultaneous events
ginqnameopenseg	inquire name of open segment

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ginqntran	inquire normalization transformation
ginqnumavailinput	inquire number of available logical input devices
ginqnumsegpri	input number of segment priorities supported
ginqopst	inquire operating state value
ginqpatfacil	inquire pattern facilities
ginqpatrefpt	inquire pattern reference point
ginqpatrep	inquire pattern representation
ginqpatheight	inquire pattern height vector
ginqpatwidth	inquire pattern width vector
ginqpickst	inquire pick device state
ginqpickid	inquire pick identifier
ginqpixel	inquire pixel
ginqpixelarray	inquire pixel array
ginqpixelarraydim	inquire pixel array dimensions
ginqlinecolourind	inquire polyline colour index
ginqlinefacil	inquire polyline facilities
ginqlineind	inquire polyline index
ginqlinerep	inquire polyline representation
ginqmarkercolourind	inquire polymarker colour index
ginqmarkerfacil	inquire polymarker facilities
ginqmarkerind	inquire polymarker index
ginqmarkerrep	inquire polymarker representation

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ginqpredcolourrep	inquire predefined colour representation
ginqpredfillrep	inquire predefined fill area representation
ginqpredpatrep	inquire predefined pattern representation
ginqpredlinerep	inquire predefined polyline representation
ginqpredmarkerrep	inquire predefined polymarker representation
ginqpredtextrep	inquire predefined text representation
ginqsegattr	inquire segment attributes
ginqactivews	inquire set of active workstations
ginqassocws	inquire set of associated workstations
ginqopenws	inquire set of open workstations
ginqsegnames	inquire set of segment names in use
ginqsegnamesws	inquire set of segment names on workstation
ginqstringst	inquire string device state
ginqstokest	inquire stroke device state
ginqtextalign	inquire text alignment
ginqtextcolourind	inquire text colour index
ginqtextextent	inquire text extent
ginqtextfacil	inquire text facilities
ginqtextfontprec	inquire text font and precision

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
ginqtextind	inquire text index
ginqtextpath	inquire text path
ginqtextrep	inquire text representation
ginqvalst	inquire valuator device state
ginqwscategory	inquire workstation category
ginqwsclass	inquire workstation classification
ginqwsconntype	inquire workstation connection and type
ginqwsdeferupdatest	inquire workstation deferral and update states
ginqwsmaxnum	inquire workstation maximum number
ginqwstran	inquire workstation transformation
ginqwsst	inquire workstation state
ginsertseg	insert segment
ginterpret	interpret item
gmessage	message
gopengks	open gks
gopenwk	open workstation
gpolyline	polyline
gpolymarker	polymarker
greadgksm	read item from gksm
gredrawsegws	redraw all segments on workstation
grenameseg	rename segment
greqchoice	request choice

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
greqloc	request locator
greqpick	request pick
greqstring	request string
greqstroke	request stroke
greqval	request valuator
gsamplechoice	sample choice
gsampleloc	sample locator
gsamplepick	sample pick
gsamplestring	sample string
gsamplestroke	sample stroke
gsampleval	sample valuator
gselntran	select normalization transformation
gsetasf	set aspect source flags
gsetcharexpan	set character expansion factor
gsetcharheight	set character height
gsetcharspace	set character spacing
gsetcharup	set character up vector
gsetchoicemode	set choice mode
gsetclip	set clipping indicator
gsetcolourrep	set colour representation
gsetdeferst	set deferral state
gsetdet	set detectability
gsetfillcolourind	set fill area colour index

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
gsetfillind	set fill area index
gsetfillintstyle	set fill area interior style
gsetfillrep	set fill area representation
gsetfillstyleind	set fill area style index
gsethighlight	set highlighting
gsetlinetype	set linetype
gsetlinewidth	set linewidth scale factor
gsetlocmode	set locator mode
gsetmarkersize	set marker size scale factor
gsetmarkertype	set marker type
gsetpatrefpt	set pattern reference point
gsetpatrep	set pattern representation
gsetpatsize	set pattern size
gsetpickid	set pick identifier
gsetpickmode	set pick mode
gsetlinecolourind	set polyline colour index
gsetlineind	set polyline index
gsetlinerep	set polyline representation
gsetmarkercolourind	set polymarker colour index
gsetmarkerind	set polymarker index
gsetmarkerrep	set polymarker representation
gsetsegpri	set segment priority
gsetsegtran	set segment transformation

