INTERNATIONAL STANDARD

ISO/IEC 8651-4

Second edition
1995-06-01

Information technology — Computer graphics — Graphical Kernel System (GKS) language bindings —

Part 4: C

Technologies de l'information — Infographie — Interfaces langage avec GKS —

Partie 4: C

Reference number
ISO/IEC 8651-4:1995(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. 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 particular 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 JTC 1. 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 8651-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

This second edition cancels and replaces the first edition (ISO/IEC 8651-4:1991), which has been technically revised.

ISO/IEC 8651-4 consists of the following parts, under the general title *Information technology — Computer graphics — Graphical Kernel System (GKS) language bindings*:

— *Part 1: FORTRAN*

— *Part 2: Pascal*

— *Part 3: Ada*

— *Part 4: C*

Annexes A to F of this part of ISO/IEC 8651 are for information only.

Introduction

The Graphical Kernel System (GKS) functional description is registered as ISO/IEC 7942-1:1994. As explained in the Scope and Field of Application of ISO/IEC 7942-1, 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 8651 is to define a standard binding for the C computer programming language.

Information technology — Computer graphics — Graphical Kernel System (GKS) language bindings —

Part 4: C

1 Scope

The Graphical Kernel System (GKS), ISO/IEC 7942–1:1994 , specifies a language independent nucleus of a graphics system. For integration into a programming language, GKS is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 8651 specifies such a language dependent layer for the C language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 8651. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 8651 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/IEC 7942-1:1994, *Information technology – Computer graphics and image processing – Graphical Kernel System (GKS) – Part 1 : Functional description.*
ISO/IEC 9899:1990, *Programming languages – C.*
ISO/IEC TR 9973:1994, *Information technology – Computer graphics and image processing – Procedures for registration of graphical Items.*

3 The C language binding

The C language binding of GKS shall be as described in clauses 3 to 7.

3.1 Classification and designation

This part of ISO/IEC 8651 incorporates the rules of conformance defined in the GKS Standard (ISO/IEC 7942-1) for GKS implementations, with those additional requirements specifically defined for C bindings in GKS.

The following criteria shall determine conformance of an implementation to this part of ISO/IEC 8651:

> In order to conform, an implementation of the C binding of GKS shall implement GKS as specified in ISO/IEC 7942–1. It shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 8651 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 are all mapped to C functions which begin with the letter **g**. Words and phrases used in the GKS 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 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 language (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 object-like macros in the header which equate the long names to a set of short names. A possible set of short names for a compiler that accepts only 8 characters for external definitions may be found in annex C.

3.5 Registration

ISO/IEC 7942 reserves certain value ranges for registration[1] 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 presented in this part of ISO/IEC 8651.

[1] For the purpose of this part of ISO/IEC 8651 and according to the rules for the designation and operation of registration authorities in the ISO/IEC Directives, the ISO/IEC council has 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.

3.6 Identifiers for graphical items

Generalized Drawing Primitives and Escape functions are referenced via identifiers. This part of ISO/IEC 8651 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** 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)          /* where 'n' is the registered GDP
                                      identifier */
```

The format for unregistered GDP identifiers is:

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

The format for registered Escape function identifiers is:

