

INTERNATIONAL STANDARD

ISO/IEC
8806-4

First edition
1991-12-15

Information technology — Computer graphics — Graphical Kernel System for Three Dimensions (GKS-3D) language bindings —

Part 4:
C
(standards.iteh.ai)

<https://standards.iteh.ai/catalog/standards/sist/6c2e2e3-e62a-400c-bc16-7ca56151953c/iso-iec-8806-4-1991>
*Technologies de l'information — Infographie — Interfaces de langages
pour GKS (Graphical Kernel System) pour trois dimensions —
Partie 4: C*

INTERNATIONAL

ISO/IEC



Reference number
ISO/IEC 8806-4:1991(E)

Contents

1	Scope.....	1
2	Normative references.....	2
3	The C Language Binding of GKS-3D	3
3.1	Conformance	3
3.2	Functions versus Macros	3
3.3	Character Strings	3
3.4	Function Identifiers	3
3.5	Registration.....	3
3.6	Identifiers for Graphical Items.....	4
3.7	Return Values	4
3.8	Header Files.....	4
3.9	Memory Management.....	4
3.9.1	Functions which Return Simple Lists.....	5
3.9.2	Functions which Return Complex Data Structures	5
3.10	Error Handling.....	6
3.10.1	Application Supplied Error Handlers	6
3.10.2	Error Codes.....	7
3.10.3	C-specific GKS-3D errors	7
3.11	Colour Representations.....	7
3.12	Storage of Multi-dimensional Arrays	8
3.12.1	Storage of 2*3 Matrices.....	8
3.12.2	Storage of 3*4 Matrices.....	8
3.12.3	Storage of 4*4 Matrices.....	8
3.12.4	Storage of Colour Arrays.....	8
4	Tables.....	9
4.1	Abbreviation Policy in Construction of Identifiers.....	9
4.2	Table of Abbreviations Used.....	9
4.3	Function Names	13
4.3.1	List Ordered Alphabetically by Bound Name.....	13
4.3.2	List Ordered Alphabetically by GKS-3D Name.....	19
4.3.3	List Ordered Alphabetically by Bound Name within Level.....	25
5	Type Definitions	32
5.1	Mapping of GKS-3D data types	32
5.2	Environmental Type Definitions.....	32
5.3	Implementation Dependent Type Definitions	33
5.4	Implementation Independent Type Definitions	44
6	Macro Definitions	65
6.1	Function identifiers	65
6.2	Error Codes.....	68
6.3	Miscellaneous Macros	75
6.3.1	Linetypes.....	75
6.3.2	Marker Types.....	75
6.3.3	Colour Models	75
6.3.4	Prompt and Echo Types.....	75
6.3.5	Default Parameters of OPEN GKS.....	76
7	C GKS-3D Function Interface	77
7.1	Notational Conventions	77
7.2	Control Functions	77
7.3	Output Functions	79

© ISO/IEC 1991

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

7.4	Output Attribute Functions	81
7.4.1	Workstation Independent Primitive Attributes	81
7.4.2	Workstation Attributes	85
7.5	Transformation Functions	87
7.5.1	Normalization Transformation	87
7.5.2	View transformation functions	88
7.5.3	HLHSR functions	88
7.5.4	Workstation transformation	89
7.6	Segment Functions	89
7.6.1	Segment Manipulation Functions	89
7.6.2	Segment Attribute Functions	91
7.7	Input Functions	92
7.7.1	Initialization of Input Devices Functions	92
7.7.2	Setting the Mode of Input Devices Functions	95
7.7.3	Request Input Functions	96
7.7.4	Sample Input Functions	98
7.7.5	Event Input Functions	100
7.8	Metafile Functions	102
7.9	Inquiry Functions	102
7.9.1	Inquiry Functions for Operating State Value	102
7.9.2	Inquiry Functions for GKS-3D Description Table	103
7.9.3	Inquiry Functions for GKS-3D State List	103
7.9.4	Inquiry Functions for Workstation State List	112
7.9.5	Inquiry Functions for Workstation Description Table	124
7.9.6	Inquire functions for the Segment State List	135
7.9.7	Pixel Inquiries	136
7.9.8	Inquiry Functions for Error State List	137
7.10	Utility Functions	137
7.10.1	Utility Functions in GKS-3D	137
7.10.2	Binding Specific Utilities	138
7.11	Error Handling	139
A	Compiled GKS-3D/C Specification	141
B	Sample Programs	204
B.1	STAR Program	204
B.2	IRON Program	206
B.3	MAP Program	213
B.4	MANIPULATE Program	215
B.5	SHOW LINE Program	221
B.6	DODECA Program	226
C	Metafile Items	231
D	Short Function Identifiers	233
E	Memory Management	240
E.1	Introduction	240
E.2	Functions That Return Simple Lists	240
E.2.1	Operation of <code>ginq_list_line_inds</code>	240
E.3	Functions That Return Structured Data	243
E.3.1	Operation of <code>gcreate_store</code>	244
E.3.2	Operation of <code>ginq_stroke_st</code> and <code>ginq_pat_rep</code>	246
E.3.3	Operation of <code>gdel_store</code>	250
F	Function Lists	253
F.1	Alphabetic by GKS-3D Name	253
F.2	Alphabetic by Binding Name	259