TABLE XXX (continued)

GKS-2D FUNCTION NAMES ORDERED BY FUNCTION NAME

Bound Name	GKS-2D Function Name
gsetstringmode	set string mode
gsetstrokemode	set stroke mode
gsettextalign	set text alignment
gsettextcolourind	set text colour index
gsettextfontprec	set text font and precision
gsettextind	set text index
gsettextpath	set text path
gsettextrep	set text representation
gsetvalmode	set valuator mode
gsetviewport	set viewport
gsetviewportinputpri	set viewport input priority
gsetvis	set visibility
gsetwindow	set window
gsetwsviewport	set workstation viewport
gsetwswindow	set workstation window
gtext	text
gupdatews	update workstation
gwritegksm	write item to gksm

APPENDIX F

GKS 3D AND 2D FUNCTIONS ORDERED BY LEVEL

TABLE XXXI

GKS-3D LEVEL 0a FUNCTIONS

Bound Name	GKS-3D Function Name
gcellarray3	cell array 3
gevalview	evaluate view matrix
gfillarea3	fill area 3
gfillareaset	fill area set
gfillareaset3	fill area set 3
ggdp3	generalized drawing primitive 3
ginqclip3	inquire clipping 3
ginqcurasf3	inquire current aspect source flags 3
ginqcuredgecolourind	inquire current edge colour index
ginqcuredge	inquire current edge flag
ginqcuredgeind	inquire current edge index
ginqcuredgetype	inquire current edgetype
ginqcuredgewidth	inquire current edgewidth scale factor
ginqcurcurpatrefpt3	inquire current pattern reference points 3
ginqcurviewind	inquire current view index
ginqdisplayvolsize	inquire display volume size
ginqdynamicmodws3attr	inquire dynamic modification of workstation 3 attributes
ginqedgefacil	inquire edge facilities

TABLE XXXI (continued)

GKS-3D LEVEL 0a FUNCTIONS

Bound Name	GKS-3D Function Name
ginqgdp3	inquire generalized drawing primitive 3
ginqavailgdp3	inquire list of available generalized drawing primitives 3
ginqviewindices	inquire list of view indices
ginqmaxlengthwssttable	inquire maximum length of GKS-3D workstation state tables
ginqntran3	inquire normalization transformation 3
ginqpixel3	inquire pixel 3
ginqpixelarray3	inquire pixel array 3
ginqpixelarraydim3	inquire pixel array dimensions 3
ginqprededgerep	inquire predefined edge representation
ginqpredviewrep	inquire predefined view representation
ginqtextextent3	inquire text extent 3
ginqviewfacil	inquire view facilities
ginqviewrep	inquire view representation
ginqwstran3	inquire workstation transformation 3
gpolyline3	polyline 3
gpolymarker3	polymarker 3
gsetasf3	set aspect source flags 3
gsetedgecolourind	set edge colour index
gsetedgeflag	set edge flag
gsetedgeind	set edge index

TABLE XXXI (continued)

GKS-3D LEVEL 0a FUNCTIONS

Bound Name	GKS-3D Function Name
gsetedgetype	set edgetype
gsetedgewidth	set edgewidth scale factor
gsetpatrefpt3	set pattern reference points 3
gsetviewind	set view index
gsetviewrep	set view representation
gsetviewtraninputpri	set view transformation Input priority
gsetviewport3	set viewport 3
gsetwindow3	set window 3
gsetwsviewport3	set workstation viewport 3
gsetwswindow3	set workstation window 3
gtext3	text 3

TABLE XXXII

GKS-3D LEVEL 0b FUNCTIONS

Bound Name	GKS-3D Function Name
ginitchoice3	initialise choice 3
ginitloc3	initialise locator 3
ginitpick3	initialise pick 3
ginitstring3	initialise string 3
ginitstroke3	initialise stroke 3
ginitval3	initialise valuator 3
ginqchoice3st	inquire choice 3 device state
ginqdefchoice3	inquire default choice 3 device data
ginqdefloc3	inquire default locator 3 device data
ginqdefstring3	inquire default string 3 device data
ginqdefstroke3	inquire default stroke 3 device data
ginqdefval3	inquire default valuator 3 device data
ginqloc3st	inquire locator 3 device state
ginqstring3st	inquire string 3 device state
ginqstroke3st	inquire stroke 3 device state
ginqval3st	inquire valuator 3 device state
greqloc3	request locator 3
greqstroke3	request stroke 3