```
#define     GESCAPE_Rn   (n)          /* where 'n' is the registered
                                      Escape identifier */
```

The format for unregistered Escape function identifiers is:

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

3.7 Return values

All GKS/C functions have return value **void**.

3.8 Headers

3.8.1 gks.h

C provides a mechanism to access information stored in a header via the **#include** preprocessing directive. Clause 5 of this part of ISO/IEC 8651 describes the data types that shall be defined in the header **gks.h** which shall be included in any application program that intends to use GKS via the C binding.

This part of ISO/IEC 8651 uses the data type **size_t** (inter alia as a field in the data type **Gdata**). The type **size_t** is environment-defined (i.e. **unsigned long** or **unsigned int**) and is defined in the headers **<stdio.h>, <stddef.h>, <stdlib.h>, <string.h>, <time.h>**.

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

The header **gks.h** shall also contain external declarations for all GKS/C functions because they have a **void** return type. For example, the declaration for the function **gopen_gks** would look like this:

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

3.8.2 gks_compat.h

For application programs which used to run on top of the 1991 edition of this part of ISO/IEC, the header **gks_compat.h** is provided. **gks_compat.h** includes GKS/C data types that are no longer supported, as well as external declarations for all GKS/C functions that are no longer supported. Implementations of this part of ISO/IEC 8651 shall support these functions in a compatibility layer, according to the guidelines in Annex G of ISO/IEC 7942–1:1994.

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 LIST OF MARKER INDICES returns a list of the available marker indices. The other class returns complex, heterogeneous data structures. For example, the function INQUIRE GKS STATE LIST ENTRY returns a piece of the GKS state which may include several data structures of different length. 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.

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_INVAL`** 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 D, a possible underlying mechanism is described.

3.9.2 Functions which return complex data structures

The data returned by inter alia the ESCAPE function, the AWAIT INPUT function and the functions which return state lists or description 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 8651 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 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 8651 defines two errors that can occur when using or creating a *Store*; these errors are described in 6.2. 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

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 error indicator is set to one of the following error values:

GE_GKS_NOT_OPEN	GKS not in proper state; GKS shall be in the state (GKOP, *)
GE_ERR_ALLOC_STORE	Error while allocating Store

Errors: None.

DELETE STORE

Parameters:

Out	error indicator	I
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 indiocator is set to one of the following values:

GE_GKS_NOT_OPEN	GKS not in proper state; GKS shall be in the state (GKOP, *)

Errors: None.

In 7.2.13.2, the C specification of these functions is given. In annex D, 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 7.2.13.2)

SET ERROR HANDLER

Parameters:

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

Effect: Sets the GKS 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/IEC 7942 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 8651 defines a set of constants for the GKS error numbers. Each error constant begins with the characters **GE_**. See also 6.2 for the error macros.

3.10.3 C-specific GKS errors

This part of ISO/IEC 8651 defines some additional error messages. In 6.2 the numbers and their macros are given.

3.11 Colour representations and specifications

GKS 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 and specification are defined in 5.4 by the types **Gcolr_rep** and **Gcolr_specif.**

3.12 Colour characteristics

GKS defines the colour characteristics as a basic type. The colour characteristics is defined in this part of ISO/IEC 8651 by the data type **Gcolr_chars**, which is documented in 5.3.

3.13 Storage of multi-dimensional arrays

3.13.1 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**.

3.13.2 Storage of conics in 3*3 matrices

The entries of **Gconic_matrix** data types shall be stored such that the conic is defined by

$$a[0,0]x^2 + (a[0,1] + a[1,0])\ xy + a[1,1]y^2$$
$$+ (a[0,2] + a[2,0])x + (a[1,2] + a[2,1])y + a[2,2] = 0$$

where **a** is of type **Gconic_matrix**.

3.13.3 Storage of colour arrays

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

$xxx_{i,j}$ = `colr_rect.dir_colr.xxx[i + DX*j]`; i = 0, . . . , DX - 1; j
$colr\_ind_{i,j}$ = `colr_rect.colr_array[i + DX*j]`;
DX = `colr_rect.dims.size_x`;
DY = `colr_rect.dims.size_y`;

where xxx = (rgb|cieluv|hls|hsv) is of type **Gxxx** and **colr_rect** is of type **Gpat_rep**.

3.14 Compatibility with the 1991 edition

In the second (1994) edition of ISO/IEC 7942 some functions and data types of the 1985 edition have been overtaken or deprecated. They are documented in the informative annex G. Examples are the GKSM functions, which have been overtaken by AUDIT functions.

In this edition of ISO/IEC 8651–4 *all* functions and *all* data types of of the 1991 edition (the C binding of the 1985 edition of ISO/IEC 7942) have been preserved. The support of overtaken or deprecated functions or data types falls into two categories:

1) *Functions which have been renamed (e.g. INQUIRE TEXT EXTENT → GET TEXT EXTENT) or which have been incorporated into more compact functions (e.g. the OUTPUT functions).*
   These functions and their related data types are documented in clauses 5 and 7. For example, the function CREATE OUTPUT PRIMITIVE has been bound to the C function **gcreate_out_prim**. Besides that, however, the output functions **gpolyline, gpolyline_set, gnurb_set, gconic_sec_set, gpolymarker, gfill_area, gfill_area_set, gell_sec_set, gell_seg_set, gell_disc_set, gclosed_nurb_set, gtext, gcell_array, gdesign, ggdp** have been defined.

2) *Functions which have been deprecated (INQUIRE GKS LEVEL) or which have been overtaken by other functions (GKSM functions, for example).*
   These functions are documented in annex E. They will be deleted at the next edition of this part of ISO/IEC 8651.

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.

4) The word REALIZED is not abbreviated in the second field of the enumeration data type **`Ginq_type`**; in all other cases it is abbreviated using the list in 4.2.

5) Construction of GKS/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 linewidth in the field of **`Gline_bundle`** is abbreviated to **`width`**, because the context makes clear which width is used;

   d) Function macros:
   "**`Gfn_`**" followed by abbreviation of function name;

   e) Error macros:
   "**`GE_`**" followed by some abbreviated expression;

   f) 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.

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

Word or Phrase	**Abbreviation**
CIE L*u*v* (colour model)	cieluv
accumulate	accum
actual	act
addition	add
alignment	align
allocate	alloc
and	NULL

application	appl
arithmetic	arith
associate(d)	assoc
at	NULL
attribute	attr
attribute source flag	asf
availability	avail
available	avail
begin	beg
between	NULL
boundaries	bndries
boundary	bndry
buffer	buf
cannot	cant
centre	ctr
character	char
characteristics	chars
circular	circ
classification	class
clipping	clip
colour	colr
component	comp
composite	comp
composition	comp
concatenation	concat
connection	conn
contour	cont
control	ctrl
conversion	conv
coordinate	coord
correspondence	corr
criterion	crit
current	cur
dashed	dash
default	def
defined	def
delete	del
deletion	del
dependent	dep
description table	dt
designation	design
detectability	det
detectable	det
device	dev
device coordinate(s)	dc
digital	digit
dimension	dim
direction	dir
display	disp
dotted	dot
duplicate	dup
elliptic	ell

equal	eq
error	err
evaluate	eval
expansion	expan
facility	fac
factor	NULL
fill area	fill
from	NULL
function	func
generalized drawing primitive	gdp
generic	gen
gks closed	gkcl
gks open	gkop
graphical	graph
handling	hand
height	ht
highlighted	highl
highlighting	highl
homogeneous	homo
horizontal	hor
hue lightness saturation (colour model)	hls
hue saturation value (colour model)	hsv
hyperbola	hyp
identifier	id
implemantation	impl
implicit	impl
in	NULL
in use	NULL
index	ind
indicator	ind
individual	indiv
initial	init
initialize	init
input	in
input/output	io
inquire	inq
instance	inst
integer	int
interior	int
invalid	inval
invisible	invis
length	NULL
less than	lt
less than or equal	le
library	lib
locator	loc
logical	NULL
luminance	lum
mapped	map
mapping	map
maximum	max
measure	meas

memory	mem
metafile	mf
minimum	min
monochrome	monochr
nominal	nom
normal	norm
normalization	norm
normalized device coordinate(s)	ndc
number	num
of	NULL
on	NULL
operating	op
operation	op
orientation	ori
output	out
overflow	overf
parallelogram	paral
parameter	param
pattern	pat
picture	pic
picture part closed	ppcl
picture part open	ppop
pointer	ptr
polynomial	pol
position	pos
precision	prec
predefined	pred
primary	prim
primitive	prim
priority	pri
process	proc
prompt	pr
prompt and echo type	pet
queue	NULL
rational	rat
realized	real
record	rec
rectangle	rect
red green blue (colour model)	rgb
reference	ref
registered	reg
reject	rej
relative	rel
replication	repl
representation	rep
request(ed)	req
retrieve	ret
rubber	rub
saturation	satur
scale factor	NULL
section	sec
sector	sec

segment	seg
segment closed	sgcl
segment open	sgop
select	sel
selection	sel
selector	sel
sequence	seq
source	src
spacing	space
specification	specif
specified	specif
specifier	specif
state	st
state list	sl
stencil closed	stcl
stencil open	stop
storage	store
supported	NULL
suppress	suppr
tiling closed	tlcl
tiling open	tlop
to	NULL
tracking	track
transform(ation)	tran
unavailable	unavail
undefined	undef
undetectable	undet
unregistered	u
used	NULL
valuator	val
vector	vec
vertical	vert
viewport	vp
visibility	vis
visible	vis
visual	vis
which	NULL
window	win
with	NULL
workstation	ws
world coordinate	wc

4.3 Function names

4.3.1 List ordered alphabetically by bound name

gaccum_tran_matrix	ACCUMULATE TRANSFORMATION MATRIX
gactivate_ws	ACTIVATE WORKSTATION
gadd_set_names_nameset	ADD SET OF NAMES TO NAMESET
gadd_set_names_ndc_pic	ADD SET OF NAMES TO NDC PICTURE
gadd_set_scissors_ndc_pic	ADD SET OF SCISSORS TO NDC PICTURE
gadd_set_scissors_scissor_set	ADD SET OF SCISSORS TO SCISSOR SET

gappend_pic_part	APPEND PICTURE PART
gar_pic_part	ARCHIVE PICTURE PART
gassoc_seg_ws	ASSOCIATE SEGMENT WITH WORKSTATION
gaudit	AUDIT
gawait_event	AWAIT EVENT
gawait_in	AWAIT INPUT
gcell_array	CELL ARRAY
gclear_ws	CLEAR WORKSTATION
gclose_gks	CLOSE GKS
gclose_pic_part	CLOSE PICTURE PART
gclose_seg	CLOSE SEGMENT
gclose_stencil	CLOSE STENCIL
gclose_tiling	CLOSE TILING
gclose_ws	CLOSE WORKSTATION
gclosed_nurb_set	CLOSED NURB SET
gconic_sec_set	CONIC SECTION SET
gconv_colr	CONVERT COLOUR
gcopy_blank_real_pic_real_mf	COPY BLANK REALIZED PICTURE TO REALIZED METAFILE
gcopy_ndc_mf_pic_ndc_pic	COPY NDC METAFILE PICTURE TO NDC PICTURE
gcopy_ndc_pic_ndc_mf	COPY NDC PICTURE TO NDC METAFILE
gcopy_ndc_pic_pic_part_store	COPY NDC PICTURE TO PICTURE PART STORE
gcopy_pic_part_pic_part_store	COPY PICTURE PART FROM PICTURE PART STORE
gcopy_real_mf_pic_backdrop	COPY REALIZED METAFILE PICTURE TO BACKDROP
gcopy_real_pic_real_mf	COPY REALIZED PICTURE TO REALIZED METAFILE
gcopy_seg_ws	COPY SEGMENT FROM WORKSTATION
gcreate_out_prim	CREATE OUTPUT PRIMITIVE
gcreate_seg	CREATE SEGMENT
gcreate_stencil_bndry	CREATE STENCIL FROM BOUNDARY
gcreate_stencil_conts	CREATE STENCIL FROM CONTOURS
gcreate_store	CREATE STORAGE
gcreate_tiling_comp	CREATE TILING COMPONENT
gdeactivate_ws	DEACTIVATE WORKSTATION
gdef_in_dev	DEFINE LOGICAL INPUT DEVICE
gdel_pic_part	DELETE PICTURE PART
gdel_prims_ndc_pic	DELETE PRIMITIVES FROM NDC PICTURE
gdel_seg	DELETE SEGMENT
gdel_seg_ws	DELETE SEGMENT FROM WORKSTATION
gdel_stencil	DELETE STENCIL
gdel_store	DELETE STORAGE
gdel_tiling	DELETE TILING
gdesign	DESIGN
gell_disc_set	ELLIPTIC DISC SET
gell_sec_set	ELLIPTIC SECTOR SET
gell_seg_set	ELLIPTIC SEGMENT SET
gemergency_close_gks	EMERGENCY CLOSE GKS
gerr_hand	ERROR HANDLING
gerr_log	ERROR LOGGING
gescape	ESCAPE
geval_circ_arc_3_point	EVALUATE CIRCULAR ARC 3 POINT
geval_circ_arc_ctr	EVALUATE CIRCULAR ARC CENTRE
geval_circle	EVALUATE CIRCLE
geval_ell	EVALUATE ELLIPSE

`geval_ell_arc`	EVALUATE ELLIPTIC ARC
`geval_hyp_arc`	EVALUATE HYPERBOLIC ARC
`geval_par_arc`	EVALUATE PARABOLIC ARC
`geval_tran_matrix`	EVALUATE TRANSFORMATION MATRIX
`geval_view_map_matrix`	EVALUATE VIEW MAPPING MATRIX
`geval_view_ori_matrix`	EVALUATE VIEW ORIENTATION MATRIX
`geval_wc_ord`	EVALUATE WC ORDINATE
`gfill_area`	FILL AREA
`gfill_area_set`	FILL AREA SET
`gflush_dev_events`	FLUSH DEVICE EVENTS
`ggdp`	GENERALIZED DRAWING PRIMITIVE
`gget_choice`	GET CHOICE
`gget_glyph_name`	GET GLYPH NAME
`gget_loc`	GET LOCATOR
`gget_map_seg_name`	GET MAPPED SEGMENT NAME
`gget_map_ws_id`	GET MAPPED WORKSTATION IDENTIFIER
`gget_pick`	GET PICK
`gget_stencil_attr`	GET STENCIL ATTRIBUTE
`gget_string`	GET STRING
`gget_stroke`	GET STROKE
`gget_text_extent`	GET TEXT EXTENT
`gget_val`	GET VALUATOR
`gget_ws_status`	GET WORKSTATION STATUS
`ginit_choice`	INITIALIZE CHOICE
`ginit_in_dev`	INITIALIZE LOGICAL INPUT DEVICE
`ginit_loc`	INITIALIZE LOCATOR
`ginit_pick`	INITIALIZE PICK
`ginit_string`	INITIALIZE STRING
`ginit_stroke`	INITIALIZE STROKE
`ginit_val`	INITIALIZE VALUATOR
`ginq_area_facs`	INQUIRE AREA FACILITIES
`ginq_area_ind`	INQUIRE AREA INDEX
`ginq_area_rep`	INQUIRE AREA REPRESENTATION
`ginq_asfs`	INQUIRE ATTRIBUTE SOURCE FLAGS
`ginq_char_expan`	INQUIRE CHARACTER EXPANSION FACTOR
`ginq_char_ht`	INQUIRE CHARACTER HEIGHT
`ginq_char_space`	INQUIRE CHARACTER SPACING
`ginq_char_up_vec`	INQUIRE CHARACTER UP VECTOR
`ginq_choice_st`	INQUIRE CHOICE DEVICE STATE
`ginq_colr_facs`	INQUIRE COLOUR FACILITIES
`ginq_colr_rep`	INQUIRE COLOUR REPRESENTATION
`ginq_cur_cont_attrs`	INQUIRE CURRENT CONTOUR ATTRIBUTES
`ginq_cur_norm_tran_num`	INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER
`ginq_cur_pick_id`	INQUIRE CURRENT PICK IDENTIFIER VALUE
`ginq_def_choice_data`	INQUIRE DEFAULT CHOICE DEVICE DATA
`ginq_def_font_ind_map`	INQUIRE DEFAULT FONT INDEX MAPPING
`ginq_def_loc_data`	INQUIRE DEFAULT LOCATOR DEVICE DATA
`ginq_def_pick_data`	INQUIRE DEFAULT PICK DEVICE DATA
`ginq_def_string_data`	INQUIRE DEFAULT STRING DEVICE DATA
`ginq_def_stroke_data`	INQUIRE DEFAULT STROKE DEVICE DATA
`ginq_def_val_data`	INQUIRE DEFAULT VALUATOR DEVICE DATA
`ginq_disp_space_size`	INQUIRE DISPLAY SPACE SIZE

ginq_edge_colr_ind	INQUIRE EDGE COLOUR INDEX
ginq_edge_colr_specif	INQUIRE EDGE COLOUR SPECIFIER
ginq_edge_flag	INQUIRE EDGE FLAG
ginq_edgetype	INQUIRE EDGETYPE
ginq_edgewidth	INQUIRE EDGEWIDTH SCALE FACTOR
ginq_font_ind_map	INQUIRE FONT INDEX MAPPING
ginq_gen_ws_dt_entry	INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY
ginq_gks_dt_entry	INQUIRE GKS DESCRIPTION TABLE ENTRY'
ginq_gks_sl_entry	INQUIRE GKS STATE LIST ENTRY
ginq_global_tran_matrix	INQUIRE GLOBAL TRANSFORMATION MATRIX
ginq_in_dev_comps	INQUIRE INPUT DEVICE COMPOSITIONS
ginq_in_dev_init_values	INQUIRE INPUT DEVICE INITIAL VALUES
ginq_in_overf	INQUIRE INPUT QUEUE OVERFLOW
ginq_in_queue	INQUIRE INPUT QUEUE
ginq_in_table	INQUIRE INPUT DEVICE OPERATING MODES AND INITIAL VALUES
ginq_int_colr_ind	INQUIRE INTERIOR COLOUR INDEX
ginq_int_colr_specif	INQUIRE INTERIOR COLOUR SPECIFIER
ginq_int_style	INQUIRE INTERIOR STYLE
ginq_int_style_ind	INQUIRE INTERIOR STYLE INDEX
ginq_line_colr_ind	INQUIRE LINE COLOUR INDEX
ginq_line_colr_specif	INQUIRE LINE COLOUR SPECIFIER
ginq_line_facs	INQUIRE LINE FACILITIES
ginq_line_ind	INQUIRE LINE INDEX
ginq_line_rep	INQUIRE LINE REPRESENTATION
ginq_linetype	INQUIRE LINETYPE
ginq_linewidth	INQUIRE LINEWIDTH SCALE FACTOR
ginq_list_area_inds	INQUIRE LIST OF AREA INDICES
ginq_list_avail_fonts	INQUIRE LIST OF AVAILABLE FONTS
ginq_list_avail_gdps	INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES
ginq_list_avail_ws_types	INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
ginq_list_colr_inds	INQUIRE LIST OF COLOUR INDICES
ginq_list_line_inds	INQUIRE LIST OF LINE INDICES
ginq_list_marker_inds	INQUIRE LIST OF MARKER INDICES
ginq_list_norm_trans	INQUIRE LIST OF NORMALIZATION TRANSFORMATIONS
ginq_list_pat_inds	INQUIRE LIST OF PATTERN INDICES
ginq_list_text_inds	INQUIRE LIST OF TEXT INDICES
ginq_list_vp_in_pris	INQUIRE LIST OF VIEWPORT INPUT PRIORITIES
ginq_loc_st	INQUIRE LOCATOR DEVICE STATE
ginq_local_tran_matrix	INQUIRE LOCAL TRANSFORMATION MATRIX
ginq_marker_colr_ind	INQUIRE MARKER COLOUR INDEX
ginq_marker_colr_specif	INQUIRE MARKER COLOUR SPECIFIER
ginq_marker_facs	INQUIRE MARKER FACILITIES
ginq_marker_ind	INQUIRE MARKER INDEX
ginq_marker_rep	INQUIRE MARKER REPRESENTATION
ginq_marker_size	INQUIRE MARKER SIZE SCALE FACTOR
ginq_marker_type	INQUIRE MARKER TYPE
ginq_name_open_pic_part	INQUIRE NAME OF OPEN PICTURE PART
ginq_name_open_seg	INQUIRE NAME OF OPEN SEGMENT
ginq_name_open_stencil	INQUIRE NAME OF OPEN STENCIL
ginq_name_open_tiling	INQUIRE NAME OF OPEN TILING
ginq_nameset	INQUIRE NAMESET
ginq_norm_tran	INQUIRE NORMALIZATION TRANSFORMATION

`ginq_num_avail_in`	INQUIRE NUMBER OF AVAILABLE INPUT DEVICES
`ginq_op_st_entry`	INQUIRE OPERATING STATE ENTRY
`ginq_pat_facs`	INQUIRE PATTERN FACILITIES
`ginq_pat_ref_point`	INQUIRE PATTERN REFERENCE POINT
`ginq_pat_rep`	INQUIRE PATTERN REPRESENTATION
`ginq_pat_size`	INQUIRE PATTERN SIZE
`ginq_pick_st`	INQUIRE PICK DEVICE STATE
`ginq_pred_area_rep`	INQUIRE PREDEFINED AREA REPRESENTATION
`ginq_pred_colr_rep`	INQUIRE PREDEFINED COLOUR REPRESENTATION
`ginq_pred_line_rep`	INQUIRE PREDEFINED LINE REPRESENTATION
`ginq_pred_marker_rep`	INQUIRE PREDEFINED MARKER REPRESENTATION
`ginq_pred_pat_rep`	INQUIRE PREDEFINED PATTERN REPRESENTATION
`ginq_pred_text_rep`	INQUIRE PREDEFINED TEXT REPRESENTATION
`ginq_route_dir`	INQUIRE ROUTE DIRECTION
`ginq_scissor_mode`	INQUIRE SCISSOR MODE
`ginq_scissor_set`	INQUIRE SCISSOR SET
`ginq_seg_attrs`	INQUIRE SEGMENT ATTRIBUTES
`ginq_sel_table`	INQUIRE TABLE OF SELECTION CRITERIA
`ginq_set_active_wss`	INQUIRE SET OF ACTIVE WORKSTATIONS
`ginq_set_assoc_wss`	INQUIRE SET OF ASSOCIATED WORKSTATIONS
`ginq_set_avail_in_devs`	INQUIRE SET OF AVAILABLE INPUT DEVICES
`ginq_set_avail_meass`	INQUIRE SET OF AVAILABLE MEASURES
`ginq_set_avail_triggers`	INQUIRE SET OF AVAILABLE TRIGGERS
`ginq_set_open_audits`	INQUIRE SET OF OPEN AUDITS
`ginq_set_open_playbacks`	INQUIRE SET OF OPEN PLAYBACKS
`ginq_set_open_wss`	INQUIRE SET OF OPEN WORKSTATIONS
`ginq_set_pic_part_names`	INQUIRE SET OF PICTURE PART NAMES IN USE
`ginq_set_seg_names`	INQUIRE SET OF SEGMENT NAMES IN USE
`ginq_set_seg_names_ws`	INQUIRE SET OF SEGMENT NAMES ON WORKSTATION
`ginq_set_stencil_names`	INQUIRE SET OF STENCIL NAMES IN USE
`ginq_set_tiling_names`	INQUIRE SET OF TILING NAMES IN USE
`ginq_specif_ws_dt_entry`	INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY
`ginq_specif_ws_dt_entry_st`	INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE
`ginq_stencil_attrs`	INQUIRE STENCIL ATTRIBUTES
`ginq_string_st`	INQUIRE STRING DEVICE STATE
`ginq_stroke_st`	INQUIRE STROKE DEVICE STATE
`ginq_text_align`	INQUIRE TEXT ALIGNMENT
`ginq_text_colr_ind`	INQUIRE TEXT COLOUR INDEX
`ginq_text_colr_specif`	INQUIRE TEXT COLOUR SPECIFIER
`ginq_text_extent`	INQUIRE TEXT EXTENT
`ginq_text_facs`	INQUIRE TEXT FACILITIES
`ginq_text_font_prec`	INQUIRE TEXT FONT AND PRECISION
`ginq_text_ind`	INQUIRE TEXT INDEX
`ginq_text_path`	INQUIRE TEXT PATH
`ginq_text_rep`	INQUIRE TEXT REPRESENTATION
`ginq_text_skew_angle`	INQUIRE TEXT SKEW ANGLE
`ginq_val_st`	INQUIRE VALUATOR DEVICE STATE
`ginq_view_pris`	INQUIRE VIEW PRIORITIES
`ginq_view_rep`	INQUIRE VIEW REPRESENTATION
`ginq_vis_effects_st`	INQUIRE VISUAL EFFECTS STATE
`ginq_ws_class`	INQUIRE WORKSTATION CLASSIFICATION
`ginq_ws_conn_type`	INQUIRE WORKSTATION CONNECTION AND TYPE

ginq_ws_sl_entry	INQUIRE WORKSTATION STATE LIST ENTRY
ginq_ws_st	INQUIRE WORKSTATION STATE
ginq_ws_win_vp	INQUIRE WORKSTATION WINDOW AND VIEWPORT
ginsert_seg	INSERT SEGMENT
ginst_stencil	INSTANCE STENCIL
ginst_stencil_path	INSTANCE STENCIL ALONG PATH
ginst_stencil_seq_path	INSTANCE STENCIL SEQUENCE ALONG PATH
gmessage	MESSAGE
gnurb_set	NURB SET
gopen_gks	OPEN GKS
gopen_pic_part	OPEN PICTURE PART
gopen_stencil	OPEN STENCIL
gopen_tiling	OPEN TILING
gopen_ws	OPEN WORKSTATION
gplayback	PLAYBACK
gpolyline	POLYLINE