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) together form a system for worldwide standardization as a whole. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with the fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 8806-4 was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology*.

ISO/IEC 8806 consists of the following parts, under the general title

Information technology - Computer graphics - Graphical Kernel System for Three Dimensions (GKS-3D) language bindings -

- *Part 1: FORTRAN*
- *Part 2: PASCAL*
- *Part 3: ADA*
- *Part 4: C*

Annexes A to F of this part of ISO/IEC 8806 are for information only.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 8806-4:1991

<https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991>

Introduction

The Graphical Kernel System for Three Dimensions (GKS-3D) functional description is registered as ISO 8805 : 1988. As explained in the Scope and Field of Application of ISO 8805, that International Standard is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this part of ISO/IEC 8806 is to define a standard binding for the C computer programming language.

Some functions in ISO 8805 also appear in the Graphical Kernel System (GKS) functional description (ISO 7942). The binding of those functions in this part of ISO/IEC 8806 is the same as in ISO/IEC 8651-4, the C language binding of GKS.

iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC 8806-4:1991](https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991)

<https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991>

iTeh STANDARD PREVIEW (standards.iteh.ai)

This page intentionally left blank

[ISO/IEC 8806-4:1991](#)

<https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991>

Information technology – Computer graphics – Graphical Kernel System for Three Dimensions (GKS-3D) language bindings -

Part 4:

C

1 Scope

The Graphical Kernel System for Three Dimensions (GKS-3D), ISO 8805 : 1988, specifies a language independent nucleus of a graphics system. For integration into a programming language, GKS-3D is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 8806 specifies such a language dependent layer for the C language.

[ISO/IEC 8806-4:1991](https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991)

<https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991>

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provision of this part of ISO/IEC 8806. At the time of publication, the editions indicated were valid. All standards are subject to revisions, and parties to agreements based on this part of ISO/IEC 8806 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7942:1985, *Information processing systems – Computer graphics – Graphical Kernel System (GKS) functional description*.

ISO/IEC 8651-4:1991, *Information technology – Computer graphics – Graphical Kernel System (GKS) language bindings - Part 4 : C*.

ISO 8805:1988, *Information processing systems – Computer graphics – Graphical Kernel System for Three Dimensions (GKS-3D) functional description*.

ISO/IEC 8806-1:1991, *Information processing systems – Computer graphics – Graphical Kernel System for three dimensions (GKS-3D) - language bindings - Part 1 : FORTRAN*.

ISO/IEC 9899:1990, *Programming languages - C*.

ISO/IEC TR 9973:1988, *Information processing – Procedures for registration of graphical items*.

ITeH STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 8806-4:1991](https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991)

<https://standards.iteh.ai/catalog/standards/sist/1bede2e3-e62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991>

3 The C Language Binding of GKS-3D

The C language binding of GKS-3D shall be as described in clauses 3, 4, 5 and 6.

3.1 Conformance

This part of ISO/IEC 8806 incorporates the rules of conformance defined in the GKS-3D Standard (ISO 8805) for GKS-3D implementations, with those additional requirements specifically defined for C bindings in GKS-3D.

The following criteria shall determine conformance of an implementation to this part of ISO/IEC 8806:

In order to conform, an implementation of the C binding of GKS-3D shall implement a specific level of GKS-3D as specified in ISO 8805. It shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 8806 for that same level of GKS-3D and all lower levels and for a specific level of C.

Thus, for example, the syntax of the function names shall be precisely as specified in the binding and parameters shall be of the data types stated in the binding.