TABLE XXXIII

GKS-3D LEVEL 0c FUNCTIONS

Bound Name	GKS-3D Function Name
ggetloc3	get locator 3
ggetstroke3	get stroke 3
gsampleloc3	sample locator 3
gsamplestroke3	sample stroke 3

TABLE XXXIV

GKS-3D LEVEL 1a FUNCTIONS

Bound Name	GKS-3D Function Name
gaccumtran3	accumulate transformation matrix 3
gevaltran3	evaluate transformation matrix 3
ginqcurhlhsrid	inquire current HLHSR identifier
ginqedgerep	inquire edge representation
ginqhlhsrfacil	inquire HLHSR facilities
ginqhlhsrmode	inquire HLHSR mode
ginqedgeindices	inquire list of edge indices
ginqsegattr3	inquire segment attributes 3
gsetedgerep	set edge representation
gsethlhsrid	set HLHSR identifier
gsethlhsrmode	set HLHSR mode
gsetsegtran3	set segment transformation 3

TABLE XXXV

GKS-3D LEVEL 1b FUNCTIONS

Bound Name	GKS-3D Function Name
ginqdefpick3	inquire default pick 3 device data
ginqpick3st	inquire pick 3 device state

TABLE XXXVI

GKS-3D LEVEL 2a FUNCTIONS

Bound Name	GKS-3D Function Name
ginsertseg	insert segment 3

TABLE XXXVII

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
gactivatews	activate workstation
gcellarray	cell array
gclearws	clear workstation
gclosegks	close gks
gclosews	close workstation
gdeactivatews	deactivate workstation
gemergencyclosegks	emergency close gks
gerrorhand	error handling
gerrorlog	error logging
gescape	escape
gfillarea	fill area
ggdp	generalized drawing primitive
ggettypegksm	get item type from gksm
ginqasf	inquire aspect source flags
ginqavailgdp	inquire list of available generalized drawing primitives
ginqavailwstypes	inquire list of available workstation types
ginqcharbase	inquire character base vector
ginqcharexpan	inquire character expansion factor
ginqcharheight	inquire character height
ginqcharspace	inquire character spacing
ginqcharup	inquire character up vector
ginqcharwidth	inquire character width
ginqclip	inquire clipping indicator

TABLE XXXVII (continued)

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
ginqcolourfacil	inquire colour facilities
ginqcolourindices	inquire list of colour indices
ginqcurntrannum	inquire current normalization transformation number
ginqdgp	inquire generalized drawing primitive
ginqdisplaysize	inquire display surface size
ginqfillcolourind	inquire fill area colour index
ginqfillfacil	inquire fill area facilities
ginqfillind	inquire fill area index
ginqfillintstyle	inquire fill area interior style
ginqfilstyleind	inquire fill area style index
ginqindivattr	inquire current individual attributes
ginqlevelgks	inquire level of gks
ginqlinecolourind	inquire polyline colour index
ginqlinefacil	inquire polyline facilities
ginqlineind	inquire polyline index
ginqlinetype	inquire linetype
ginqlinewidth	inquire linewidth scale factor
ginqmarkercolourind	inquire polymarker colour index
ginqmarkerfacil	inquire polymarker facilities
ginqmarkerindices	inquire list of polymarker indices
ginqmarkersize	inquire marker size scale factor
ginqmarkertype	inquire marker type

TABLE XXXVII (continued)