gpolyline_set	POLYLINE SET
gpolymarker	POLYMARKER
gproc_audit_item	PROCESS AUDIT ITEM
gread_item_audit	READ ITEM FROM AUDIT
gread_item_func_name_audit	READ ITEM FUNCTION NAME FROM AUDIT
gremove_backdrop	REMOVE BACKDROP
gremove_set_names_nameset	REMOVE SET OF NAMES FROM NAMESET
gremove_set_names_ndc_pic	REMOVE SET OF NAMES FROM NDC PICTURE
gremove_set_scissors_ndc_pic	REMOVE SET OF SCISSORS FROM NDC PICTURE
gremove_set_scissors_scissor_set	REMOVE SET OF SCISSORS FROM SCISSOR SET
grename_pic_part	RENAME PICTURE PART
grename_seg	RENAME SEGMENT
grename_stencil	RENAME STENCIL
grename_tiling	RENAME TILING
greopen_pic_part	REOPEN PICTURE PART
greorder_ndc_pic	REORDER NDC PICTURE
greq_choice	REQUEST CHOICE
greq_in	REQUEST INPUT
greq_loc	REQUEST LOCATOR
greq_pick	REQUEST PICK
greq_string	REQUEST STRING
greq_stroke	REQUEST STROKE
greq_val	REQUEST VALUATOR
greset_specif_ws_dt_entry_st	RESET SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE
grestore_gks_sl	RESTORE GKS STATE LIST
grestore_ws_sl	RESTORE WORKSTATION STATE LIST
gret_pic_part_ar	RETRIEVE PICTURE PART FROM ARCHIVE
groute	ROUTE
gsample_choice	SAMPLE CHOICE
gsample_in	SAMPLE INPUT
gsample_loc	SAMPLE LOCATOR
gsample_pick	SAMPLE PICK
gsample_string	SAMPLE STRING
gsample_stroke	SAMPLE STROKE
gsample_val	SAMPLE VALUATOR
gsave_gks_sl	SAVE GKS STATE LIST

gsave_ws_sl	SAVE WORKSTATION STATE LIST
gsel_norm_tran	SELECT NORMALIZATION TRANSFORMATION
gset_area_ind	SET AREA INDEX
gset_area_rep	SET AREA REPRESENTATION
gset_asfs	SET ATTRIBUTE SOURCE FLAGS
gset_char_code	SET CHARACTER CODE
gset_char_expan	SET CHARACTER EXPANSION FACTOR
gset_char_ht	SET CHARACTER HEIGHT
gset_char_space	SET CHARACTER SPACING
gset_char_up_vec	SET CHARACTER UP VECTOR
gset_choice_mode	SET CHOICE MODE
gset_colr_rep	SET COLOUR REPRESENTATION
gset_cont_attr	SET CONTOUR ATTRIBUTE
gset_det	SET DETECTABILITY
gset_edge_colr_ind	SET EDGE COLOUR INDEX
gset_edge_colr_specif	SET EDGE COLOUR SPECIFIER
gset_edge_flag	SET EDGE FLAG
gset_edgetype	SET EDGETYPE
gset_edgewidth	SET EDGEWIDTH SCALE FACTOR
gset_err_hand	SET ERROR HANDLING
gset_font_ind_map	SET FONT INDEX MAPPING
gset_global_tran_matrix	SET GLOBAL TRANSFORMATION MATRIX
gset_highl	SET HIGHLIGHTING
gset_in_dev_mode	SET LOGICAL INPUT DEVICE MODE
gset_int_colr_ind	SET INTERIOR COLOUR INDEX
gset_int_colr_specif	SET INTERIOR COLOUR SPECIFIER
gset_int_style	SET INTERIOR STYLE
gset_int_style_ind	SET INTERIOR STYLE INDEX
gset_line_colr_ind	SET LINE COLOUR INDEX
gset_line_colr_specif	SET LINE COLOUR SPECIFIER
gset_line_ind	SET LINE INDEX
gset_line_rep	SET LINE REPRESENTATION
gset_linetype	SET LINETYPE
gset_linewidth	SET LINEWIDTH SCALE FACTOR
gset_loc_mode	SET LOCATOR MODE
gset_local_tran_matrix	SET LOCAL TRANSFORMATION MATRIX
gset_marker_colr_ind	SET LINE COLOUR INDEX
gset_marker_colr_specif	SET MARKER COLOUR SPECIFIER
gset_marker_ind	SET MARKER INDEX
gset_marker_rep	SET MARKER REPRESENTATION
gset_marker_size	SET MARKER SIZE SCALE FACTOR
gset_marker_type	SET MARKER TYPE
gset_nameset	SET NAMESET
gset_ndc_pic_prim_attr	SET NDC PICTURE PRIMITIVE ATTRIBUTE
gset_norm_tran_num	SET NORMALIZATION TRANSFORMATION NUMBER
gset_pat_ref_point	SET PATTERN REFERENCE POINT
gset_pat_rep	SET PATTERN REPRESENTATION
gset_pat_size	SET PATTERN SIZE
gset_pick_id	SET PICK IDENTIFIER
gset_pick_mode	SET PICK MODE
gset_prim_attr	SET PRIMITIVE ATTRIBUTE
gset_rep	SET REPRESENTATION

`gset_scissor_mode`	SET SCISSOR MODE
`gset_scissor_set`	SET SCISSOR SET
`gset_seg_attr`	SET SEGMENT ATTRIBUTE
`gset_seg_pri`	SET SEGMENT PRIORITY
`gset_seg_tran`	SET SEGMENT TRANSFORMATION
`gset_stencil_attr`	SET STENCIL ATTRIBUTE
`gset_string_mode`	SET STRING MODE
`gset_stroke_mode`	SET STROKE MODE
`gset_text_align`	SET TEXT ALIGNMENT
`gset_text_colr_ind`	SET TEXT COLOUR INDEX
`gset_text_colr_specif`	SET TEXT COLOUR SPECIFIER
`gset_text_font_prec`	SET TEXT FONT AND PRECISION
`gset_text_ind`	SET TEXT INDEX
`gset_text_path`	SET TEXT PATH
`gset_text_rep`	SET TEXT REPRESENTATION
`gset_text_skew_angle`	SET TEXT SKEW ANGLE
`gset_val_mode`	SET VALUATOR MODE
`gset_view`	SET VIEW
`gset_view_pri`	SET VIEW PRIORITY
`gset_view_sel_crit`	SET VIEW SELECTION CRITERION
`gset_vis`	SET VISIBILITY
`gset_vp`	SET VIEWPORT
`gset_vp_in_pri`	SET VIEWPORT INPUT PRIORITY
`gset_win`	SET WINDOW
`gset_win_vp`	SET WINDOW AND VIEWPORT
`gset_ws_sel_crit`	SET WORKSTATION SELECTION CRITERION
`gset_ws_vis_effects`	SET WORKSTATION VISUAL EFFECTS
`gset_ws_vp`	SET WORKSTATION VIEWPORT
`gset_ws_win`	SET WORKSTATION WINDOW
`gset_ws_win_vp`	SET WORKSTATION WINDOW AND VIEWPORT
`gtext`	TEXT
`gwrite_user_rec_audit`	WRITE USER RECORD TO AUDIT

4.3.2 List ordered alphabetically by GKS name

ACCUMULATE TRANSFORMATION MATRIX	`gaccum_tran_matrix`
ACTIVATE WORKSTATION	`gactivate_ws`
ADD SET OF NAMES TO NAMESET	`gadd_set_names_nameset`
ADD SET OF NAMES TO NDC PICTURE	`gadd_set_names_ndc_pic`
ADD SET OF SCISSORS TO NDC PICTURE	`gadd_set_scissors_ndc_pic`
ADD SET OF SCISSORS TO SCISSOR SET	`gadd_set_scissors_scissor_set`
APPEND PICTURE PART	`gappend_pic_part`
ARCHIVE PICTURE PART	`gar_pic_part`
ASSOCIATE SEGMENT WITH WORKSTATION	`gassoc_seg_ws`
AUDIT	`gaudit`
AWAIT EVENT	`gawait_event`
AWAIT INPUT	`gawait_in`
CELL ARRAY	`gcell_array`
CLEAR WORKSTATION	`gclear_ws`
CLOSE GKS	`gclose_gks`
CLOSE PICTURE PART	`gclose_pic_part`
CLOSE SEGMENT	`gclose_seg`

CLOSE STENCIL	gclose_stencil
CLOSE TILING	gclose_tiling
CLOSE WORKSTATION	gclose_ws
CLOSED NURB SET	gclosed_nurb_set
CONIC SECTION SET	gconic_sec_set
CONVERT COLOUR	gconv_colr
COPY BLANK REALIZED PICTURE TO REALIZED METAFILE	gcopy_blank_real_pic_real_mf
COPY NDC METAFILE PICTURE TO NDC PICTURE	gcopy_ndc_mf_pic_ndc_pic
COPY NDC PICTURE TO NDC METAFILE	gcopy_ndc_pic_ndc_mf
COPY NDC PICTURE TO PICTURE PART STORE	gcopy_ndc_pic_pic_part_store
COPY PICTURE PART FROM PICTURE PART STORE	gcopy_pic_part_pic_part_store
COPY REALIZED METAFILE PICTURE TO BACKDROP	gcopy_real_mf_pic_backdrop
COPY REALIZED PICTURE TO REALIZED METAFILE	gcopy_real_pic_real_mf
COPY SEGMENT FROM WORKSTATION	gcopy_seg_ws
CREATE OUTPUT PRIMITIVE	gcreate_out_prim
CREATE SEGMENT	gcreate_seg
CREATE STENCIL FROM BOUNDARY	gcreate_stencil_bndry
CREATE STENCIL FROM CONTOURS	gcreate_stencil_conts
CREATE STORAGE	gcreate_store
CREATE TILING COMPONENT	gcreate_tiling_comp
DEACTIVATE WORKSTATION	gdeactivate_ws
DEFINE LOGICAL INPUT DEVICE	gdef_in_dev
DELETE PICTURE PART	gdel_pic_part
DELETE PRIMITIVES FROM NDC PICTURE	gdel_prims_ndc_pic
DELETE SEGMENT	gdel_seg
DELETE SEGMENT FROM WORKSTATION	gdel_seg_ws
DELETE STENCIL	gdel_stencil
DELETE STORAGE	gdel_store
DELETE TILING	gdel_tiling
DESIGN	gdesign
ELLIPTIC DISC SET	gell_disc_set
ELLIPTIC SECTOR SET	gell_sec_set
ELLIPTIC SEGMENT SET	gell_seg_set
EMERGENCY CLOSE GKS	gemergency_close_gks
ERROR HANDLING	gerr_hand
ERROR LOGGING	gerr_log
ESCAPE	gescape
EVALUATE CIRCLE	geval_circle
EVALUATE CIRCULAR ARC 3 POINT	geval_circ_arc_3_point
EVALUATE CIRCULAR ARC CENTRE	geval_circ_arc_ctr
EVALUATE ELLIPSE	geval_ell
EVALUATE ELLIPTIC ARC	geval_ell_arc
EVALUATE HYPERBOLIC ARC	geval_hyp_arc
EVALUATE PARABOLIC ARC	geval_par_arc
EVALUATE TRANSFORMATION MATRIX	geval_tran_matrix
EVALUATE VIEW MAPPING MATRIX	geval_view_map_matrix
EVALUATE VIEW ORIENTATION MATRIX	geval_view_ori_matrix
EVALUATE WC ORDINATE	geval_wc_ord
FILL AREA	gfill_area
FILL AREA SET	gfill_area_set
FLUSH DEVICE EVENTS	gflush_dev_events
GENERALIZED DRAWING PRIMITIVE	ggdp

GET CHOICE	gget_choice
GET GLYPH NAME	gget_glyph_name
GET LOCATOR	gget_loc
GET MAPPED SEGMENT NAME	gget_map_seg_name
GET MAPPED WORKSTATION IDENTIFIER	gget_map_ws_id
GET PICK	gget_pick
GET STENCIL ATTRIBUTE	gget_stencil_attr
GET STRING	gget_string
GET STROKE	gget_stroke
GET TEXT EXTENT	gget_text_extent
GET VALUATOR	gget_val
GET WORKSTATION STATUS	gget_ws_status
INITIALIZE CHOICE	ginit_choice
INITIALIZE LOCATOR	ginit_loc
INITIALIZE LOGICAL INPUT DEVICE	ginit_in_dev
INITIALIZE PICK	ginit_pick
INITIALIZE STRING	ginit_string
INITIALIZE STROKE	ginit_stroke
INITIALIZE VALUATOR	ginit_val
INQUIRE AREA FACILITIES	ginq_area_facs
INQUIRE AREA INDEX	ginq_area_ind
INQUIRE AREA REPRESENTATION	ginq_area_rep
INQUIRE ATTRIBUTE SOURCE FLAGS	ginq_asfs
INQUIRE CHARACTER EXPANSION FACTOR	ginq_char_expan
INQUIRE CHARACTER HEIGHT	ginq_char_ht
INQUIRE CHARACTER SPACING	ginq_char_space
INQUIRE CHARACTER UP VECTOR	ginq_char_up_vec
INQUIRE CHOICE DEVICE STATE	ginq_choice_st
INQUIRE COLOUR FACILITIES	ginq_colr_facs
INQUIRE COLOUR REPRESENTATION	ginq_colr_rep
INQUIRE CURRENT CONTOUR ATTRIBUTES	ginq_cur_cont_attrs
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	ginq_cur_norm_tran_num
INQUIRE CURRENT PICK IDENTIFIER VALUE	ginq_cur_pick_id
INQUIRE DEFAULT CHOICE DEVICE DATA	ginq_def_choice_data
INQUIRE DEFAULT FONT INDEX MAPPING	ginq_def_font_ind_map
INQUIRE DEFAULT LOCATOR DEVICE DATA	ginq_def_loc_data
INQUIRE DEFAULT PICK DEVICE DATA	ginq_def_pick_data
INQUIRE DEFAULT STRING DEVICE DATA	ginq_def_string_data
INQUIRE DEFAULT STROKE DEVICE DATA	ginq_def_stroke_data
INQUIRE DEFAULT VALUATOR DEVICE DATA	ginq_def_val_data
INQUIRE DISPLAY SPACE SIZE	ginq_disp_space_size
INQUIRE EDGE COLOUR INDEX	ginq_edge_colr_ind
INQUIRE EDGE COLOUR SPECIFIER	ginq_edge_colr_specif
INQUIRE EDGE FLAG	ginq_edge_flag
INQUIRE EDGETYPE	ginq_edgetype
INQUIRE EDGEWIDTH SCALE FACTOR	ginq_edgewidth
INQUIRE FONT INDEX MAPPING	ginq_font_ind_map
INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY	ginq_gen_ws_dt_entry
INQUIRE GKS DESCRIPTION TABLE ENTRY'	ginq_gks_dt_entry
INQUIRE GKS STATE LIST ENTRY	ginq_gks_sl_entry
INQUIRE GLOBAL TRANSFORMATION MATRIX	ginq_global_tran_matrix
INQUIRE INPUT DEVICE COMPOSITIONS	ginq_in_dev_comps

INQUIRE INPUT DEVICE INITIAL VALUES	ginq_in_dev_init_values
INQUIRE INPUT DEVICE OPERATING MODES AND INITIAL VALUES	ginq_in_table
INQUIRE INPUT QUEUE	ginq_in_queue
INQUIRE INPUT QUEUE OVERFLOW	ginq_in_overf
INQUIRE INTERIOR COLOUR INDEX	ginq_int_colr_ind
INQUIRE INTERIOR COLOUR SPECIFIER	ginq_int_colr_specif
INQUIRE INTERIOR STYLE	ginq_int_style
INQUIRE INTERIOR STYLE INDEX	ginq_int_style_ind
INQUIRE LINE COLOUR INDEX	ginq_line_colr_ind
INQUIRE LINE COLOUR SPECIFIER	ginq_line_colr_specif
INQUIRE LINE FACILITIES	ginq_line_facs
INQUIRE LINE INDEX	ginq_line_ind
INQUIRE LINE REPRESENTATION	ginq_line_rep
INQUIRE LINETYPE	ginq_linetype
INQUIRE LINEWIDTH SCALE FACTOR	ginq_linewidth
INQUIRE LIST OF AREA INDICES	ginq_list_area_inds
INQUIRE LIST OF AVAILABLE FONTS	ginq_list_avail_fonts
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	ginq_list_avail_gdps
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	ginq_list_avail_ws_types
INQUIRE LIST OF COLOUR INDICES	ginq_list_colr_inds
INQUIRE LIST OF LINE INDICES	ginq_list_line_inds
INQUIRE LIST OF MARKER INDICES	ginq_list_marker_inds
INQUIRE LIST OF NORMALIZATION TRANSFORMATIONS	ginq_list_norm_trans
INQUIRE LIST OF PATTERN INDICES	ginq_list_pat_inds
INQUIRE LIST OF TEXT INDICES	ginq_list_text_inds
INQUIRE LIST OF VIEWPORT INPUT PRIORITIES	ginq_list_vp_in_pris
INQUIRE LOCAL TRANSFORMATION MATRIX	ginq_local_tran_matrix
INQUIRE LOCATOR DEVICE STATE	ginq_loc_st
INQUIRE MARKER COLOUR INDEX	ginq_marker_colr_ind
INQUIRE MARKER COLOUR SPECIFIER	ginq_marker_colr_specif
INQUIRE MARKER FACILITIES	ginq_marker_facs
INQUIRE MARKER INDEX	ginq_marker_ind
INQUIRE MARKER REPRESENTATION	ginq_marker_rep
INQUIRE MARKER SIZE SCALE FACTOR	ginq_marker_size
INQUIRE MARKER TYPE	ginq_marker_type
INQUIRE NAME OF OPEN PICTURE PART	ginq_name_open_pic_part
INQUIRE NAME OF OPEN SEGMENT	ginq_name_open_seg
INQUIRE NAME OF OPEN STENCIL	ginq_name_open_stencil
INQUIRE NAME OF OPEN TILING	ginq_name_open_tiling
INQUIRE NAMESET	ginq_nameset
INQUIRE NORMALIZATION TRANSFORMATION	ginq_norm_tran
INQUIRE NUMBER OF AVAILABLE INPUT DEVICES	ginq_num_avail_in
INQUIRE OPERATING STATE ENTRY	ginq_op_st_entry
INQUIRE PATTERN FACILITIES	ginq_pat_facs
INQUIRE PATTERN REFERENCE POINT	ginq_pat_ref_point
INQUIRE PATTERN REPRESENTATION	ginq_pat_rep
INQUIRE PATTERN SIZE	ginq_pat_size
INQUIRE PICK DEVICE STATE	ginq_pick_st
INQUIRE PREDEFINED AREA REPRESENTATION	ginq_pred_area_rep
INQUIRE PREDEFINED COLOUR REPRESENTATION	ginq_pred_colr_rep
INQUIRE PREDEFINED LINE REPRESENTATION	ginq_pred_line_rep
INQUIRE PREDEFINED MARKER REPRESENTATION	ginq_pred_marker_rep

INQUIRE PREDEFINED PATTERN REPRESENTATION	ginq_pred_pat_rep
INQUIRE PREDEFINED TEXT REPRESENTATION	ginq_pred_text_rep
INQUIRE ROUTE DIRECTION	ginq_route_dir
INQUIRE SCISSOR MODE	ginq_scissor_mode
INQUIRE SCISSOR SET	ginq_scissor_set
INQUIRE SEGMENT ATTRIBUTES	ginq_seg_attrs
INQUIRE SET OF ACTIVE WORKSTATIONS	ginq_set_active_wss
INQUIRE SET OF ASSOCIATED WORKSTATIONS	ginq_set_assoc_wss
INQUIRE SET OF AVAILABLE INPUT DEVICES	ginq_set_avail_in_devs
INQUIRE SET OF AVAILABLE MEASURES	ginq_set_avail_meass
INQUIRE SET OF AVAILABLE TRIGGERS	ginq_set_avail_triggers
INQUIRE SET OF OPEN AUDITS	ginq_set_open_audits
INQUIRE SET OF OPEN PLAYBACKS	ginq_set_open_playbacks
INQUIRE SET OF OPEN WORKSTATIONS	ginq_set_open_wss
INQUIRE SET OF PICTURE PART NAMES IN USE	ginq_set_pic_part_names
INQUIRE SET OF SEGMENT NAMES IN USE	ginq_set_seg_names
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	ginq_set_seg_names_ws
INQUIRE SET OF STENCIL NAMES IN USE	ginq_set_stencil_names
INQUIRE SET OF TILING NAMES IN USE	ginq_set_tiling_names
INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY	ginq_specif_ws_dt_entry
INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE	ginq_specif_ws_dt_entry_st
INQUIRE STENCIL ATTRIBUTES	ginq_stencil_attrs
INQUIRE STRING DEVICE STATE	ginq_string_st
INQUIRE STROKE DEVICE STATE	ginq_stroke_st
INQUIRE TABLE OF SELECTION CRITERIA	ginq_sel_table
INQUIRE TEXT ALIGNMENT	ginq_text_align
INQUIRE TEXT COLOUR INDEX	ginq_text_colr_ind
INQUIRE TEXT COLOUR SPECIFIER	ginq_text_colr_specif
INQUIRE TEXT EXTENT	ginq_text_extent
INQUIRE TEXT FACILITIES	ginq_text_facs
INQUIRE TEXT FONT AND PRECISION	ginq_text_font_prec
INQUIRE TEXT INDEX	ginq_text_ind
INQUIRE TEXT PATH	ginq_text_path
INQUIRE TEXT REPRESENTATION	ginq_text_rep
INQUIRE TEXT SKEW ANGLE	ginq_text_skew_angle
INQUIRE VALUATOR DEVICE STATE	ginq_val_st
INQUIRE VIEW PRIORITIES	ginq_view_pris
INQUIRE VIEW REPRESENTATION	ginq_view_rep
INQUIRE VISUAL EFFECTS STATE	ginq_vis_effects_st
INQUIRE WORKSTATION CLASSIFICATION	ginq_ws_class
INQUIRE WORKSTATION CONNECTION AND TYPE	ginq_ws_conn_type
INQUIRE WORKSTATION STATE	ginq_ws_st
INQUIRE WORKSTATION STATE LIST ENTRY	ginq_ws_sl_entry
INQUIRE WORKSTATION WINDOW AND VIEWPORT	ginq_ws_win_vp
INSERT SEGMENT	ginsert_seg
INSTANCE STENCIL	ginst_stencil
INSTANCE STENCIL ALONG PATH	ginst_stencil_path
INSTANCE STENCIL SEQUENCE ALONG PATH	ginst_stencil_seq_path
MESSAGE	gmessage
NURB SET	gnurb_set
OPEN GKS	gopen_gks
OPEN PICTURE PART	gopen_pic_part

OPEN STENCIL	gopen_stencil
OPEN TILING	gopen_tiling
OPEN WORKSTATION	gopen_ws
PLAYBACK	gplayback
POLYLINE	gpolyline
POLYLINE SET	gpolyline_set
POLYMARKER	gpolymarker
PROCESS AUDIT ITEM	gproc_audit_item
READ ITEM FROM AUDIT	gread_item_audit
READ ITEM FUNCTION NAME FROM AUDIT	gread_item_func_name_audit
REMOVE BACKDROP	gremove_backdrop
REMOVE SET OF NAMES FROM NAMESET	gremove_set_names_nameset
REMOVE SET OF NAMES FROM NDC PICTURE	gremove_set_names_ndc_pic
REMOVE SET OF SCISSORS FROM NDC PICTURE	gremove_set_scissors_ndc_pic
REMOVE SET OF SCISSORS FROM SCISSOR SET	gremove_set_scissors_scissor_set
RENAME PICTURE PART	grename_pic_part
RENAME SEGMENT	grename_seg
RENAME STENCIL	grename_stencil
RENAME TILING	grename_tiling
REOPEN PICTURE PART	greopen_pic_part
REORDER NDC PICTURE	greorder_ndc_pic
REQUEST CHOICE	greq_choice
REQUEST INPUT	greq_in
REQUEST LOCATOR	greq_loc
REQUEST PICK	greq_pick
REQUEST STRING	greq_string
REQUEST STROKE	greq_stroke
REQUEST VALUATOR	greq_val
RESET SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE	greset_specif_ws_dt_entry_st
RESTORE GKS STATE LIST	grestore_gks_sl
RESTORE WORKSTATION STATE LIST	grestore_ws_sl
RETRIEVE PICTURE PART FROM ARCHIVE	gret_pic_part_ar
ROUTE	groute
SAMPLE CHOICE	gsample_choice
SAMPLE INPUT	gsample_in
SAMPLE LOCATOR	gsample_loc
SAMPLE PICK	gsample_pick
SAMPLE STRING	gsample_string
SAMPLE STROKE	gsample_stroke
SAMPLE VALUATOR	gsample_val
SAVE GKS STATE LIST	gsave_gks_sl
SAVE WORKSTATION STATE LIST	gsave_ws_sl
SELECT NORMALIZATION TRANSFORMATION	gsel_norm_tran
SET AREA INDEX	gset_area_ind
SET AREA REPRESENTATION	gset_area_rep
SET ATTRIBUTE SOURCE FLAGS	gset_asfs
SET CHARACTER CODE	gset_char_code
SET CHARACTER EXPANSION FACTOR	gset_char_expan
SET CHARACTER HEIGHT	gset_char_ht
SET CHARACTER SPACING	gset_char_space
SET CHARACTER UP VECTOR	gset_char_up_vec
SET CHOICE MODE	gset_choice_mode

SET COLOUR REPRESENTATION	gset_colr_rep
SET CONTOUR ATTRIBUTE	gset_cont_attr
SET DETECTABILITY	gset_det
SET EDGE COLOUR INDEX	gset_edge_colr_ind
SET EDGE COLOUR SPECIFIER	gset_edge_colr_specif
SET EDGE FLAG	gset_edge_flag
SET EDGETYPE	gset_edgetype
SET EDGEWIDTH SCALE FACTOR	gset_edgewidth
SET ERROR HANDLING	gset_err_hand
SET FONT INDEX MAPPING	gset_font_ind_map
SET GLOBAL TRANSFORMATION MATRIX	gset_global_tran_matrix
SET HIGHLIGHTING	gset_highl
SET INTERIOR COLOUR INDEX	gset_int_colr_ind
SET INTERIOR COLOUR SPECIFIER	gset_int_colr_specif
SET INTERIOR STYLE	gset_int_style
SET INTERIOR STYLE INDEX	gset_int_style_ind
SET LINE COLOUR INDEX	gset_line_colr_ind
SET LINE COLOUR INDEX	gset_marker_colr_ind
SET LINE COLOUR SPECIFIER	gset_line_colr_specif
SET LINE INDEX	gset_line_ind
SET LINE REPRESENTATION	gset_line_rep
SET LINETYPE	gset_linetype
SET LINEWIDTH SCALE FACTOR	gset_linewidth
SET LOCAL TRANSFORMATION MATRIX	gset_local_tran_matrix
SET LOCATOR MODE	gset_loc_mode
SET LOGICAL INPUT DEVICE MODE	gset_in_dev_mode
SET MARKER COLOUR SPECIFIER	gset_marker_colr_specif
SET MARKER INDEX	gset_marker_ind
SET MARKER REPRESENTATION	gset_marker_rep
SET MARKER SIZE SCALE FACTOR	gset_marker_size
SET MARKER TYPE	gset_marker_type
SET NAMESET	gset_nameset
SET NDC PICTURE PRIMITIVE ATTRIBUTE	gset_ndc_pic_prim_attr
SET NORMALIZATION TRANSFORMATION NUMBER	gset_norm_tran_num
SET PATTERN REFERENCE POINT	gset_pat_ref_point
SET PATTERN REPRESENTATION	gset_pat_rep
SET PATTERN SIZE	gset_pat_size
SET PICK IDENTIFIER	gset_pick_id
SET PICK MODE	gset_pick_mode
SET PRIMITIVE ATTRIBUTE	gset_prim_attr
SET REPRESENTATION	gset_rep
SET SCISSOR MODE	gset_scissor_mode
SET SCISSOR SET	gset_scissor_set
SET SEGMENT ATTRIBUTE	gset_seg_attr
SET SEGMENT PRIORITY	gset_seg_pri
SET SEGMENT TRANSFORMATION	gset_seg_tran
SET STENCIL ATTRIBUTE	gset_stencil_attr
SET STRING MODE	gset_string_mode
SET STROKE MODE	gset_stroke_mode
SET TEXT ALIGNMENT	gset_text_align
SET TEXT COLOUR INDEX	gset_text_colr_ind
SET TEXT COLOUR SPECIFIER	gset_text_colr_specif

SET TEXT FONT AND PRECISION	gset_text_font_prec
SET TEXT INDEX	gset_text_ind
SET TEXT PATH	gset_text_path
SET TEXT REPRESENTATION	gset_text_rep
SET TEXT SKEW ANGLE	gset_text_skew_angle
SET VALUATOR MODE	gset_val_mode
SET VIEW	gset_view
SET VIEW PRIORITY	gset_view_pri
SET VIEW SELECTION CRITERION	gset_view_sel_crit
SET VIEWPORT	gset_vp
SET VIEWPORT INPUT PRIORITY	gset_vp_in_pri
SET VISIBILITY	gset_vis
SET WINDOW	gset_win
SET WINDOW AND VIEWPORT	gset_win_vp
SET WORKSTATION SELECTION CRITERION	gset_ws_sel_crit
SET WORKSTATION VIEWPORT	gset_ws_vp
SET WORKSTATION VISUAL EFFECTS	gset_ws_vis_effects
SET WORKSTATION WINDOW	gset_ws_win
SET WORKSTATION WINDOW AND VIEWPORT	gset_ws_win_vp
TEXT	gtext
WRITE USER RECORD TO AUDIT	gwrite_user_rec_audit

5 Type definitions

5.1 Mapping of GKS data types

The GKS document specifies a set of abstract data types. This clause gives the mapping from those data types to the data types defined in this part of ISO/IEC 8651.

GKS data type	C binding data type
Archive name	`Gint`
Archive specifier	`void *`
Audit identifier	`Gint`
Audit specifier	`void *`
Audit user data	`Gaudit_user_data`
Character code	`char`
Colour characteristics	`Gcolr_chars`
Error file	`char *`
Escape function identifier	`Gint`
Escape input data record	`Gescape_in_data`
Escape output data record	`Gescape_out_data`
GDP data record	`Ggdp_data`
GDP function identifier	`Gint`
Glyph name	`Gint`
ISO/IEC 9541 font name	`Gint`
Input data record	`Gin_data`
Integer	`Gint`
Measure process identifier	`Gint`
NDC metafile specifier	`void *`
Name	`Gint`
Nonnegative integer	`Gint`
Pick identifier	`Gint`
Picture identifier	`Gint`
Picture part name	`Gint`
Real number	`Gfloat`
Realized metafile specifier	`void *`
Scissor identifier	`Gint`
Segment name	`Gint`
Stencil name	`Gint`
Tiling name	`Gint`
Trigger process identifier	`Gint`
Workstation generic type	`Gint`
Workstation identifier	`Gint`
Workstation specifier	`void *`

5.2 Environment-defined type definitions

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

An implementation shall document the C types used for GKS/C types **`Gfloat`** and **`Gint`**.

`Gfloat` floating point number

This data type shall be defined by the implementation as a floating point type suitable for use within

GKS/C. Suggest:

```
                typedef     float                    Gfloat;
```

Gint integer

This data type shall be defined by the implementation as a signed integral type suitable for use within GKS/C. Suggest:

```
                typedef     int                      Gint;
```

size_t

An additional environment-defined data type is the type **size_t**. Depending on the environment, **size_t** can be defined as **unsigned int** or **unsigned long**. **size_t** is defined in the headers **<stdio.h>**, **<stdlib.h>**, **<stddef.h>**, **<string.h>**, **<time.h>**.

5.3 Implementation dependent type definitions

This subclause presents the skeleton structure of the implementation dependent data types.

REMARKS:

- In order to produce syntactically correct data types, several implementation and registration dependent parts are of type **Gdata**. They can be replaced by any other structure.
- The data of registered prompt and echo types are represented by structures with the names **pet_rn** (n the prompt and echo type), whereas the data of unregistered prompt and echo types are represented by structures with the names **pet_un** (-n the prompt and echo type).
- The data of registered and unresgistered GDPs, escape input and escape output are denoted in a similar way.
- For the input data, the mandatory part, as specified in GKS (ISO/IEC 7942–1), is defined outside the union part of **Gxxx_data**. This mandatory part is implementation independent. Some of the **pet_rn** fields are given as structures of other type than **Gdata** for illustrative reasons. Thus, e.g. the **pet_r2** field of **Gchoice_data** contains a list of prompts, because GKS defines prompt and echo type 2 this way for the CHOICE device;

Gaudit_user_data audut user data record

```
 typedef struct {
    Gint      id;        /* identifier          */
    union {
       Gdata  impl_dep;  /* impl. dependent  */
    } data;
 } Gaudit_user_data;
```

Gchoice_data choice data

```
typedef struct {
     union Gchoice_pets {
         struct Gchoice_pet_r1 {
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_r1;  /* pet 1 data */
         struct Gchoice_pet_r2 {
                    Gint       num_prs;        /* number of prompts */
                    Gpr_flag   *prs;           /* prompt array */
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_r2;  /* pet 2 data */
         struct Gchoice_pet_r3 {
                    Gint       num_strings;    /* number of choice strings */
                    char       **strings;      /* array of choice strings */
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_r3;  /* pet 3 data */
         struct Gchoice_pet_r4 {
                    Gint       num_strings;    /* number of choice strings */
                    char       **strings;      /* array of choice strings */
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_r4;  /* pet 4 data */
         struct Gchoice_pet_r5 {
                    Gint       seg_name;       /* segment name */
                    Gint       num_pick_ids;   /* number of pick identifiers */
                    Gint       *pick_ids;      /* array of pick identifiers */
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_r5;  /* pet 5 data */
         struct Gchoice_pet_u1 {
                    Gdata      impl_dep;       /* impl. dep. */
         } pet_u1;  /* pet -1 data */
         /*. . . impl. defined PET's */
       } pets;
} Gchoice_data;
```

Gcolr_chars colour characteristics

```
typedef struct {
    enum Gcolr_chars_type {GTYPE_CIELUV, GTYPE_OTHER } type;
    union {
        Gcieluv   prim_colr[3];   /* CIELUV primary colours */
        Gdata     impl_dep;       /* impl.dep. */
    } chars;
} Gcolr_chars;
```

`Gescape_in_data` escape input data record

```
typedef union {
    struct Gescape_in_r1 {
                  Gdata   impl_dep;   /* impl. dep. */
    } escape_r1;  /* escape 1 data */
    struct Gescape_in_u1 {
                  Gdata   impl_dep;   /* impl. dep. */
    } escape_u1;  /* escape -1 data */
    /* etc. */
} Gescape_in_data;
```

`Gescape_out_data` escape output data record

```
typedef union {
    struct Gescape_out_r1 {
                  Gdata   impl_dep;   /* impl. dep. */
    } escape_r1;  /* escape 1 data */
    struct Gescape_out_u1 {
                  Gdata   impl_dep;   /* impl. dep. */
    } escape_u1;  /* escape -1 data */
    /* etc. */
} Gescape_out_data;
```

`Ggdp_data` gdp data record

```
typedef union {
    struct Ggdp_r1 {
                Gdata   impl_dep;   /* impl. dep. */
    } gdp_r1;   /* gdp 1 data */
    struct Ggdp_u1 {
                Gdata   impl_dep;   /* impl. dep. */
    } gdp_u1;   /* gdp -1 data */
    /* etc. */
} Ggdp_data;
```

Gloc_data locator data record

```
 typedef struct {
     union Gloc_pets {
         struct Gloc_pet_r1 {
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r1;   /* pet 1 data */
         struct Gloc_pet_r2 {
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r2;   /* pet 2 data */
         struct Gloc_pet_r3 {
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r3;   /* pet 3 data */
         struct Gloc_pet_r4 {
                     Gattr_ctrl_flag         attr_ctrl_flag;
                     /* attribute            control flag */
                     Gline_attrs             line_attrs;
                     /* line                 attributes */
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r4;   /* pet 4 data */
         struct Gloc_pet_r5 {
                     Gattr_ctrl_flag         attr_ctrl_flag;
                     /* attribute            control flag */
                     Gline_fill_ctrl_flag    line_fill_ctrl_flag;
                     /* line/area            control flag */
                     union Gloc_attrs {
                                             Gline_attrs   line_attrs;
                                             /* line       attrs. */
                                             Garea_attrs   area_attrs;
                                             /* interior   attrs. */
                     } attrs;
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r5;   /* pet 5 data */
         struct Gloc_pet_r6 {
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_r6;   /* pet 6 data */
         struct Gloc_pet_u1 {
                     Gdata                   impl_dep;     /* impl. dep. */
         } pet_u1;   /* pet -1 data */
         /*. . . impl. defined PET's */
     } pets;
 } Gloc_data;
```

`Gpick_data` pick data

```
typedef struct {
    union Gpick_pets {
        struct Gpick_pet_r1 {
                    Gdata       impl_dep;     /* impl. dep. */
        } pet_r1;   /* pet 1 data */
        struct Gpick_pet_r2 {
                    Gint        pick_id;      /* pick identifier*/
                    Gint        seg_name;     /* segment name*/
                    Gnameset    set_names;    /* set of names*/
                    Gdata       impl_dep;     /* impl. dep. */
        } pet_r2;   /* pet 2 data */
        struct Gpick_pet_r3 {
                    Gnameset    set_names;    /* set of names*/
                    Gdata       impl_dep;     /* impl. dep. */
        } pet_r3;   /* pet 3 data */
        struct Gpick_pet_u1 {
                    Gdata       data;         /* data */
        } pet_u1;   /* pet -1 data */
        /* etc. */
      } pets;
} Gpick_data;
```

`Gstring_data` string data record

```
typedef struct {
    Gint    in_buf_size;    /* input buffer size (nr. of bytes) */
    Gint    init_pos;       /* initial [cursor] position */
    union Gstring_pets {
          struct Gstring_pet_r1 {
                         Gdata    impl_dep;     /* impl. dep. */
          } pet_r1;      /* pet 1 data */
          struct Gstring_pet_u1 {
                         Gdata    impl_dep;     /* impl. dep. */
          } pet_u1;      /* pet -1 data */
          /* etc. */
      } pets;
} Gstring_data;
```

Gstroke_data stroke data

```
typedef struct {
    Gint     init_pos;        /* initial buffer editing position */
    Gint     in_buf_size;     /* input buffer size (nr. of points) */
    Gfloat   x_interval;      /* X interval */
    Gfloat   y_interval;      /* Y interval */
    Gfloat   time_interval;   /* time interval */
    union Gstroke_pets {
             struct Gstroke_pet_r1 {
                              Gdata              impl_dep;   /* impl. dep. */
             } pet_r1;        /* pet 1 data */
             struct Gstroke_pet_r2 {
                              Gdata              impl_dep;   /* impl. dep. */
             } pet_r2;        /* pet 2 data */
             struct Gstroke_pet_r3 {
                              Gattr_ctrl_flag    attr_ctrl_flag;
                              /* attribute       control flag */
                              Gmarker_attrs      marker_attrs;
                              /* marker          attrs. */
                              Gdata              impl_dep;   /* impl. dep. */
             } pet_r3;        /* pet 3 data */
             struct Gstroke_pet_r4 {
                              Gattr_ctrl_flag    attr_ctrl_flag;
                              /* attribute       control flag */
                              Gline_attrs        line_attrs;
                              /* line            attrs. */
                              Gdata              impl_dep;   /* impl. dep. */
             } pet_r4;        /* pet 4 data */
             struct Gstroke_pet_u1 {
                              Gdata              impl_dep;   /* impl. dep. */
             } pet_u1;        /* pet -1 data */
             /* etc. */
    } pets;
} Gstroke_data;
```

Gval_data valuator data record