3.2 Functions versus Macros

An implementation may substitute macros for functions. However, the macros shall be designed so that side-effects work properly. In general, a macro cannot be used to replace the error handling function `gerr_hand`. See also 3.10.

3.3 Character Strings

The C language represents character strings as an array of characters terminated by the null character (i.e. `'\0'`). This means that the null character is not usable as a printable character.

3.4 Function Identifiers

The function names of GKS-3D are all mapped to C functions which begin with the letter `g`. Words and phrases used in the GKS-3D function names are often abbreviated in the representation and are always separated with the underscore character `"_"`. The set of such abbreviations is given in 4.2, and the resulting C function names are listed in 4.3. For example, the abbreviation for the GKS-3D function DELETE SEGMENT FROM WORKSTATION is `gdel_seg_ws`. `del`, `seg`, and `ws` are abbreviations for DELETE, SEGMENT, and WORKSTATION. The conjunctive FROM is mapped to the null string.

The C standard (ISO/IEC 9899) requires that compilers recognize internal identifiers which are distinct in at least 31 characters. That standard also requires that external identifiers (i.e. those seen by the linker) be recognized to a minimum of six characters, independent of case.

Implementations which run in environments where two distinct C internal identifiers would be equivalent, if they were both external identifiers, shall include a set of `#defines` in the header file which equate the long names to a set of short names. A possible set of short names for a compiler that accepts only eight characters for external definitions may be found in annex D.

3.5 Registration

ISO 8805 reserves certain value ranges for registration¹ as graphical items. The registered graphical items will be bound to the C programming language (and other programming languages). The registered item

¹) For the purpose of this part of ISO/IEC 8806 and according to the rules for the designation and operation of registration authorities in the ISO/IEC Directives, the ISO and IEC councils have designated the National Institute of Standards and Technology (Institute of Computer Sciences and Technology), A-266 Technology Building, Gaithersburg, MD 20899, USA to act as registration authority.

Registration**The C Language Binding of GKS-3D**

binding will be consistent with the binding presented in this part of ISO/IEC 8806.

3.6 Identifiers for Graphical Items

Generalized Drawing Primitives and Escape functions are referenced via identifiers. This part of ISO/IEC 8806 specifies the format of the identifiers but it does not specify the registration of the identifiers. The identifiers are used as arguments to the functions `ggdp`, `ggdp3`, and `gescape`.

An implementation may also represent GDPs and Escapes as separate functions, but this is not required.

There are two formats for these identifiers. One format is for registered GDPs and Escapes and the other format is for unregistered GDPs and Escapes.

The format for registered GDP identifiers is:

```
#define GGDP_Rn (n) /* 'n' is the registered GDP id. */
#define GGDP3_Rn (n) /* 'n' is the registered GDP3 id. */
```

The format for unregistered GDP identifiers is:

```
#define GGDP_Un (-n) /* 'n' is implementation dependent */
#define GGDP3_Un (-n) /* 'n' is the registered GDP3 id. */
```

The format for registered Escape function identifiers is:

```
#define GESCAPE_Rn (n) /* 'n' is the registered Escape id. */
```

The format for unregistered Escape function identifiers is:

```
#define GESCAPE_Un (-n) /* 'n' is implementation dependent */
```

3.7 Return Values

All GKS-3D/C functions return `void`.

3.8 Header Files

C provides a mechanism to allow external files to be included in a compilation. Clause 5 of this part of ISO/IEC 8806 describes the data types that shall be defined in the file `gks.h` which should be included in any application program that intends to use GKS-3D via the C binding.

This part of ISO/IEC 8806 uses the data type `size_t` (as a field in the data type `Gdata`). The type `size_t` is environment-dependent (e.g. `int`, `long int`, `unsigned int`) and is defined in the file `<stddef.h>`. Therefore the file `gks.h` shall also include the file `<stddef.h>`.

Additional implementation-dependent items may be placed in this file if needed. These items should start with the sentinel "G" or "g", as far as applicable.

The file `gks.h` shall also contain external declarations for all GKS-3D/C functions because they return `void`. For example, the declaration for the function `gopen_gks` would look like this:

```
extern void gopen_gks(char *err_file, size_t mem_units);
```

3.9 Memory Management

The application shall allocate the memory needed for the data returned by the implementation. In general, the application will allocate a C structure and pass a pointer to that structure to an inquiry routine, which will then place information into the structure. However, a number of inquiry functions return variable length data, the length of which is not known *a priori* by the application.