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
ginqmaxntrannum	inquire maximum normalization transformation number
ginqmaxwssttables	inquire maximum length of workstation state tables
ginqntran	inquire normalization transformation
ginqntrannum	inquire list of normalization transformation numbers
ginqopenws	inquire set of open workstations
ginqopst	inquire operating state value
ginqpatfacil	inquire pattern facilities
ginqpatheight	inquire pattern height vector
ginqpatrefpt	inquire pattern reference point
ginqpatwidth	inquire pattern width vector
ginqpixel	inquire pixel
ginqpixelarray	inquire pixel array
ginqpixelarraydim	inquire pixel array dimensions
ginqpredcolourrep	inquire predefined colour representation
ginqpredfillrep	inquire predefined fill area representation
ginqpredlinerep	inquire predefined polyline representation
ginqpredmarkerrep	inquire predefined polymarker representation
ginqpredpatrep	inquire predefined pattern representation

TABLE XXXVII (continued)

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
ginqpredtextrep	inquire predefined text representation
ginqprimattr	inquire current primitive attribute values
ginqtextalign	inquire text alignment
ginqtextcolourind	inquire text colour index
ginqtextextent	inquire text extent
ginqtextfacil	inquire text facilities
ginqtextfontprec	inquire text font and precision
ginqtextind	inquire text index
ginqtextpath	inquire text path
ginqwscategory	inquire workstation category
ginqwsclass	inquire workstation classification
ginqwsconntype	inquire workstation connection and type
ginqwsdeferupdatest	inquire workstation deferral and update states
ginqwsst	inquire workstation transformation
ginterpret	interpret item
gopengks	open gks
gopenwk	open workstation
gpolyline	polyline
gpolymarker	polymarker
greadgksm	read item from gksm
gselntran	select normalization transformation

TABLE XXXVII (continued)

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
gsetasf	set aspect source flags
gsetcharexpan	set character expansion factor
gsetcharheight	set character height
gsetcharspace	set character spacing
gsetcharup	set character up vector
gsetclip	set clipping indicator
gsetcolourrep	set colour representation
gsetfillcolourind	set fill area colour index
gsetfillind	set fill area index
gsetfillintstyle	set fill area interior style
gsetfillstyleind	set fill area style index
gsetlinecolourind	set polyline colour index
gsetlineind	set polyline index
gsetlinetype	set linetype
gsetlinewidth	set linewidth scale factor
gsetmarkercolourind	set polymarker colour index
gsetmarkerind	set polymarker index
gsetmarkersize	set marker size scale factor
gsetmarkertype	set marker type
gsetpatrefpt	set pattern reference point
gsetpatsize	set pattern size
gsettextalign	set text alignment
gsettextcolourind	set text colour index

TABLE XXXVII (continued)

GKS-2D LEVEL 0a FUNCTIONS

Bound Name	GKS-2D Function Name
gsettextfontprec	set text font and precision
gsettextind	set text index
gsettextpath	set text path
gsetviewport	set viewport
gsetwindow	set window
gsetwsviewport	set workstation viewport
gsetwswindow	set workstation window
gtext	text
gupdatews	update workstation
gwritegksm	write item to gksm

TABLE XXXVIII

GKS-2D LEVEL 0b FUNCTIONS

Bound Name	GKS-2D Function Name
ginitchoice	initialise choice
ginitloc	initialise locator
ginitstring	initialise string
ginitstroke	initialise stroke
ginitval	initialise valuator
ginqchoicest	inquire choice device state
ginqdefchoice	inquire default choice device data
ginqdefloc	inquire default locator device data
ginqdefstring	inquire default string device data
ginqdefstroke	inquire default stroke device data
ginqdefval	inquire default valuator device data
ginqlocst	inquire locator device state
ginqnumavailinput	inquire number of available logical input devices
gingstokest	inquire stroke device state
ginqstringst	inquire string device state
ginqvalst	inquire valuator device state
greqchoice	request choice
greqloc	request locator
greqstring	request string
greqstroke	request stroke
greqval	request valuator
gsetchoicemode	set choice mode

TABLE XXXVIII (continued)

GKS-2D LEVEL 0b FUNCTIONS

Bound Name	GKS-2D Function Name
gsetlocmode	set locator mode
gsetstringmode	set string mode
gsetstrokemode	set stroke mode
gsetvalmode	set valuator mode
gsetviewportinputpri	set viewport input priority