```
typedef struct {
    Gfloat   low_value;    /* low value */
    Gfloat   high_value;   /* high value */
    union Gval_pets {
            struct Gval_pet_r1 {
                          Gdata   impl_dep;   /* impl. dep. */
            } pet_r1;     /* pet 1 data */
            struct Gval_pet_r2 {
                          Gdata   impl_dep;   /* impl. dep. */
            } pet_r2;     /* pet 2 data */
            struct Gval_pet_r3 {
                          Gdata   impl_dep;   /* impl. dep. */
            } pet_r3;     /* pet 3 data */
            struct Gval_pet_u1 {
                          Gdata   impl_dep;   /* impl. dep. */
            } pet_u1;     /* pet -1 data */
            /* etc. */
      } pets;
} Gval_data;
```

5.4 Implementation independent type definitions

Gabut_specif ABUT specifier

```
typedef enum {
    GABUT_RIGHT,
    GABUT_LEFT,
    GABUT_TOP,
    GABUT_BOTTOM,
    GABUT_CTR_VERT,
    GABUT_CTR_HOR,
    GABUT_CAP_LINE,
    GABUT_BASE_LINE
} Gabut_specif;
```

Garea_attrs area attributes

```
typedef struct {
    Gasf           int_style_asf;      /* interior style ASF              */
    Gasf           style_ind_asf;      /* style index ASF                 */
    Gasf           colr_ind_asf;       /* colour specifier ASF            */
    Gasf           edge_flag_asf;      /* edge flag ASF                   */
    Gasf           edgetype_asf;       /* edgetype ASF                    */
    Gasf           edgewidth_asf;      /* edgewidth scale factor ASF      */
    Gasf           edge_colr_ind_asf;  /* edge colour specifier ASF       */
    Gint           ind;                /* area index                      */
    Garea_bundle   bundle;             /* area bundle                     */
} Garea_attrs;
```

Garea_bundle area bundle

```
typedef struct {
    Gint_style   int_style;        /* interior style           */
    Gint         style_ind;        /* interior style index     */
    Gcolr_kind   colr_kind;        /* interior colour kind     */
    Gint         colr_ind;         /* interior colour index    */
    Gcolr_rep    int_dir_colr;     /* direct interior colour   */
    Gedge_flag   edge_flag;        /* edge flag                */
    Gint         edgetype;         /* edgetype                 */
    Gfloat       edgewidth;        /* edgewidth scale factor   */
    Gcolr_kind   edge_colr_kind;   /* edge colour kind         */
    Gint         edge_colr_ind;    /* edge colour index        */
    Gcolr_rep    edge_dir_colr;    /* direct edge colour       */
} Garea_bundle;
```

Garea_bundle_table area bundle table

```
typedef struct {
    Gint            num_inds;    /* number of area indices   */
    Gint            *inds;       /* list of area indices     */
    Garea_bundle    *bundles;    /* list of area bundles     */
} Garea_bundle_table;
```

Garea_facs area facilities

```
typedef struct {
    Gint         num_int_styles;    /* number of interior styles            */
    Gint_style   int_styles[5];     /* list of available interior styles    */
    Gint_list    hatch_styles;      /* list of available hatch styles       */
    Gint         num_pred_inds;     /* number of predef. area indices       */
    Gint_list    edgetypes;         /* list of edgetypes                    */
    Gint         num_edgewidths;    /* number of available edgewidths       */
    Gfloat       nom_edgewidth;     /* nominal edgewidth                    */
    Gfloat       min_edgewidth;     /* min. edgewidth                       */
    Gfloat       max_edgewidth;     /* max. edgewidth                       */
} Garea_facs;
```

Gasf attribute source flag

```
typedef enum {
    GASF_BUNDLED,
    GASF_INDIV
} Gasf;
```

Gasfs attribute source flags

```
typedef struct {
    Gasf   linetype;         /* linetype ASF                       */
    Gasf   linewidth;        /* linewidth scale factor ASF         */
    Gasf   line_colr_ind;    /* line colour ASF                    */
    Gasf   marker_type;      /* marker type ASF                    */
    Gasf   marker_size;      /* marker size scale factor ASF       */
    Gasf   marker_colr_ind;  /* marker colour ASF                  */
    Gasf   text_font_prec;   /* text font and precision ASF        */
    Gasf   char_expan;       /* character expansion factor ASF     */
    Gasf   char_space;       /* character spacing ASF              */
    Gasf   text_colr_ind;    /* text colour ASF                    */
    Gasf   int_style;        /* interior style ASF                 */
    Gasf   int_style_ind;    /* interior style index ASF           */
    Gasf   int_colr_ind;     /* interior colour ASF                */
    Gasf   edge_flag;        /* edge flag ASF                      */
    Gasf   edgetype;         /* edgetype ASF                       */
    Gasf   edgewidth;        /* edgewidth scale factor ASF         */
    Gasf   edge_colr_ind;    /* edge colour ASF                    */
} Gasfs;
```

Gattr_ctrl_flag attribute control flag

```
typedef enum {
    GFLAG_CUR,
    GFLAG_SPECIF
} Gattr_ctrl_flag;
```

Gaudit_flag audit flag

```
typedef enum {
    GAUDIT_OFF,
    GAUDIT_ON
} Gaudit_flag;
```

Gaudit_op audit operation

```
typedef struct {
    enum Gaudit_op_type {
            GAUDIT_OPEN, GAUDIT_CLOSE, GAUDIT_BEGIN, GAUDIT_END
    } type;
    void   *specif;   /* audit specifier           */
} Gaudit_op;
```

Gbasic_in_data basic input data

```
typedef struct{
    Gin_class class; /* basic input [measure] class */
    union {
        Gloc_data       loc_data;       /* locator data */
        Gstroke_data    stroke_data;    /* stroke data */
        Gval_data       val_data;       /* valuator data */
        Gchoice_data    choice_data;    /* choice data */
        Gpick_data      pick_data;      /* pick data */
        Gstring_data    string_data;    /* string data */
    } in_data; /* input data */
} Gbasic_in_data;
```

Gbasic_in_value basic input value

```
typedef struct{
    Gin_class class; /* basic input [measure] class */
    union {
        Gloc        loc;        /* locator value */
        Gstroke     stroke;     /* stroke value */
        Gfloat      val;        /* valuator value */
        Gchoice     choice;     /* choice value */
        Gpick       pick;       /* pick value */
        char        *string;    /* string value */
    } in_value; /* input value */
} Gbasic_in_value;
```

Gbndry boundary

```
typedef struct {
    enum Gbndry_type {
        GBNDRY_CLOSED_PATH,
        GBNDRY_ELL_DISC
    } type;
    union {
        Gpath_list      closed_path_seq;    /* closed path sequence    */
        Gconic_matrix   ell_disc;           /* elliptic disc           */
    } value;
} Gbndry;
```

Gbndry_list boundary list

```
typedef struct {
    Gint     num_bndries;    /* number of boundaries in list    */
    Gbndry   *bndries;       /* list of boundaries              */
} Gbndry_list;
```

Gcap cap

```
typedef enum {
    GCAP_BUT,
    GCAP_ROUND,
    GCAP_SQUARE
} Gcap;
```

Gchange_flag change flag

```
typedef enum {
    GFLAG_CHANGED,
    GFLAG_NOT_CHANGED
} Gchange_flag;
```

Gchoice choice [value]

```
typedef struct {
    Gchoice_status   status;    /* status         */
    Gint             num;       /* choice number  */
} Gchoice;
```

Gchoice_status choice status

```
typedef enum {
    GCHOICE_OK,
    GCHOICE_NO_CHOICE
} Gchoice_status;
```

Gcieluv CIE L*u*v* [colour specification]

```
typedef struct {
    Gfloat   u_prime;   /* u' coefficient   */
    Gfloat   v_prime;   /* v' coefficient   */
    Gfloat   y;         /* y luminance      */
} Gcieluv;
```

Gclip_ind clipping indicator

```
typedef enum {
    GIND_NO_CLIP,
    GIND_CLIP
} Gclip_ind;
```

Gclosed_path_op_st closed path operating state

```
typedef enum {
    GCP_ST_CPCL,
    GCP_ST_CPOP
} Gclosed_path_op_st;
```

Gcolr_avail colour available

```
typedef enum {
    GAVAIL_MONOCHR,
    GAVAIL_COLR
} Gcolr_avail;
```

Gcolr_facs colour facilities

```
typedef struct {
    Gint            num_colr_models;    /* number of colour models              */
    Gint            colr_models[4];     /* list of available colour models      */
    Gint            num_colrs;          /* number of colours                     */
    Gcolr_chars     chars;              /* colour characteristics                */
    Gcolr_avail     colr_avail;         /* colour availability                   */
    Gint            num_pred_inds;      /* number of predef. colour indices      */
} Gcolr_facs;
```

Gcolr_kind colour kind

```
typedef enum {
     GCOLR_INDIR,
     GCOLR_RGB,
     GCOLR_CIELUV,
     GCOLR_HLS,
     GCOLR_HSV
} Gcolr_kind;
```

Gcolr_rep colour representation

```
typedef union {
    Grgb       rgb;        /* Red Green Blue colour specification */
    Gcieluv    cieluv;     /* CIE L*u*v* 1976 colour specification */
    Ghls       hls;        /* Hue Luminance Saturation colour specification */
    Ghsv       hsv;        /* Hue Saturation Value colour specification */
                           /* etc. */
} Gcolr_rep;
```

Gcolr_specif colour specifier

```
typedef struct {
    Gcolr_kind    kind;       /* colour kind */
    union {
        Gint      ind;        /* colour index*/
        Grgb      rgb;        /* Red Green Blue colour specification */
        Gcieluv   cieluv;     /* CIE L*u*v* 1976 colour specification */
        Ghls      hls;        /* Hue Luminance Saturation colour specification */
        Ghsv      hsv;        /* Hue Saturation Value colour specification */
    } value;
} Gcolr_specif;
```

Gcolr_table colour table

```
typedef struct {
    Gint            num_inds;        /* number of colour indices        */
    Gint            *inds;           /* list of colour indices          */
    Gcolr_specif    *colr_specifs;   /* list of colour specifications   */
} Gcolr_table;
```

Gcomp_data composite data

```
typedef struct {
    Gint              num_devs;         /* number of [basic] devices */
    Gbasic_in_data    *basic_in_data;   /* list of basic input data */
} Gcomp_data;
```

Gcomp_value composite value

```
typedef struct {
    Gint               num_devs;          /* number of [basic] devices */
    Gbasic_in_value    *basic_in_value;   /* list of basic input values*/
} Gcomp_value;
```

Gconic_matrix conic matrix

```
typedef Gfloat Gconic_matrix[3][3];
```

Gconic_sec conic section

```
typedef struct {
    Gconic_matrix   matrix;       /* conic matrix */
    Gpoint          point_1;      /* point # 1 */
    Gpoint          point_2;      /* point # 2 */
    Gsense_flag     sense_flag;   /* sense flag */
} Gconic_sec;
```

Gconic_sec_list conic primitive list

```
typedef struct {
    Gint          num_conic_prims;   /* number of conic primitives in list   */
    Gconic_sec    *conic_prims;      /* list of conic primitives             */
} Gconic_sec_list;
```

Gcont_attr_name contour attribute name

```
typedef enum {
    GNAME_STYLE,
    GNAME_WIDTH,
    GNAME_CAP,
    GNAME_JOIN
} Gcont_attr_name;
```

`Gcont_attr_value` contour attribute value

```
typedef struct {
    Gcont_attr_name   name;    /* attribute name    */
    union {
                      Gint     style;   /* style     */
                      Gfloat   width;   /* width     */
                      Gcap     cap;     /* cap       */
                      Gjoin    join;    /* join      */
    } value;
} Gcont_attr_value;
```

`Gcont_attrs` contour attributes

```
typedef struct {
    Gint      style;   /* style    */
    Gfloat    width;   /* width    */
    Gcap      cap;     /* cap      */
    Gjoin     join;    /* join     */
} Gcont_attrs;
```

`Gconv_flag` conversion flag

```
typedef enum {
    GCONV_DONE,
    GCONV_APPROX,
    GCONV_NOT_CONV
} Gconv_flag;
```

`Gcoord_switch` coordinate switch

```
typedef enum {
    GCOORD_WC,
    GCOORD_NDC
} Gcoord_switch;
```

`Gctrl_flag` control flag

```
typedef enum {
    GFLAG_COND,
    GFLAG_ALWAYS
} Gctrl_flag;
```

Gctrl_point_list control point list

```
typedef struct {
    enum Grat { /* rationality (RATIONAL|POLYNOMIAL) */
            GNURB_RAT,
            GNURB_POL
    } rat;
    Gint    num_points;   /* number of points in the list */
    union {
            Ghomo_point   *homo_points;   /* list of homg. points  */
            Gpoint        *points;        /* list of points        */
     } ctrl_points;
} Gctrl_point_list;
```

Gdata data

```
typedef struct {
    size_t   size;    /* size of data       */
    void     *data;   /* pointer to data    */
} Gdata;
```

Gdc_units device coordinate units

```
typedef enum {
    GDC_METRES,
    GDC_OTHER
} Gdc_units;
```

Gdesign design

```
typedef struct {
    Gint           stencil_name;     /* stencil name            */
    Gpoint         stencil_origin;   /* stencil origin          */
    Gtran_matrix   stencil_tran;     /* stencil transformation  */
    Gint           tiling_name;      /* tiling name             */
    Gpoint         tiling_origin;    /* tiling origin           */
    Gtran_matrix   tiling_tran;      /* tiling transformation   */
} Gdesign;
```

Gdet detectability

```
typedef enum {
    GSEG_UNDET,
    GSEG_DET
} Gdet;
```

Gdev_comp device composition

```
typedef struct {
    Gdev_id     id;            /* device id     */
    Gint_list   meas_set;      /* measure set   */
    Gint_list   trigger_set;   /* triger set    */
} Gdev_comp;
```

Gdev_comp_list device composition list

```
typedef struct {
    Gint        num_devs;     /* number of [basic] devices in list   */
    Gdev_comp   *dev_comps;   /* list of device compositions         */
} Gdev_comp_list;
```

Gdev_id device identifier

```
typedef struct {
    Gint        ws_id;      /* workstation identifier   */
    Gint        num;        /* device number            */
    Gin_class   in_class;   /* input [measure] class    */
} Gdev_id;
```

Gdev_id_in_value_list device identifier and input value list

```
typedef struct {
    Gint        num_devs;     /* number of [basic] devices in list   */
    Gdev_id     *dev_ids;     /* list of device id values            */
    Gin_value   *in_values;   /* list of device input values         */
} Gdev_id_in_value_list;
```

Gdev_id_list device identifier list

```
typedef struct {
    Gint        num_dev_ids;   /* number of device identifiers in list   */
    Gdev_id     *dev_ids;      /* list of device identifiers             */
} Gdev_id_list;
```

Gdisp_space_size display space size

```
typedef struct {
    Gdc_units     dc_units;      /* device coordinate units               */
    Gfloat_size   size_dc;       /* display space size [in] dc [units]    */
    Gint_size     size_raster;   /* display space size [in]
                                    raster [units]                        */
} Gdisp_space_size;
```

Gecho_switch echo switch

```
typedef enum {
    GSWITCH_NO_ECHO,
    GSWITCH_ECHO
} Gecho_switch;
```

Gedge_flag edge flag

```
typedef enum {
    GEDGE_OFF,
    GEDGE_ON
} Gedge_flag;
```

Gell_disc_list elliptic disc list

```
typedef struct {
    Gint              num_ell_discs;    /* number of elliptic discs in list   */
    Gconic_matrix     *ell_discs;       /* list of elliptic discs             */
} Gell_disc_list;
```

Gfloat_list float list

```
typedef struct {
    Gint    num_floats;     /* number of floats in list   */
    Gint    *floats;        /* list of floats             */
} Gfloat_list;
```

Gfloat_size float size

```
typedef struct {
    Gfloat    size_x;   /* x size   */
    Gfloat    size_y;   /* y size   */
} Gfloat_size;
```

Gfont_ind_map font index mapping

```
typedef struct {
    Gint    ind;            /* font index          */
    Gint    iso9541_name;   /* ISO 9541 font name  */
} Gfont_ind_map;
```

`Gfunc_params` function parameters

```
typedef struct {
    Gint   func_id;
    union {

          struct Gopen_gks {
              char            *err_file;
          } open_gks;
          struct Gsave_gks_sl {
              Ggks_sl_name_list    entry_names;
              Ggks_sl_entry_list   entries;
          } save_gks_sl;
          struct Grestore_gks_sl {
              Ggks_sl_entry_list   entries;
          } restore_gks_sl;
          struct Gescape {
              Gint                func_id;
              Gescape_in_data     in_data;
              Gescape_out_data    out_data;
          } escape;
          struct Groute {
              Groute_dir     dir;
          } route;
          struct Gcreate_out_prim {
              Gprim_params   prim_params;
          } create_out_prim;
          struct Gopen_stencil {
              Gint            stencil_name;
              Ginside_rule    inside_rule;
          } open_stencil;
          struct Grename_stencil {
              Gint            old_name;
              Gint            new_name;
          } rename_stencil;
          struct Gdel_stencil {
              Gint            name;
          } del_stencil;
          struct Gcreate_stencil_bndry {
              Gint            stencil_name;
              Ginside_rule    inside_rule;
              Gbndry_list     bndry;
          } create_stencil_bndry;
          struct Gcreate_stencil_conts {
              Gint            stencil_name;
              Ginside_rule    inside_rule;
              Gpath_list      path_seq;
          } create_stencil_conts;
          struct Gset_cont_attr {
              Gcont_attr_value   cont_attr;
          } set_cont_attr;
```