These functions fall into two classes. One class of functions returns a simple, homogeneous, list of items. For example, the function INQUIRE SET OF SEGMENT NAMES returns a list of the segment names in use. The other class returns complex, heterogeneous data structures. For example, the function INQUIRE LOCATOR DEVICE STATE returns the device state which includes a locator data record; the data record can contain arbitrarily complex implementation-defined data structures. The binding of these two classes of functions is described in detail below. Subclause 3.10 describes the errors that can be invoked during execution of functions which use the memory management policy.

The C Language Binding of GKS-3D

Memory Management

3.9.1 Functions which Return Simple Lists

Inquiry functions which return a list of items are bound such that the application can inquire about a portion of the list. This list is a subset of the implementation's internal list and is called the application's list. This allows the application to process the implementation's list in a piecewise manner rather than all at once.

The application allocates the memory for a list and passes that list to the implementation. The implementation places the results of the inquiry into the list. In order to support this policy of memory management, three additional parameters have been added to functions which return lists:

- a) `num_elems_appl_list`: An integer input parameter which is the length of the application's list. The value of `num_elems_appl_list` indicates the number of items (i.e. list elements) which will fit into the application list. A value of 0 is valid and allows the application to determine the size of the implementation's list (which is returned via `num_elems_impl_list`) without having the implementation return any of the elements of its list. If `num_elems_appl_list` is negative, `GE_APPL_LIST_LENGTH_LT_ZERO` is returned as the value of the error indicator parameter.
- b) `start_ind`: An integer input parameter which is an index into the implementation's list. (Index 0 is the first element of both the implementation's and application's list.) `start_ind` indicates the first item in the implementation's list that is copied into index 0 of the application's list. Items are copied sequentially from the implementation's list into the application's list until the application's list is full or there are no more items in the implementation's list. If `start_ind` is out of range, error `GE_START_IND_INVALID` is returned as the value of the error indicator parameter.
- c) `num_elems_impl_list`: An output parameter which is a pointer to an integer. The implementation stores into this parameter the number of items that are in the implementation's list.

In annex E, a possible underlying mechanism is described.

[ISO/IEC 8806-4:1991](https://standards.iso.org/standards/catalog/standards/sist/1bede2e3-c62a-4bc8-ac1b-7ca56151933c/iso-iec-8806-4-1991)

3.9.2 Functions which Return Complex Data Structures

The data returned by the `ESCAPE` function and the functions which return input device data records or pattern tables can be complex in structure. They cannot be represented by a simple list of items. It would be an onerous task for the application to have to allocate and prepare data structures for these routines. In order to facilitate this task of using these inquiry functions, the binding defines a new resource, called a *Store*, to manage the memory for these functions.

The *Store* resource is opaque to the application. The application does not know the structure of the *Store* or how it is implemented. The *Store* is defined as a `void *`. This part of ISO/IEC 8806 defines two new functions which create (in `CREATE STORE`, bound as `gcreate_store`) and delete (in `DELETE STORE`, bound as `del_store`) a *Store*.

A *Store* is used by the implementation to manage the memory needed by the functions which return complex data structures. Without specifying an implementation of a *Store*, it is safe to say that it will contain and control memory needed to hold the data returned by these functions and also contain some bookkeeping information about the contents and size of the memory.

The semantics of the *Store* resource provide two levels of memory management. The implementation is responsible for managing the memory at a low level because it uses, reuses, allocates and deallocates memory from the system in order to return information to the application. But the application is ultimately responsible for managing the memory at a high level because it creates and deletes *Stores*.

A *Store* is passed as a parameter to a function returning complex data structures. Another parameter to this function is a pointer to a pointer to a structure which defines the format of the returned data. The *Store* contains memory for the structure and any additional memory referenced by fields within the structure. The application accesses the returned data through its pointer to the structure. It does not use the *Store* to access the data.

A *Store* continues to hold the information from the function until the *Store* is deleted by the `DELETE STORE` function or until the *Store* is used as an argument to a subsequent function, which returns complex

Memory Management

The C Language Binding of GKS-3D

data structures. At that time, the old information is replaced with the new. Thus multiple calls to functions overwrite the contents of a *Store*. A *Store* only contains the results of the last function.