TABLE XXXIX

GKS-2D LEVEL 0c FUNCTIONS

Bound Name	GKS-2D Function Name
gawaitevent	await event
gflushevents	flush device events
ggetchoice	get choice
ggetloc	get locator
ggetstring	get string
ggetstroke	get stroke
ggetval	get valuator
ginqinputoverflow	inquire input queue overflow
ginqmoreevents	inquire more simultaneous events
gsamplechoice	sample choice
gsampleloc	sample locator
gsamplestring	sample string
gsamplestroke	sample stroke
gsampleval	sample valuator

TABLE XL

GKS-2D LEVEL 1a FUNCTIONS

Bound Name	GKS-2D Function Name
gaccumtran	accumulate transformation matrix
gcloseseg	close segment
gcreateseg	create segment
gdelseg	delete segment
gdelsegws	delete segment from workstation
gevaltran	evaluate transformation matrix
ginqactivews	inquire set of active workstations
ginqassocws	inquire set of associate workstations
ginqdefdeferst	inquire default deferral state values
ginqfillindices	inquire list of fill area indices
ginqfillrep	inquire fill area representation
ginqlineindices	inquire list of polyline indices
ginqlinerep	inquire polyline representation
ginqmarkerind	inquire polymarker index
ginqmarkerrep	inquire polymarker representation
ginqmodsegattr	inquire dynamic modification of segment attributes
ginqmodwsattr	inquire dynamic modification of workstation attributes
ginqnameopenseg	inquire name of open segment
ginqnumsegpri	input number of segment priorities supported
ginqpatindices	inquire list of pattern indices
ginqpatrep	inquire pattern representation

TABLE XL (continued)

GKS-2D LEVEL 1a FUNCTIONS

Bound Name	GKS-2D Function Name
ginqsegattr	inquire segment attributes
ginqsegnames	inquire set of segment names in use
ginqsegnamesws	inquire set of segment names on workstation
ginqtextindices	inquire list of text indices
ginqtextrep	inquire text representation
ginqwsmaxnum	inquire workstation maximum number
gmessage	message
gredrawsegws	redraw all segments on workstation
grenameseg	rename segment
gsetdeferst	set deferral state
gsetfillrep	set fill area representation
gsethighlight	set highlighting
gsetlinerep	set polyline representation
gsetmarkerrep	set polymarker representation
gsetpatrep	set pattern representation
gsetsegpri	set segment priority
gsetsegtran	set segment transformation
gsettextrep	set text representation
gsetvis	set visibility

TABLE XLI

GKS-2D LEVEL 1b FUNCTIONS

Bound Name	GKS-2D Function Name
ginitpick	initialise pick
ginqdefpick	inquire default pick device data
ginqpickid	inquire pick identifier
ginqpickst	inquire pick device state
greqpick	request pick
gsetdet	set detectability
gsetpickid	set pick identifier
gsetpickmode	set pick mode

TABLE XLII

GKS-2D LEVEL 1c FUNCTIONS

Bound Name	GKS-2D Function Name
ggetpick	get pick
gsamplepick	sample pick

TABLE XLIII

GKS-2D LEVEL 2a FUNCTIONS

Bound Name	GKS-2D Function Name
gassocsegws	associate segment with workstation
gcopysegws	copy segment to workstation
ginsertseg	insert segment

APPENDIX G

GKS ERROR CODES

TABLE XLIV

GKS NON-ERROR CONDITION ERROR CODE

Symbolic Constant	Err No	Error Message
NO_ERROR	0	No error

TABLE XLV

GKS STATE ERROR CODES

Symbolic Constant	Err No	Error Message
ENOTGKCL	1	GKS not in proper state: GKS shall be in the state GKCL
ENOTGKOP	2	GKS not in proper state: GKS shall be in the state GKOP
ENOTWSAC	3	GKS not in proper state: GKS shall be in the state WSAC
ENOTSGOP	4	GKS not in proper state: GKS shall be in the state SGOP
ENOTACOP	5	GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
ENOTOPAC	6	GKS not in proper state: GKS shall be either in the state WSOP or in the state WSAC
ENOTWSOP	7	GKS not in proper state: GKS shall be in one of the states WSOP, WSAC or SGOP
ENOTGWWS	8	GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