```
struct Gset_stencil_attr {
    Gint                  stencil_name;
    Gstencil_attr_value   stencil_attr;
} set_stencil_attr;
struct Gget_stencil_attr {
    Gint                  stencil_name;
    Gstencil_attr_name    attr_name;
    Gstencil_attr_value   attr;
} get_stencil_attr;
struct Ginst_stencil {
    Gint           stencil_name;
    Gint           inst_name;
    Gtran_matrix   stencil_tran;
    Ginst_specif   inst_specif;
} inst_stencil;
struct Ginst_stencil_path {
    Gint           stencil_name;
    Gint           inst_name;
    Gpoint         start_map;
    Gpoint         end_map;
    Gint           inst_num;
    Gpath          path;
} inst_stencil_path;
struct Ginst_stencil_seq_path {
    Gseq_specif_list   inst_seq_specif;
    Gpath_specif       path_specif;
    Gtran_matrix       stencil_tran;
} inst_stencil_seq_path;
struct Gopen_tiling {
    Gint           tiling_name;
} open_tiling;
struct Grename_tiling {
    Gint           old_name;
    Gint           new_name;
} rename_tiling;
struct Gdel_tiling {
    Gint           name;
} del_tiling;
struct Gcreate_tiling_comp {
    Gtiling_params   tiling_params;
    Gpoint           origin;
    Grepl_tech       repl_tech;
} create_tiling_comp;
struct Gset_prim_attr {
    Gprim_attr_value   attr_value;
} set_prim_attr;
struct Gadd_set_names_nameset {
    Gnameset         set_names;
} add_set_names_nameset;
struct Gremove_set_names_nameset {
    Gnameset            set_names;
} remove_set_names_nameset;
```

```
struct Gadd_set_scissors_scissor_set {
    Gscissor_list      scissor_set;
} add_set_scissors_scissor_set;
struct Gremove_set_scissors_scissor_set {
    Gint_list            scissor_ids;
} remove_set_scissors_scissor_set;
struct Gset_win_vp {
    Gint             tran_num;
    Gtran            norm_tran;
} set_win_vp;
struct Gset_vp_in_pri {
    Gint             tran_num;
    Gint             ref_tran_num;
    Grel_pri         rel_pri;
} set_vp_in_pri;
struct Gset_scissor_mode {
    Gscissor_mode   scissor_mode;
} set_scissor_mode;
struct Gset_norm_tran_num {
    Gint             tran_num;
} set_norm_tran_num;
struct Gdel_prims_ndc_pic {
    Gsel_crit        sel_crit;
} del_prims_ndc_pic;
struct Gadd_set_names_ndc_pic {
    Gnameset           set_names;
    Gsel_crit          sel_crit;
} add_set_names_ndc_pic;
struct Gremove_set_names_ndc_pic {
    Gnameset             set_names;
    Gsel_crit            sel_crit;
} remove_set_names_ndc_pic;
struct Gset_ndc_pic_prim_attr {
    Gsel_crit             sel_crit;
    Gprim_attr_value   attr_value;
} set_ndc_pic_prim_attr;
struct Gadd_set_scissors_ndc_pic {
    Gscissor_list    scissor_set;
    Gsel_crit        sel_crit;
} add_set_scissors_ndc_pic;
struct Gremove_set_scissors_ndc_pic {
    Gint_list          scissor_ids;
    Gsel_crit          sel_crit;
} remove_set_scissors_ndc_pic;
struct Greorder_ndc_pic {
    Gsel_crit       source_sel_crit;
    Gsel_crit       ref_sel_crit;
    Grel_pos        rel_pos;
} reorder_ndc_pic;
struct Gcopy_ndc_pic_ndc_mf {
    void            *mf_specif;
    Gint            pic_id;
    Gsel_crit       sel_crit;
    Gnameset        set_names;
} copy_ndc_pic_ndc_mf;
struct Gcopy_ndc_mf_pic_ndc_pic {
    void              *mf_specif;
    Gint              pic_id;
    Gnameset          set_names;
} copy_ndc_mf_pic_ndc_pic;
struct Gopen_pic_part {
    Gint            pic_part_name;
} beg_pic_part;
struct Gar_pic_part {
    Gint            pic_part_name;
    void            *ar_specif;
    Gint            ar_name_pic_part;
} ar_pic_part;
struct Gret_pic_part_ar {
    void            *ar_specif;
    Gint            ar_name_pic_part;
    Gint            pic_part_name;
} ret_pic_part_ar;
struct Greopen_pic_part {
    Gint            pic_part_name;
} reopen_pic_part;
struct Gappend_pic_part {
    Gint            source_pic_part_name;
    Gint            sink_pic_part_name;
    Gtran_matrix    global_tran_matrix;
    Gtran_mode      global_tran_mode;
    Gtran_matrix    local_tran_matrix;
    Gtran_mode      local_tran_mode;
    Gnameset        set_names;
} append_pic_part;
struct Grename_pic_part {
    Gint            old_pic_part_name;
    Gint            new_pic_part_name;
} rename_pic_part;
struct Gdel_pic_part {
    Gint            pic_part_name;
} del_pic_part;
struct Gcopy_pic_part_pic_part_store {
    Gint            pic_part_name;
    Gsel_crit       sel_crit;
    Gtran_matrix    global_tran_matrix;
    Gtran_mode      global_tran_mode;
    Gtran_matrix    local_tran_matrix;
    Gtran_mode      local_tran_mode;
    Gnameset        set_names;
    Gscissor_sel    scissor_sel;
} copy_pic_part_pic_part_store;
struct Gcopy_ndc_pic_pic_part_store {
    Gsel_crit         sel_crit;
    Gint                pic_part_name;
    Gnameset            set_names;
} copy_ndc_pic_pic_part_store;
struct Gset_in_dev_mode {
    Gdev_id          dev_id;
    Gop_mode         op_mode;
} set_in_dev_mode;
struct Greq_in {
    Gdev_id          dev_id;
    Gin_status       in_status;
    Gin_value        log_in_value;
} req_in;
struct Gsample_in {
    Gdev_id          dev_id;
    Gin_value        log_in_value;
} sample_in;
struct Gawait_in {
    Gfloat                  timeout;
    Gdev_id_in_value_list   dev_id_values;
} await_in;
struct Gflush_dev_events {
    Gdev_id          dev_id;
} flush_dev_events;
struct Gset_font_ind_map {
    Gfont_ind_map    font_ind_map;
} set_font_ind_map;
struct Gget_glyph_name {
    Gint             font_ind;
    char             character;
    Gint             glyph_name;
} get_glyph_name;
struct Gset_char_code {
    Gint             font_ind;
    char             character;
    Gint             glyph_name;
} set_char_code;
struct Geval_tran_matrix {
    Gpoint           point;
    Gvec             shift;
    Gfloat           angle;
    Gvec             scale;
    Gcoord_switch    coord_switch;
    Gtran_matrix     tran_matrix;
} eval_tran_matrix;
struct Gaccum_tran_matrix {
    Gtran_matrix     matrix;
    Gpoint           point;
    Gvec             shift;
    Gfloat           angle;
    Gvec             scale;
    Gcoord_switch    coord_switch;
    Gtran_matrix     tran_matrix;
```

```
} accum_tran_matrix;
struct Geval_circle {
   Gpoint          ctr;
   Gfloat          radius;
   Gconic_matrix   circle;
} eval_circle;
struct Geval_ell {
   Gpoint          ctr;
   Gpoint          conj_radius_end_point[2];
   Gconic_matrix   ell;
} eval_ell;
struct Geval_circ_arc_3_point {
   Gpoint         circ_point[3];
   Gconic_sec     circ_arc;
} eval_circ_arc_3_point;
struct Geval_circ_arc_ctr {
   Gpoint         ctr;
   Gvec           vec[2];
   Gsense_flag    sense_flag;
   Gfloat         radius;
   Gconic_sec     circ_arc;
} eval_circ_arc_ctr;
struct Geval_ell_arc {
   Gpoint         ctr;
   Gpoint         conj_radius_end_point[2];
   Gvec           vec[2];
   Gconic_sec     conic_arc;
} eval_ell_arc;
struct Geval_hyp_arc {
   Gpoint         ctr;
   Gpoint         tran_radius_end_point;
   Gpoint         conj_radius_end_point;
   Gvec           vec[2];
   Gconic_sec     conic_arc;
} eval_hyp_arc;
struct Geval_par_arc {
   Gpoint         intersec_point;
   Gpoint         start_point;
   Gpoint         end_point;
   Gconic_sec     conic_arc;
} eval_par_arc;
struct Geval_wc_ord {
   Gfloat         wc_ord;
   Gord_sel       ord_sel;
   Gfloat         ndc_value;
} eval_wc_ord;
struct Gopen_ws {
   Gint           ws_id;
   void           *ws_specif_info;
   Gint           ws_gen_type;
} open_ws;
struct Gclose_ws {
    Gint            ws_id;
} close_ws;
struct Gsave_ws_sl {
    Gint                 ws_id;
    Gws_sl_name_list     entry_names;
    Gws_sl_entry_list    entry_list;
} save_ws_sl;
struct Grestore_ws_sl {
    Gint                 ws_id;
    Gws_sl_entry_list    entries;
} restore_ws_sl;
struct Gget_ws_status {
    Gint            ws_id;
    Gws_status      ws_status;
} get_ws_status;
struct Gremove_backdrop {
    Gint             ws_id;
} remove_backdrop;
struct Gset_rep {
    Gint            ws_id;
    Grep            rep_value;
} set_rep;
struct Gset_view_pri {
    Gint            ws_id;
    Gint            view_ind;
    Gint            ref_view_ind;
    Grel_pri        rel_pri;
} set_view_pri;
struct Gset_view_sel_crit {
    Gint            ws_id;
    Gint            view_ind;
    Gsel_crit       sel_crit;
} set_view_sel_crit;
struct Gset_view {
    Gint            ws_id;
    Gint            ind;
    Gtran_matrix    ori_matrix;
    Gtran_matrix    map_matrix;
    Gscissor        scissor;
} set_view;
struct Gset_ws_vis_effects {
    Gint               ws_id;
    Gvis_effects_st    vis_effects_st;
} set_ws_vis_effects;
struct Gset_ws_win_vp {
    Gint            ws_id;
    Gtran           ws_tran;
} set_ws_win_vp;
struct Gdef_in_dev {
    Gdev_id         dev_id;
    Gint_list       meas_set;
    Gint_list       trigger_set;
    Ginit_value    init_value;
} def_in_dev;
struct Ginit_in_dev {
    Gdev_id        dev_id;
    Ginit_value    init_value;
} init_in_dev;
struct Gset_ws_sel_crit {
    Gint            ws_id;
    Gsel            sel;
} set_ws_sel_crit;
struct Gcopy_real_pic_real_mf {
    Gint            ws_id;
    void            *mf_specif;
    Gint            pic_id;
} copy_real_pic_real_mf;
struct Gcopy_blank_real_pic_real_mf {
    Gint                ws_id;
    void                *mf_specif;
    Gint                pic_id;
} copy_blank_real_pic_real_mf;
struct Gcopy_real_mf_pic_backdrop {
    Gint               ws_id;
    void               *mf_specif;
    Gint               pic_id;
} copy_real_mf_pic_backdrop;
struct Gmessage {
    Gint           ws_id;
    char           *message;
} message;
struct Greset_specif_ws_dt_entry_st {
    Gint                ws_id;
    Gws_dt_name         ws_dt_name;
} reset_specif_ws_dt_entry_st;
struct Gget_text_extent {
    Gint           ws_id;
    Gpoint         pos;
    char           *str;
    Gtext_extent   extent;
} get_text_extent;
struct Geval_view_ori_matrix {
    Gpoint         ref_point;
    Gvec           view_up;
    Gtran_matrix   view_ori_matrix;
} eval_view_ori_matrix;
struct Geval_view_map_matrix {
    Gtran          win_vp;
    Gtran_matrix   view_map_matrix;
} eval_view_map_matrix;
struct Gconv_colr {
    Gint           ws_id;
    Gcolr_specif   colr_specif;
    Gint           colr_model;
```

```
            Gcolr_specif   colr_coords;
            Gconv_flag     conv_flag;
        } conv_colr;
        struct Gcreate_seg {
            Gint           seg_name;
        } create_seg;
        struct Grename_seg {
            Gint           old_seg_name;
            Gint           new_seg_name;
        } rename_seg;
        struct Gdel_seg {
            Gint           seg_name;
        } del_seg;
        struct Gdel_seg_ws {
            Gint           ws_id;
            Gint           seg_name;
        } del_seg_ws;
        struct Gassoc_seg_ws {
            Gint           ws_id;
            Gint           seg_name;
        } assoc_seg_ws;
        struct Gcopy_seg_ws {
            Gint           ws_id;
            Gint           seg_name;
        } copy_seg_ws;
        struct Ginsert_seg {
            Gint           seg_name;
            Gtran_matrix   tran_matrix;
        } insert_seg;
        struct Gset_seg_attr {
            Gint              seg_name;
            Gseg_attr_value   seg_attr_value;
        } set_seg_attr;
        struct Gactivate_ws {
            Gint           ws_id;
        } activate_ws;
        struct Gdeactivate_ws {
            Gint           ws_id;
        } deactivate_ws;
        struct Gclear_ws {
            Gint           ws_id;
            Gctrl_flag     ctrl_flag;
        } clear_ws;
        struct Gget_map_seg_name {
            Gint           seg_name;
            Gint           map_seg_name;
        } get_map_seg_name;
        struct Gget_map_ws_id {
            Gint           ws_id;
            Gint           map_ws_id;
        } get_map_ws_id;
    } func_data;
```

```
  } Gfunc_params;
```

REMARK. For each field **xxx** the value of **func_id** is given by the macro **Gfn_xxx**. See subclause 6.1 for numerical values of **Gfn_xxx**.

Ggks_dt_entry GKS description table entry

```
  typedef struct {
      Ggks_dt_name        name;                 /* entry name */
      union {
          Gint_list       avail_ws_gen_types;   /* list of available ws. types     */
          Gint_list       avail_fonts;          /* list of available fonts         */
          Gfont_ind_map   def_font_ind_map;     /* default font index mapping*/
      } value;
  } Ggks_dt_entry;
```

Ggks_dt_name GKS description table name

```
  typedef enum {
      GGKS_DT_AVAIL_WS_GEN_TYPES,    /* list of available ws. types    */
      GGKS_DT_AVAIL_FONTS,           /* list of available fonts        */
      GGKS_DT_DEF_FONT_IND_MAP       /* default font index mapping*/
  } Ggks_dt_name;
```

Ggks_op_st GKS operating state

```
  typedef enum {
      GOP_ST_GKCL,
      GOP_ST_GKOP
    } Ggks_op_st;
```

Ggks_sl_entry GKS state list entry

```
  typedef struct {
      Ggks_sl_name           name;                  /* entry name */
      union {
          Groute_dir         route_dir;             /* route direction                */
          Gscissor_mode      scissor_mode;          /* scissor mode                   */
          Gint_list          open_wss;              /* list of open workstations.     */
          Gint_list          active_wss;            /* list of active workstations    */
          Gopen_audit_list   open_audits;           /* set of open audits             */
          Gint_list          open_playbacks;        /* set of open playbacks          */
          Gin_queue          in_queue;              /* input queue                    */
          Gfont_ind_map      font_ind_map;          /* font index mapping             */
          Gint               pick_id;               /* pick identifier                */
          Gnameset           nameset;               /* nameset                        */
          Gtran_matrix       global_tran_matrix;    /* global
                                                       transformation matrix          */
          Gtran_matrix       local_tran_matrix;     /* local
                                                       transformation matrix          */
          Gfloat_size        pat_size;              /* pattern size                   */
          Gpoint             pat_ref_point;         /* pattern reference point        */
          Gfloat             char_ht;               /* character height               */
```