This part of ISO/IEC 8806 defines two new errors that can occur when using or creating a *Store*; these errors are described in 3.10.3. For most functions using a *Store*, these and other errors are returned via the "error indicator" parameter. However, the function ESCAPE does not have an error indicator parameter. For this function, the error reporting mechanism is used when an error is encountered. For this function, the implementation shall, in addition to reporting the error, set the pointer to the returned data to NULL when an error occurs. See the binding of these functions for more information.

The definitions for the functions CREATE STORE and DELETE STORE follow:

CREATE STORE		GKOP, WSOP, WSAC, SGOP	L0a
Parameters:			
Out	error indicator	I	
Out	store	STORE	

Effect: Creates a *Store* and returns a handle to it in the output parameter *store*. If the *Store* cannot be created, the *store* parameter is set to NULL and the error indicator is set to one of the following error values:

- 8 GKS-3D not in proper state; GKS-3D shall be in one of the states GKOP, WSOP, WSAC or SGOP
- 2203 Error while allocating Store

Errors: None.

DELETE STORE		GKOP, WSOP, WSAC, SGOP	L0a
Parameters:			
Out	error indicator	I	
In/Out	store	STORE	

Effect: Deletes the *Store* and all internal resources associated with it. If there is not an error, the parameter *store* will be set to NULL to signify that it is no longer valid. If an error is detected, the error indicator is set to one of the following values:

- 8 GKS-3D not in proper state; GKS-3D shall be in one of the states GKOP, WSOP, WSAC or SGOP

Errors: None.

In 7.10.2, the C specification of these functions is given. In annex E, a possible underlying mechanism is illustrated.

3.10 Error Handling

3.10.1 Application Supplied Error Handlers

User-defined error handlers shall accept the same arguments as the standard error handler. The user-defined error handler is specified by the utility function (see also 7.10.2)

The C Language Binding of GKS-3D

Error Handling

SET ERROR HANDLER

GKCL, GKOP, WSOP, WSAC, SGOP

L0a

Parameters:

In	New error handling function	Function
Out	Old error handling function	Function

Effect: Sets the GKS-3D error handling function to *New error handling function* and returns the current error handling function to *Old error handling function*.

Errors: None.

Application defined error handling functions accept the same arguments as the standard error handler. They may invoke the standard error logging function ERROR LOGGING.

ISO 8805 defines the initial error handling function to be ERROR HANDLING, that is, the value of the parameter *Old error handling function* points to ERROR HANDLING, when SET ERROR HANDLER is called for the first time.

When the application changes the error handling function, the implementation will invoke the new function when an error is detected. If the application calls the default error handling function ERROR HANDLING, ERROR HANDLING will always call the function ERROR LOGGING.

If *New error handler* is not a valid pointer, the error handling will automatically be done by the standard error handler ERROR HANDLING.

User-defined error handlers may invoke the standard error logging function ERROR LOGGING.

3.10.2 Error Codes

Hard coding numbers into a program decreases its maintainability. Therefore, this part of ISO/IEC 8806 defines a set of constants for the GKS-3D error numbers. Each error constant begins with the characters GE_. See also 6.2 for the error macros.

3.10.3 C-specific GKS-3D errors

This part of ISO/IEC 8806 knows some specific error messages. This section gives the messages and the error macros.

Value	Message
2200	Start index out of range Is issued when the start index is less than zero or larger than the last element in the implementation's list
2201	Length of application list is negative Is issued when the length of the application's list is less than zero
2202	Enumeration type out of range Is issued when a parameter value whose type is an enumeration is out range of the enumeration
2203	Error while allocating Store Is issued when an error is detected during CREATE STORE
2204	Error while allocating memory for Store Is issued when a function using a <i>Store</i> is unable to allocate memory for the <i>Store</i>

3.11 Colour Representations

GKS-3D defines 4 colour models (RGB, CIE L*u*v* 1976, HLS and HSV) of which RGB and CIE L*u*v* are mandatory. For each of these models, a colour specification is defined (Grgb, Gcieluv, Ghsv, Ghls). The colour representation is defined in GKS-3D/C as

Colour Representations

The C Language Binding of GKS-3D

```
typedef union {
    Grgb      rgb;          /* Red Green Blue colour specification */
    Gcieluv   cielv;        /* CIE L*u*v* 1976 colr. spec. */
    Ghls      hls;          /* Hue Luminance Saturation colr. spec. */
    Ghsv      hsv;          /* Hue Saturation Value colr. spec. */
    Gdata     impl_dep;     /* impl. dep. colr. spec. */
} Gcolr_rep;
```

which guarantees extensibility of GKS-3D/C for other colour models.