TABLE XLVI

GKS WORKSTATION ERROR CODES

Symbolic Constant	Err No	Error Message
EWSIDINV	20	Specified workstation identifier is invalid
ECNIDINV	21	Specified connection identifier is invalid
EWSTPINV	22	Specified workstation type is invalid
ENOWSTYP	23	Specified workstation type does not exist
EWSISOPN	24	Specified workstation is open
EWSNOTOP	25	Specified workstation is not open
EWSCNTOP	26	Specified workstation cannot be opened
EWISSNOP	27	Workstation Independent Segment Storage is not open
EWISSOPN	28	Workstation Independent Segment Storage is already open
EWSISACT	29	Specified workstation is active
EWSNTACT	30	Specified workstation is not active
EWSCATMO	31	Specified workstatoin is of category MO
EWSNOTMO	32	Specified workstation is not of category MO
EWSCATMI	33	Specified workstation is of category MI
EWSNOTMI	34	Specified workstation is not of category MI
EWSCATIN	35	Specified workstation is of category INPUT
EWSIWISS	36	Specified workstation is Workstation Independent Segment Storage
EWSNOTOI	37	Specified workstatoin is not of category OUTIN
EWSNOTIO	38	Specified workstation is neither of category INPUT nor of category OUTIN
EWSNOTOO	39	Specified workstation is neither of category OUTPUT nor of category OUTIN

TABLE XLVI (continued)

GKS WORKSTATION ERROR CODES

Symbolic Constant	Err No	Error Message
EWSNOPXL	40	Specified workstation has no pixel store readback capability
EWSNOGDP	41	Specified workstation type is not able to generate the specified generalized drawing primitive
EWSMXOPN	42	Maximum number of simultaneously open workstations would be exceeded
EWSMXACT	43	Maximum number of simultaneously active workstation would be exceeded

TABLE XLVII

GKS TRANSFORMATION ERROR CODES

Symbolic Constant	Err No	Error Message
EBADXFRM	50	Transformation number is invalid
EBADRCTD	51	Rectangle definition is invalid
EBDVIEWP	52	Viewport is not within the Normalized Device Coordinate unit square
EBDWINDW	53	Workstation window is not within the Normalized Device Coordinate unit square
EVIEWDSP	54	Workstation viewport is not within the display space

TABLE XLVIII

GKS OUTPUT ATTRIBUTES ERROR CODES

Symbolic Constant	Err No	Error Message
EBADLINX	60	Polyline index is invalid
ENOLINEX	61	A representation for the specified polyline index has not been define on this workstation
ENOPLINX	62	A representation for the specified polyline index has not been predefined on this workstation
ELINEEQZ	63	Linetype is equal to zero
ENOLINTP	64	Specified linetype is not supported on this workstation
ELNWDLTZ	65	Linewidth scale factor is less than zero
EBADMRKX	66	Polymarker index is invalid
ENOMARKX	67	A representation for the specified polymarker index has not been defined on this workstation
ENOPMRKX	68	A representation for the specified polymarker index has not been predefined on this workstation
EMARKEQZ	69	Marker type is equal to zero
ENOMRKTP	70	Specified marker type is not supported on this workstation
EMKSZLTZ	71	Marker size scale factor is less than zero
EBADTXTX	72	Text index is invalid
ENOTEXTX	73	A representation for the specified text index has not been defined on this workstation
ENOPTXTX	74	A representation for the specified text index has not been predefined on this workstation
ETXTFEQZ	75	Text font is equal to zero

TABLE XLVIII (continued)

GKS OUTPUT ATTRIBUTES ERROR CODES

Symbolic Constant	Err No	Error Message
TNOTXTFP	76	Requested text font is not supported for the specified precision on this workstation
ECEXFLEZ	77	Character expansion factor is less than or equal to zero
ECHHTLEZ	78	Character height is less than or equal to zero
ECHRUPVZ	79	Length of character up vector is zero
EBADFILX	80	Fill area index is invalid
ENOFILLX	81	A representation for the specified fill area index has not been defined on this workstation
ENOPFILX	82	A representation for the specified fill area index has not been predefined on this workstation
ENOFSTYL	83	Specified Fill area interior style is not supported on this workstation
ESTYLEQZ	84	Style (pattern or hatch) index is equal to zero
EBADPATN	85	Specified pattern index is invalid
ENOHATCH	86	Specified hatch sytle is not supported on this workstation
EPATSZLZ	87	Pattern size value is not positive
ENOPATNX	88	A representation for the specified pattern index has not been defined on this workstation
ENOPPTNZ	89	A representation for the specified pattern index has not been predefined on this workstation
ENOPSTYL	90	Interior style PATTERN is not supported on this workstation