```
        Gvec               char_up_vec;          /* character up vector           */
        Gfloat             text_skew_angle;      /* text skew angle               */
        Gtext_path         text_path;            /* text path                     */
        Gtext_align        text_align;           /* text alignment                */
        Gscissor_list      scissor_set;          /* scissor set                   */
        Gint               line_ind;             /* line index                    */
        Gint               linetype;             /* linetype                      */
        Gfloat             linewidth;            /* linewidth scale factor        */
        Gcolr_specif       line_colr_specif;     /* line colour specifier         */
        Gint               marker_ind;           /* marker index                  */
        Gint               marker_type;          /* marker type                   */
        Gfloat             marker_size;          /* marker size scale factor      */
        Gcolr_specif       marker_colr_specif;   /* marker colour specifier       */
        Gint               area_ind;             /* area ind                      */
        Gint_style         int_style;            /* interior style                */
        Gint               int_style_ind;        /* interior style index          */
        Gcolr_specif       int_colr_specif;      /* interior colour specifier     */
        Gedge_flag         edge_flag;            /* edge flag                     */
        Gint               edgetype;             /* edgetype                      */
        Gfloat             edgewidth;            /* edgewidth scale factor        */
        Gcolr_specif       edge_colr_specif;     /* edge colour specifier         */
        Gint               text_ind;             /* text index                    */
        Gtext_font_prec    text_font_prec;       /* text font
                                                    and precision                 */
        Gfloat             char_expan;           /* character expansion factor    */
        Gfloat             char_space;           /* character spacing             */
        Gcolr_specif       text_colr_specif;     /* text colour specifier         */
        Gasfs              asfs;                 /* asfs                          */
        Gint               norm_tran_num;        /* current norm. tran.
                              number             */
        Gtran              norm_tran[64];        /* list of norm. tran.           */
        Gint               vp_in_pri[64];        /* list of vp input pris         */
        Gint               name_open_pic_part;   /* name of open pic. part        */
        Gint_list          pic_part_names;       /* list of picture parts
                              in use             */
        Gint               name_open_seg;        /* name of open segment          */
        Gint_list          seg_names;            /* list of segments
                              in use             */
        Gseg_sl_list       seg_sls;              /* list of segment
                                                    state lists                   */
        Gint               name_open_stencil;    /* name of open stencil          */
        Gint_list          stencil_names;        /* list of stencils
                              in use             */
        Gstencil_sl_list   stencil_sls;          /* list of stencil
                                                    state lists in use            */
        Gcont_attrs        cont_attrs;           /* contour attributes            */
        Gint               name_open_tiling;     /* name of open tiling           */
        Gint_list          tiling_names;         /* list of tilings in use        */

    } value;
} Ggks_sl_entry;
```

Ggks_sl_entry_list GKS state list entry list

```
typedef struct {
    Gint            num_gks_sl_entries;   /* number of entries in list    */
    Ggks_sl_entry   *gks_sl_entries;      /* list of entries              */
} Ggks_sl_entry_list;
```

Ggks_sl_name GKS state list name

```
  typedef enum {
      GGKS_SL_ROUTE_DIR,           /* route direction                    */
      GGKS_SL_SCISSOR_MODE,        /* scissor mode                       */
      GGKS_SL_OPEN_WSS,            /* list of open workstations.         */
      GGKS_SL_ACTIVE_WSS,          /* list of active workstations        */
      GGKS_SL_OPEN_AUDITS,         /* set of open audits                 */
      GGKS_SL_OPEN_PLAYBACKS,      /* set of open playbacks              */
      GGKS_SL_IN_QUEUE,            /* input queue                        */
      GGKS_SL_FONT_IND_MAP,        /* font index mapping                 */
      GGKS_SL_PICK_ID,             /* pick identifier                    */
      GGKS_SL_NAMESET,             /* nameset                            */
      GGKS_SL_GLOBAL_TRAN_MATRIX,  /* global
                                     transformation matrix               */
      GGKS_SL_LOCAL_TRAN_MATRIX,   /* local
                                     transformation matrix               */
      GGKS_SL_PAT_SIZE,            /* pattern size                       */
      GGKS_SL_PAT_REF_POINT,       /* Pattern reference point            */
      GGKS_SL_CHAR_HT,             /* Character height                   */
      GGKS_SL_CHAR_UP_VEC,         /* character up vector                */
      GGKS_SL_TEXT_SKEW_ANGLE,     /* text skew angle                    */
      GGKS_SL_TEXT_PATH,           /* text path                          */
      GGKS_SL_TEXT_ALIGN,          /* text alignment                     */
      GGKS_SL_SCISSOR_SET,         /* scissor set                        */
      GGKS_SL_LINE_IND,            /* line index                         */
      GGKS_SL_LINETYPE,            /* linetype                           */
      GGKS_SL_LINEWIDTH,           /* linewidth scale factor             */
      GGKS_SL_LINE_COLR_SPECIF,    /* line colour specifier              */
      GGKS_SL_MARKER_IND,          /* marker index                       */
      GGKS_SL_MARKER_TYPE,         /* marker type                        */
      GGKS_SL_MARKER_SIZE,         /* marker size scale factor           */
      GGKS_SL_MARKER_COLR_SPECIF,  /* marker colour specifier            */
      GGKS_SL_AREA_IND,            /* area ind                           */
      GGKS_SL_INT_STYLE,           /* interior style                     */
      GGKS_SL_INT_STYLE_IND,       /* interior style index               */
      GGKS_SL_INT_COLR_SPECIF,     /* interior colour specifier          */
      GGKS_SL_EDGE_FLAG,           /* edge flag                          */
      GGKS_SL_EDGETYPE,            /* edgetype                           */
      GGKS_SL_EDGEWIDTH,           /* edgewidth scale factor             */
      GGKS_SL_EDGE_COLR_SPECIF,    /* edge colour specifier              */
      GGKS_SL_TEXT_IND,            /* text index                         */
      GGKS_SL_TEXT_FONT_PREC,      /* text font
                                     and precision                       */
      GGKS_SL_CHAR_EXPAN,          /* character expansion factor         */
      GGKS_SL_CHAR_SPACE,          /* character spacing                  */
      GGKS_SL_TEXT_COLR_SPECIF,    /* text colour specifier              */
      GGKS_SL_ASFS,                /* asfs                               */
      GGKS_SL_NORM_TRAN_NUM,       /* current norm. tran. number         */
      GGKS_SL_NORM_TRANS,          /* list of norm. tran.                */
      GGKS_SL_VP_IN_PRI,           /* list of vp input pris              */
      GGKS_SL_NAME_OPEN_PIC_PART,  /* name of open pic. part             */
```

```
    GGKS_SL_PIC_PART_NAMES,      /* list of picture part names in use   */
    GGKS_SL_CONT_ATTRS,          /* contour attributes                  */
    GGKS_SL_NAME_OPEN_SEG,       /* nme of open segment                 */
    GGKS_SL_SEG_NAMES,           /* list of segments in use             */
    GGKS_SL_SEG_SLS,             /* list of segment
                                    state lists                         */
    GGKS_SL_NAME_OPEN_STENCIL,   /* name of open stencil                */
    GGKS_SL_STENCIL_NAMES,       /* list of stencils in use             */
    GGKS_SL_STENCIL_SLS,         /* list of stencil
                                    state lists in use                  */
    GGKS_SL_NAME_OPEN_TILING,    /* name of open tiling                 */
    GGKS_SL_TILING_NAMES         /* list of tilings in use              */
} Ggks_sl_name;
```

Ggks_sl_name_list GKS state list name list

```
typedef struct {
    Gint           num_gks_sl_names;    /* number of names in list    */
    Ggks_sl_name   *gks_sl_names;       /* list of names              */
} Ggks_sl_name_list;
```

Ghighl highlighting

```
typedef enum {
    GSEG_NORM,
    GSEG_HIGHL
} Ghighl;
```

Ghls Hue Luminance Saturation [colour specification]

```
typedef struct {
    Gfloat   hue;         /* hue          */
    Gfloat   lightness;   /* lightness    */
    Gfloat   satur;       /* saturation   */
} Ghls;
```

Ghomo_point homogeneous point

```
typedef struct {
    Gfloat   xw;   /* x coordinate   */
    Gfloat   yw;   /* y coordinate   */
    Gfloat   w;    /* w coordinate   */
} Ghomo_point;
```

Ghsv Hue Saturation Value [colour specification]

```
typedef struct {
    Gfloat   hue;     /* hue          */
    Gfloat   satur;   /* saturation   */
    Gfloat   value;   /* value        */
} Ghsv;
```

Gin_class input [measure] class

```
typedef enum {
    GIN_NONE,
    GIN_LOC,
    GIN_STROKE,
    GIN_VAL,
    GIN_CHOICE,
    GIN_PICK,
    GIN_STRING,
    GIN_COMP
} Gin_class;
```

Gin_data input data

```
typedef union {
    Gloc_data      loc_data;      /* locator data    */
    Gstroke_data   stroke_data;   /* strokedata      */
    Gval_data      val_data;      /* valuator data   */
    Gchoice_data   choice_data;   /* choice data     */
    Gpick_data     pick_data;     /* pick data       */
    Gstring_data   string_data;   /* string data     */
    Gcomp_data     comp_data;     /* composite data  */
} Gin_data;
```

Gin_dev_init_value input device initial value

```
typedef struct {
    Gdev_id       dev_id;      /* device identifier   */
    Ginit_value   init_value;  /* initial value       */
} Gin_dev_init_value;
```

Gin_dev_init_value_list input initial value list

```
typedef struct {
    Gint                  num_devs;       /* number of input devices       */
    Gin_dev_init_value    *init_values;   /* list of input initial values  */
} Gin_dev_init_value_list;
```

Gin_queue input queue

```
typedef struct {
    Gint             num_items;   /* number of input items   */
    Gin_queue_item   *in_items;   /* input items             */
} Gin_queue;
```

`Gin_queue_item` input item [in queue]

```
typedef struct {
    Gdev_id     dev_id;     /* device identifier   */
    Gin_value   in_value;   /* input value         */
} Gin_queue_item;
```

`Gin_status` [input] status

```
typedef enum {
    GIN_STATUS_NONE,
    GIN_STATUS_OK ,
    GIN_STATUS_NO_IN
} Gin_status;
```

`Gin_value` input value

```
typedef union {
    Gloc          loc;        /* locator value    */
    Gstroke       stroke;     /* strokevalue      */
    Gfloat        val;        /* valuator value   */
    Gchoice       choice;     /* choice value     */
    Gpick         pick;       /* pick value       */
    char          *string;    /* string value     */
    Gcomp_value   comp;       /* composite value  */
} Gin_value;
```

`Ginit_in` initial input

```
typedef struct {
    Gin_value      in_value;       /* input value               */
    Gint           pet;            /* prompt and echo type      */
    Gecho_switch   echo_switch;    /* echo switch               */
    Glimit         echo_area;      /* echo area                 */
    Gin_data       in_data;        /* input data                */
} Ginit_in;
```

`Ginit_sel` initial selector

```
typedef enum {
    GSEL_INIT_VALUE,      /* initial value          */
    GSEL_PET,             /* prompt and echo type   */
    GSEL_ECHO_SWITCH,     /* echo switch            */
    GSEL_ECHO_AREA,       /* echo area              */
    GSEL_IN_DATA,         /* input data             */
    GSEL_ALL              /* all initial input      */
} Ginit_sel;
```

Ginit_value initial value

```
typedef struct {
    Ginit_sel sel; /* initial selector */
    union {
        Gin_value      init_value;    /* input value            */
        Gint           pet;           /* prompt and echo type   */
        Gecho_switch   echo_switch;   /* echo switch            */
        Glimit         echo_area;     /* echo area              */
        Gin_data       in_data;       /* input data             */
        Ginit_in       all;           /* all initial input      */
    } value;
} Ginit_value;
```

Ginq_type inquire type

```
typedef enum {
    GINQ_SET,
    GINQ_REALIZED
} Ginq_type;
```

Ginside_rule inside rule

```
typedef enum {
    GRULE_EVEN_ODD,
    GRULE_WINDING
} Ginside_rule;
```

`Ginst_specif` instance specifier

```
typedef struct {
    enum Ginst_type {/*  instance type */
        GINST_PUT,
        GINST_ALIGN,
        GINST_MAP
    } type;
    union {
        struct Gput_inst {
            Gpoint   ref_point_inst;              /* ref. point on instance */
            Gpoint   ref_point_open_stencil;      /* ref. point on
                                                     open stencil */
        } put;
        struct Galign_inst{
            Gpoint   ref_point_inst;              /* ref. point on instance */
            Gpoint   dir_point_inst;              /* direction point on instance */
            Gpoint   ref_point_open_stencil;      /* ref. point on
                                                     open stencil */
            Gpoint   dir_point_open_stencil;      /* direction point on
                                                     open stencil */
        } align;
        struct Gmap_inst {
            Gpoint   ref_point_inst[3];           /* 3 ref. points on instance */
            Gpoint   corr_point_open_stencil[3];  /* 3 corr. points on
                                                     open stencil */
        } map;
    } specif;
} Ginst_specif;
```

`Gint_list` integer list

```
typedef struct {
    Gint   num_ints;   /* number of integers in list   */
    Gint   *ints;      /* list of integers             */
} Gint_list;
```

`Gint_size` integer size

```
typedef struct {
    Gint   size_x;   /* x size   */
    Gint   size_y;   /* y size   */
} Gint_size;
```

Gint_style interior style

```
typedef enum {
    GSTYLE_HOLLOW,
    GSTYLE_SOLID,
    GSTYLE_PAT,
    GSTYLE_HATCH,
    GSTYLE_EMPTY
} Gint_style;
```

Gjoin join

```
typedef struct {
    enum Gjoin_name {/* name (ROUND|BEVEL|MITRED) */
            GJOIN_ROUND,
            GJOIN_BEVEL,
            GJOIN_MITRED
     } name;
    Gfloat   mitred;          /* mitred value */
} Gjoin;
```

Glimit limit

```
typedef struct {
    Gfloat   x_min;   /* x min   */
    Gfloat   x_max;   /* x max   */
    Gfloat   y_min;   /* y min   */
    Gfloat   y_max;   /* y max   */
} Glimit;
```

Glimit_list limit list

```
typedef struct {
    Gint     num_limits;   /* number of limits in list   */
    Glimit   *limits;      /* list of limits             */
} Glimit_list;
```

Gline_attrs line attributes

```
typedef struct {
    Gasf           type_asf;      /* linetype ASF                  */
    Gasf           width_asf;     /* linewidth scale factor ASF    */
    Gasf           colr_ind_asf;  /* line colour specifier ASF     */
    Gint           ind;           /* line index                    */
    Gline_bundle   bundle;        /* line bundle                   */
} Gline_attrs;
```

`Gline_bundle` line bundle

```
typedef struct {
    Gint         type;       /* linetype                 */
    Gfloat       width;      /* linewidth scale factor   */
    Gcolr_kind   colr_kind;  /* colour kind              */
    Gint         colr_ind;   /* colour index             */
    Gcolr_rep    dir_colr;   /* direct colour            */
} Gline_bundle;
```

`Gline_bundle_table` line bundle table

```
typedef struct {
    Gint            num_inds;   /* number of line indices  */
    Gint            *inds;      /* list of line indices    */
    Gline_bundle    *bundles;   /* list of line bundles    */
} Gline_bundle_table;
```

`Gline_facs` line facilities

```
typedef struct {
    Gint_list   types;          /* list of linetypes                */
    Gint        num_widths;     /* number of available linewidths   */
    Gfloat      nom_width;      /* nominal linewidth                */
    Gfloat      min_width;      /* min. linewidth                   */
    Gfloat      max_width;      /* max. linewidth                   */
    Gint        num_pred_inds;  /* number of predef. line indices   */
} Gline_facs;
```

`Gline_fill_ctrl_flag` line/fill control flag

```
typedef enum {
    GFLAG_LINE,
    GFLAG_FILL
} Gline_fill_ctrl_flag;
```

`Gloc` locator value

```
typedef struct {
    Gpoint   point;          /* loc point            */
    Gint     norm_tran_num;  /* norm. tran. number   */
} Gloc;
```

Gmarker_attrs marker attributes

```
typedef struct {
    Gasf            type_asf;       /* marker type ASF                   */
    Gasf            size_asf;       /* marker size scale factor ASF      */
    Gasf            colr_ind_asf;   /* marker colour specifier ASF       */
    Gint            ind;            /* marker index                      */
    Gmarker_bundle  bundle;         /* marker bundle                     */
} Gmarker_attrs;
```

Gmarker_bundle marker bundle

```
typedef struct {
    Gint          type;         /* marker type              */
    Gfloat        size;         /* marker size scale factor */
    Gcolr_kind    colr_kind;    /* colour kind              */
    Gint          colr_ind;     /* colour index             */
    Gcolr_rep     dir_colr;     /* direct colour            */
} Gmarker_bundle;
```

Gmarker_bundle_table marker bundle table

```
typedef struct {
    Gint            num_inds;     /* number of marker indices   */
    Gint            *inds;        /* list of marker indices     */
    Gmarker_bundle  *bundles;     /* list of marker bundles     */
} Gmarker_bundle_table;
```

Gmarker_facs marker facilities

```
typedef struct {
    Gint_list   types;          /* list of marker types                 */
    Gint        num_sizes;      /* number of available marker sizes     */
    Gfloat      nom_size;       /* nominal marker size                  */
    Gfloat      min_size;       /* min. marker size                     */
    Gfloat      max_size;       /* max. marker size                     */
    Gint        num_pred_inds;  /* number of predef. marker indices     */
} Gmarker_facs;
```

Gnameset nameset

```
typedef struct {
    Gint_list   ws_ids;      /* list of ws idenfiers    */
    Gint_list   seg_names;   /* list of segment names   */
    Gint_list   names;       /* list of names           */
} Gnameset;
```

Gnum_in number [of] input [devices]

```
typedef struct {
    Gint    loc;      /* number of locator devices    */
    Gint    stroke;   /* number of stroke devices     */
    Gint    val;      /* number of valuator devices   */
    Gint    choice;   /* number of choice devices     */
    Gint    pick;     /* number of pick devices       */
    Gint    string;   /* number of string devices     */
    Gint    comp;     /* number of composed devices   */
} Gnum_in;
```

Gnurb NURB