3.12 Storage of Multi-dimensional Arrays

3.12.1 Storage of 2*3 Matrices

The entries of `Gtran_matrix` data types shall be stored such that the segment transformation is defined by

```
Tp.x = mat[0,0]*p.x + mat[0,1]*p.y + mat[0,2];
Tp.y = mat[1,0]*p.x + mat[1,1]*p.y + mat[1,2];
```

where `p` is a 2D point, `Tp` its transformation and `mat` is of type `Gtran_matrix`.

3.12.2 Storage of 3*4 Matrices

The entries of `Gtran_matrix3` data types shall be stored such that the segment transformation is defined by

```
Tp.x = mat[0,0]*p.x + mat[0,1]*p.y + mat[0,2]*p.z + mat[0,3];
Tp.y = mat[1,0]*p.x + mat[1,1]*p.y + mat[1,2]*p.z + mat[1,3];
Tp.z = mat[2,0]*p.x + mat[2,1]*p.y + mat[2,2]*p.z + mat[2,3];
```

where `p` is a 3D point, `Tp` its transformation and `mat` is of type `Gtran_matrix3`.

3.12.3 Storage of 4*4 Matrices

The entries of `Gmatrix3` data types shall be stored such that the projection transformation is defined by

```
Tp.x = x' / w';
Tp.y = y' / w';
Tp.z = z' / w';
x' = mat[0,0]*p.x + mat[0,1]*p.y + mat[0,2]*p.z + mat[0,3];
y' = mat[1,0]*p.x + mat[1,1]*p.y + mat[1,2]*p.z + mat[1,3];
z' = mat[2,0]*p.x + mat[2,1]*p.y + mat[2,2]*p.z + mat[2,3];
w' = mat[3,0]*p.x + mat[3,1]*p.y + mat[3,2]*p.z + mat[3,3];
```

where `p` is a 3D point, `Tp` its projection transformation and `mat` is of type `Gmatrix3`.

3.12.4 Storage of Colour Arrays

The entries of `Gpat_rep` data types shall be stored such that the colour index at the (i,j)-th entry is given by

```
Colr_indi,j = colr_rect.colr_array[i + DX*j];
i = 0, ..., DX-1; j = 0, ..., DY-1;
DX = colr_rect.dims.size_x; DY = colr_rect.dims.size_y;
```

where `colr_rect` is of type `Gpat_rep`.

4 Tables

4.1 Abbreviation Policy in Construction of Identifiers

In the construction of the several data types, function names, etc., the following policy is applied:

- 1) All identifiers in the C binding are abbreviated using the same abbreviations for every component and using underscores to denote blanks;
- 2) The plural of an expression is constructed by adding an "s" after its abbreviation; so, for example, "vector" is abbreviated to "vec" and "vectors" is abbreviated to "vecs"; if an expression is mapped to NULL, so will be its plural;
- 3) Digits are also preceded by underscores, except for the 3, if the 3D extension is denoted;
- 4) The words POLYLINE, POLYMARKER and FILL AREA (SET) are not abbreviated in the output primitive function names; in all other cases they are abbreviated using the list in 4.2;
- 5) Construction of GKS-3D/C identifiers:
 - a) Function names:
"g" (lower case) followed by abbreviated function name in lower case;
 - b) Data types:
"G" (upper case) followed by abbreviated data type in lower case;
 - c) Fields of data types; the following refinements are used: "redundant" (words in the field name that are identical to those in the structure name) parts are omitted, if the context allows this; thus the colour index in the field of `Gxxx_bundle` is abbreviated to `colr_ind`, because the context makes clear which colour index is used;
 - d) Function macros:
"Gfn_" followed by abbreviation of function name;
 - e) GKSM item types:
"Gksm_" followed by abbreviation of item name;
 - f) Error macros:
"GE_" followed by some abbreviated expression;
 - g) Fields of enumeration types:
"G" (upper case) followed by a prefix followed by an abbreviation of the field name; this prefix is constant for each enumeration field; all the fields are in upper case;

4.2 Table of Abbreviations Used

In this table, only words which are abbreviated are listed. They are used for

function names;
data types;
fields of data types;
error macros;
GKSM item types.

The word "NULL" denotes those words which are deleted completely when forming function names or data types.

Word or Phrase	Abbreviation
accepted	NULL