TABLE XLVIII (continued)

GKS OUTPUT ATTRIBUTES ERROR CODES

Symbolic Constant	Err No	Error Message
ECADIMEN	91	Dimensions of colour array are invalid
ECINDXLZ	92	Colour index is less than zero
EBADCOLX	93	Colour index is invalid
ENOCOLRX	94	A representation for the specified colour index has not been defined on this workstation
ENOPCLRX	95	A representation for the specified colour index has not been predefined on this workstation
ECOLRNGE	96	Colour is outside range [0,1]
EBADPICK	97	Pick indentifier is invalid

TABLE XLIX

GKS OUTPUT PRIMITIVES ERROR CODES

Symbolic Constant	Err No	Error Message
ENPOINTS	100	Number of points is invalid
ECHRCODE	101	Invalid code in string
EBDGDPID	102	Generalized drawing primitive identifier is invalid
EGDPDATA	103	Content of generalized drawing primitive data record is invalid
ECANTGDP	104	At least one active workstation is not able to generate the specified generalized drawing primitive
ECNTGDPC	105	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformations and clipping rectangle

TABLE L

GKS SEGMENT ERROR CODES

Symbolic Constant	Err No	Error Message
EBADNAME	120	Specified segment name is invalid
ENAMUSED	121	Specified sgement name is already in use
EWHATSEG	122	Specified segment does not exist
EWORKSEG	123	Specified segment does not exist on specified workstation
EWISSSEG	124	Specified segment does not exist on Workstation Independent Segment Storage
ESEGOPEN	125	Specified segment is open
ESEGPRIR	126	Segment priority is outside the range

TABLE LI

GKS INPUT ERROR CODES

Symbolic Constant	Err No	Error Message
ENOINDEV	140	Specified input device is not present on workstation
EREQUEST	141	Input device is not in REQUEST mode
ENSAMPLE	142	Input device is not in SAMPLE mode
ENOEVSMP	143	EVENT and SAMPLE input mode are not available at this level of GKS
ENOPETWS	144	Specified prompt and echo type is not supported on this workstation
EEBOUNDS	145	Echo area is outside display space
EBADDATA	146	Contents of input data record are invalid
EINQOVFL	147	Input queue has overflowed
ENOQOVRL	148	Input queue has not overflowed since GKS was opened or the last invocation of INQUIRE INPUT QUEUE OVERFLOW
EASWSCLO	149	Input queue has overflowed, but associated workstation has been closed
ENOCURIV	150	No input value of the correct class is in the current event report
EINVTOUT	151	Timeout is invalid
EBDINITV	152	Initial value is invalid
ESTROKSZ	153	Number of points in the initial stroke is greater than the buffer size
ESTRINSZ	154	length of the initial string is greater than the buffer size

TABLE LII

GKS METAFILE ERROR CODES

Symbolic Constant	Err No	Error Message
ERESERVE	160	Item type is not allowed for user items
EBDLNGTH	161	Item length is invalid
EMNOITEM	162	No item is left in GKS Metafile input
EMITMINV	163	Metafile item is invalid
ENOTGKSI	164	Item type is not a valid GKS item
EBADCNTS	165	Content of item data record is invalid for the specified item type
EEBDMXDR	166	Maximum item data record length is invalid
EINTERPT	167	User item cannot be interpreted
ENOFUNCT	168	Specified function is not supported in this level of GKS

TABLE LIII

GKS ESCAPE ERROR CODES

Symbolic Constant	Err No	Error Message
ENOESCFN	180	Specified escape function is not supported
ESCIDINV	181	Specified escape function identification is invalid
EESCDATA	182	Contents of escape data record are invalid

TABLE LIV

GKS MISCELLANEOUS ERROR CODES

Symbolic Constant	Err No	Error Message
EBDERRFL	200	Specified error file is invalid

TABLE LV

GKS SYSTEM ERROR CODES

Symbolic Constant	Err No	Error Message
EMEMSPAC	300	Storage overflow has occurred in GKS
ESEGSPAC	301	Storage overflow has occurred in segment storage
EIO_READ	302	Input/Output error has occurred while reading
EIOWRITE	303	Input/Output error has occurred while writing
EIOSENDD	304	Input/Output error has occurred while sending data to a workstation
EIORECDA	305	Input/Output error has occurred while receiving data from a workstation
EIOLIBMG	306	Input/Output error has occurred during program library management
EIORDWDT	307	Input/Output error has occurred while reading workstation description table
EMATHERR	308	Arithmetic error has occurred

TABLE LVI

GKS-3D TRANSFORMATION ERROR CODES

Symbolic Constant	Err No	Error Message
EVUVPCOL	400	View up vector and view plane normal are collinear
EVPNNULL	401	View plane normal is a null vector
EVUVNULL	402	View up vector is a null vector
EPVPLNPC	403	Projection viewport limits are not within NPC range
EPRPFBCP	404	Projection reference point is between the front and back clipping planes
EPRPVWPL	405	Projection reference point is on the view plane
EBADBOXD	406	Box definition is invalid
EBDVPNDC	407	Viewport is not within NDC unit cube
EBADVWIN	408	Specified view index is invalid
ENOVWXDE	409	A representation for the specified view index has not been defined on this workstation
ENOVWXPR	410	A representation for the specified view index has not been predefined on this workstation
EBDWWNDC	411	Workstation window limits are not within the NPC unit cube
EBCPFFCP	412	Back clipping plane is in front of the front clipping plane

TABLE LVII

GKS-3D OUTPUT ATTRIBUTES ERROR CODES

Symbolic Constant	Err No	Error Message
EBADEDGX	420	Edge index is invalid
ENOEDGXD	421	A representation for the specified edge index has not been defined on this workstation
ENOEDGXP	422	A representation for the specified edge index has not been predefined on this workstation
EEDGTEQZ	423	Edgetype is equal to zero
ENOEDGTP	424	Specified edgetype is not supported on this workstation
EEDGSLTZ	425	Edgewidth scale factor is less than zero
EPTRVCOL	426	Pattern reference vectors are collinear
ENOHLMOD	427	Specified HLHSR mode not supported on workstation
EBADHLID	428	Specified HLHSR identifier is invalid
EBADHLMO	429	Specified HLHSR mode is invalid

TABLE LVIII

GKS-3D OUTPUT PRIMITIVES ERROR CODES

Symbolic Constant	Err No	Error Message
ETXDVCOL	430	The text direction vectors are collinear
EBADLPTL	431	List of point lists is invalid
ENOAWGDP	432	At least one active workstation is not able to generate the specified generalized drawing primitive under the current transformation and clipping volume

TABLE LIX

GKS C LANGUAGE DEPENDENT ERROR CODES

Symbolic Constant	Err No	Error Message
EBUFSPAC	2200	Buffer overflow on input or inquiry functions
ERRLOGMT	2201	Error log is empty
EOUTRANG	2202	Out of range

TABLE LX

GKS RESERVED ERROR CODES

Symbolic Constant	Err No	Error Message

Error codes 2,203-2,299 are reserved for use of C language binding dependent errors.

Unused error numbers less than 2,000 are reserved for future standardization.

Error numbers 2,300-3,999 are reserved for language bindings.

Error numbers greater than or equal to 4,000 are reserved for registration.

NOTE - Error numbers are registered in the ISO International Register of Graphical Items, which is maintained by the Registration Authority (see 4.1.2). When an error has been approved by ISO, the error number will be assigned by the Registration Authority.

APPENDIX H

LINE AND MARKER SYMBOLIC CONSTANTS

TABLE LXI

LINE AND MARKER SYMBOLIC CONSTANTS

Symbolic Constant	Value
Line Types:	
GLN_SOLID	1
GLN_DASH	2
GLN_DOT	3
GLN_DOTDASH	4
Marker Types:	
GMK_POINT	1
GMK_PLUS	2
GMK_STAR	3
GMK_O	4
GMK_X	5
Miscellaneous Macros:	
GDEFAULT_MEM_SIZE	((Glong)(-1))