```
typedef struct {
    Gint                order;              /* order */
    Gctrl_point_list    ctrl_point_list;    /* list of control points */
    Gfloat_list         weights;            /* list of weight factors */
    Gfloat_list         knots;              /* list of knots */
    Gfloat              t_min;              /* t_min */
    Gfloat              t_max;              /* t_max */
} Gnurb;
```

Gnurb_list NURB list

```
typedef struct {
    Gint     num_nurbs;    /* number of NURBs in list     */
    Gnurb    *nurbs;       /* list of NURBs               */
} Gnurb_list;
```

Gop_mode operating mode

```
typedef enum {
    GOP_REQ,
    GOP_SAMPLE,
    GOP_EVENT
} Gop_mode;
```

Gop_modes_init_values operating modes and initial values

```
typedef struct {
    Gint        num_in_devs;    /* number of input devices     */
    Gdev_id     dev_ids;        /* device identifiers          */
    Gop_mode    op_modes;       /* operating modes             */
    Ginit_in    init_ins;       /* initial input values        */
} Gop_modes_init_values;
```

Gop_st_entry operating state entry

```
typedef struct {
    Gop_st_name           name;          /* GKS operating state name */
    union {
        Ggks_op_st        gks;           /* GKS operating state          */
        Gpic_part_op_st   pic_part;      /* picture part operating state */
        Gstencil_op_st    stencil;       /* stencil operating state      */
        Gseg_op_st        seg;           /* segment operating state      */
        Gclosed_path_op_st closed_path;  /* closed path operating
                                            state                        */
        Gtiling_op_st     tiling;        /* tiling operating state       */
    } value;
} Gop_st_entry;
```

Gop_st_name operating state entry name

```
typedef enum {
    GOP_ST_GKS,             /* GKS operating state          */
    GOP_ST_PIC_PART,        /* picture part operating state */
    GOP_ST_STENCIL,         /* stencil operating state      */
    GOP_ST_SEG,             /* segment operating state      */
    GOP_ST_CLOSED_PATH,     /* closed path operating state  */
    GOP_ST_TILING           /* tiling operating state       */
} Gop_st_name;
```

Gopen_audit_list open audit list

```
typedef struct {
    Gint          num_audits;     /* number of audits          */
    Gint          *audit_ids;     /* list of audit identifiers */
    Gaudit_flag   *audit_flags;   /* list of audit flags       */
} Gopen_audit_list;
```

Gord_sel ordinate selector

```
typedef enum {
    GORD_X,
    GORD_Y
} Gord_sel;
```

Gpat_rep pattern representation

```
typedef struct {
    Gint_size    dims;           /* colour array's dimensions   */
    Gcolr_kind   colr_kind;      /* colour kind                 */
    Gint         *colr_array;    /* colour array                */
    union {
        Grgb     *rgb;           /* RGB colours                 */
        Gcieluv  *cieluv;        /* CIELUV colours              */
        Ghls     *hls;           /* HLS colours                 */
        Ghsv     *hsv;           /* HSV colours                 */
    } dir_colr;
} Gpat_rep;
```

Gpat_table pattern table

```
typedef struct {
    Gint        num_inds;     /* number of pattern indices         */
    Gint        *inds;        /* list of pattern indices           */
    Gpat_rep    *pat_reps;    /* list of pattern representations  */
} Gpat_table;
```

Gpath path

```
typedef struct {
     enum Gpath_type { /* path type */
        GPATH_LINE,
        GPATH_NURB,
        GPATH_CONIC_SEC
    } type;
    union {
        Gpoint_list   polyline;    /* polyline */
        Gnurb         nurb;        /* NURB*/
        Gconic_sec    conic_sec;   /* conic section */
    } value;
} Gpath;
```

Gpath_list path list

```
typedef struct {
    Gint    num_paths;    /* number of paths in list   */
    Gpath   *paths;       /* list of paths             */
} Gpath_list;
```

Gpath_specif path specifier

```
typedef struct {
    Gabut_specif   abut_specif;   /* Abut specifier      */
    Gpath          path;          /* path                */
    Gpoint         ref_point;     /* reference point     */
} Gpath_specif;
```

Gpic_part_op_st picture part operating state

```
typedef enum {
    GOP_ST_PPCL,
    GOP_ST_PPOP
  } Gpic_part_op_st;
```

Gpick pick [value]

```
typedef struct {
    Gpick_status   status;       /* status          */
    Gnameset       set_names;    /* set of names    */
    Gint           seg_name;     /* segment name    */
    Gint           pick_id;      /* pick identifier */
} Gpick;
```

Gpick_status pick status

```
typedef enum {
    GPICK_OK,
    GPICK_NO_PICK
} Gpick_status;
```

Gplayback_op playback operation

```
typedef struct {
    enum Gplayback_op_type {
           GPLAYBACK_OPEN, GPLAYBACK_CLOSE
    } type;
    void    *specif;   /* audit specifier   */
} Gplayback_op;
```

Gpoint point

```
typedef struct {
    Gfloat   x;   /* x coordinate   */
    Gfloat   y;   /* y coordinate   */
} Gpoint;
```

`Gpoint_list` point list

```
typedef struct {
    Gint     num_points;    /* number of points in the list   */
    Gpoint   *points;       /* list of points                  */
} Gpoint_list;
```

`Gpoint_list_list` list of point lists

```
typedef struct {
    Gint          num_point_lists;    /* number of point lists in the list   */
    Gpoint_list   *point_lists;       /* list of point lists                  */
} Gpoint_list_list;
```

`Gpr_flag` prompt flag

```
typedef enum {
    GPR_OFF,
    GPR_ON
} Gpr_flag;
```

Gprim_attr_name primitive attribute name

```
typedef enum {
    GNAME_PICK_ID,              /* pick identifier                  */
    GNAME_NAMESET,              /* nameset                          */
    GNAME_SCISSOR_SET,          /* scissor set                      */
    GNAME_GLOBAL_TRAN_MATRIX,   /* global transformation matrix     */
    GNAME_LOCAL_TRAN_MATRIX,    /* local transformation matrix      */
    GNAME_PAT_SIZE,             /* pattern size                     */
    GNAME_PAT_REF_POINT,        /* pattern reference point          */
    GNAME_CHAR_HT,              /* character height                 */
    GNAME_CHAR_UP_VEC,          /* character up vector              */
    GNAME_TEXT_SKEW_ANGLE,      /* text skew angle                  */
    GNAME_TEXT_PATH,            /* text path                        */
    GNAME_TEXT_ALIGN,           /* text alignment                   */
    GNAME_LINE_IND,             /* line index                       */
    GNAME_LINETYPE,             /* linetype                         */
    GNAME_LINEWIDTH,            /* linewidth scale factor           */
    GNAME_LINE_COLR_SPECIF,     /* line colour specifier            */
    GNAME_MARKER_IND,           /* marker index                     */
    GNAME_MARKER_TYPE,          /* marker type                      */
    GNAME_MARKER_SIZE,          /* marker size scale factor         */
    GNAME_MARKER_COLR_SPECIF,   /* marker colour specifier          */
    GNAME_AREA_IND,             /* area index                       */
    GNAME_INT_STYLE,            /* interior style                   */
    GNAME_INT_STYLE_IND,        /* interior style index             */
    GNAME_INT_COLR_SPECIF,      /* interior colour specifier        */
    GNAME_EDGE_FLAG,            /* edge flag                        */
    GNAME_EDGETYPE,             /* edgetype                         */
    GNAME_EDGEWIDTH,            /* edgewidth scale factor           */
    GNAME_EDGE_COLR_SPECIF,     /* edge colour specifier            */
    GNAME_TEXT_IND,             /* text index                       */
    GNAME_TEXT_FONT_PREC,       /* text font and precision          */
    GNAME_CHAR_EXPAN,           /* character expansion factor       */
    GNAME_CHAR_SPACE,           /* character spacing                */
    GNAME_TEXT_COLR_SPECIF,     /* text colour specifier            */
    GNAME_ASFS                  /* asfs                             */
} Gprim_attr_name;
```

Gprim_attr_value primitive attribute value

```
typedef struct {
    Gprim_attr_name name; /* primitive attribute name */
    union {
        Gint                pick_id;                /* pick identifier              */
        Gnameset            nameset;                /* nameset                      */
        Gscissor_list       scissor_set;            /* scissor set                  */
        Gtran_matrix        global_tran_matrix;     /* global
                                                       transformation matrix        */
        Gtran_matrix        local_tran_matrix;      /* local
                                                       transformation matrix        */
        Gfloat_size         pat_size;               /* pattern size                 */
        Gpoint              pat_ref_point;          /* pattern
                                                       reference point              */
        Gfloat              char_ht;                /* character height             */
        Gvec                char_up_vec;            /* character up vector          */
        Gfloat              text_skew_angle;        /* text skew angle              */
        Gtext_path          text_path;              /* text path                    */
        Gtext_align         text_align;             /* text alignment               */
        Gint                line_ind;               /* line index                   */
        Gint                linetype;               /* linetype                     */
        Gfloat              linewidth;              /* linewidth
                                                       scale factor                 */
        Gcolr_specif        line_colr_specif;       /* line colour
                                                       specifier                    */
        Gint                marker_ind;             /* marker index                 */
        Gint                marker_type;            /* marker type                  */
        Gfloat              marker_size;            /* marker size
                                                       scale factor                 */
        Gcolr_specif        marker_colr_specif;     /* marker colour
                                                       specifier                    */
        Gint                area_ind;               /* area ind                     */
        Gint_style          int_style;              /* interior style               */
        Gint                int_style_ind;          /* interior style index         */
        Gcolr_specif        int_colr_specif;        /* interior colour
                                                       specifier                    */
        Gedge_flag          edge_flag;              /* edge flag                    */
        Gint                edgetype;               /* edgetype                     */
        Gfloat              edgewidth;              /* edgewidth
                                                       scale factor                 */
        Gcolr_specif        edge_colr_specif;       /* edge colour
                                                       specifier                    */
        Gint                text_ind;               /* text index                   */
        Gtext_font_prec     text_font_prec;         /* text font
                                                       and precision                */
        Gfloat              char_expan;             /* character
                                                       expansion factor             */
        Gfloat              char_space;             /* character spacing            */
        Gcolr_specif        text_colr_specif;       /* text colour
                                                       specifier                    */
        Gasfs               asfs;                   /* asfs                         */
```

```
     }value;
 } Gprim_attr_value;
```

`Gprim_params` primitive parameters

```
 typedef struct {
     Gprim_type type; /* primitive type */
     union {
         Gpoint_list_list    polyline_set;       /* polyline set*/
         Gnurb_list          nurb_set;           /* NURB set*/
         Gconic_sec_list     conic_sec_set;      /* conic section set*/

         Gpoint_list         polymarker;         /* polymarker */

         Gpoint_list_list    fill_area_set;      /* fill area set*/
         Gnurb_list          closed_nurb_set;    /* closed NURB set*/
         Gconic_sec_list     ell_sec_set;        /* elliptic sector set*/
         Gconic_sec_list     ell_seg_set;        /* elliptic segment set*/
         Gell_disc_list      ell_disc_set;       /* elliptic disc set*/

         struct Gtext {
             char            *string;            /* character string */
             Gpoint          pos;                /* text position */
         } text;

         struct Gcell_array {
             Grect           rect;               /* cell rectangle */
             Gpat_rep        array;              /* cell array */
         } cell_array;

         Gdesign             design;             /* design */

         struct Ggdp {
             Gpoint_list     point_list;         /* point_list */
             Gint            id;                 /* gdp identifier */
             Ggdp_data       gdp_data;           /* gdp_data */
         } gdp;
     } value;
 } Gprim_params;
```

Gprim_type primtive type

```
typedef enum {
    GPRIM_LINE_SET,
    GPRIM_NURB_SET,
    GPRIM_CONIC_SEC_SET,
    GPRIM_MARKER,
    GPRIM_FILL_SET,
    GPRIM_CLOSED_NURB_SET,
    GPRIM_ELL_SEC_SET,
    GPRIM_ELL_SEG_SET,
    GPRIM_ELL_DISC_SET,
    GPRIM_TEXT,
    GPRIM_CELL_ARRAY,
    GPRIM_DESIGN,
    GPRIM_GDP
} Gprim_type;
```

Gproc_op process operation

```
typedef enum {
    GPROC_SKIP,
    GPROC_DO
} Gproc_op;
```

Grect rectangle

```
typedef struct {
    Gpoint   p;   /* point p  */
    Gpoint   q;   /* point q  */
} Grect;
```

Grel_pos relative position

```
typedef enum {
    GPOS_FRONT,
    GPOS_BACK
} Grel_pos;
```

Grel_pri relative priority

```
typedef enum {
    GPRI_HIGHER,
    GPRI_LOWER
} Grel_pri;
```

Grep representation

```
typedef struct {
    Gint ind; /* index */
    enum Grep_design { /* representation designation */
        GREP_LINE,       /* line bundle                         */
        GREP_MARKER,     /* marker bundle                       */
        GREP_AREA,       /* area bundle                         */
        GREP_PAT,        /* pattern representation              */
        GREP_TEXT,       /* text bundle                         */
        GREP_COLR,       /* colr specification                  */
        GREP_UNDEF       /* undefined                           */
     } design;
    union {
        Gline_bundle     line_bundle;     /* line bundle            */
        Gmarker_bundle   marker_bundle;   /* marker bundle          */
        Gtext_bundle     text_bundle;     /* text bundle            */
        Garea_bundle     area_bundle;     /* area bundle            */
        Gpat_rep         pat_rep;         /* pattern representation */
        Gcolr_specif     colr_specif;     /* colour specification   */
    } value;
} Grep;
```

Grepl_tech replication technique

```
typedef struct {
    enum Grepl_dir {
        GREPL_DX,
        GREPL_DY,
        GREPL_DXY,
        GREPL_DYX
    } dir;
    union {
        Gfloat   dx;    /* dx     */
        Gfloat   dy;    /* dy     */
        Gvec     dxy;   /* dxy    */
        Gvec     dyx;   /* dyx    */
    } value;
} Grepl_tech;
```

Grgb Red Green Blue [colour specification]

```
typedef struct {
    Gfloat   red;     /* red intensity     */
    Gfloat   green;   /* green intensity   */
    Gfloat   blue;    /* blue intensity    */
} Grgb;
```

Groute_dir direction

```
typedef enum {
    GDIR_NDC,
    GDIR_BACKDROP
} Groute_dir;
```

Gscissor scissor

```
typedef struct {
    Gclip_ind     clip_ind;          /* clipping indicator       */
    Glimit_list   clip_rect_set;     /* clipping rectangle set   */
    Gshield_ind   shield_ind;        /* shielding indicator      */
    Glimit_list   shield_rect_set;   /* shielding rectangle set  */
} Gscissor;
```

Gscissor_list scissor list

```
typedef struct {
    Gint       num_scissors;    /* number of scissors in list         */
    Gint       *scissor_ids;    /* list of scissor identifications    */
    Gscissor   *scissors;       /* list of scissors                   */
} Gscissor_list;
```

Gscissor_mode scissor mode

```
typedef enum {
    GSCISSOR_LOCUS,
    GSCISSOR_SHAPE
} Gscissor_mode;
```

Gscissor_sel scissor select

```
typedef enum {
    GSCISSOR_CHANGE,
    GSCISSOR_LEAVE
} Gscissor_sel;
```

Gseg_attr_name segment attribute name

```
typedef enum {
    GNAME_TRAN_MATRIX,    /* transformation matrix   */
    GNAME_VIS,            /* visibility              */
    GNAME_HIGHL,          /* hightlighting           */
    GNAME_PRI,            /* segment priority        */
    GNAME_DET             /* detectability           */
} Gseg_attr_name;
```

Gseg_attr_value segment attribute value

```
typedef struct {
    Gseg_attr_name name; /* segment attribute name */
    union {
        Gtran_matrix   tran_matrix;   /* transformation matrix   */
        Gvis           vis;           /* visibility              */
        Ghighl         highl;         /* hightlighting           */
        Gfloat         pri;           /* segment priority        */
        Gdet           det;           /* detectability           */
    } value;
} Gseg_attr_value;
```

Gseg_attrs segment attributes

```
typedef struct {
    Gtran_matrix   tran_matrix;   /* transformation matrix   */
    Gvis           vis;           /* visibility              */
    Ghighl         highl;         /* hightlighting           */
    Gfloat         pri;           /* segment priority        */
    Gdet           det;           /* detectability           */
} Gseg_attrs;
```

Gseg_op_st segment operating state

```
typedef enum {
    GOP_ST_SGCL,
    GOP_ST_SGOP
  } Gseg_op_st;
```

Gseg_sl segment state list

```
typedef struct {
    Gint           name;         /* segment name             */
    Gint_list      assoc_wss;    /* associated workstations  */
    Gseg_attrs     attrs;        /* segment attributes       */
} Gseg_sl;
```

Gseg_sl_list list of segment state lists

```
typedef struct {
    Gint      num_seg_sls;    /* number of segment state lists in list   */
    Gseg_sl   *seg_sls;       /* list of segment state lists             */
} Gseg_sl_list;
```

Gsel selection

```
typedef struct {
    Gsel_crit_type   type;   /* selection type        */
    Gsel_crit        crit;   /* selection criterion   */
} Gsel;
```

Gsel_crit selection criterion

```
typedef struct GSEL_CRIT *Gsel_crit_ptr;
typedef struct GSEL_CRIT {
    Gsel_crit_op        op;                     /* operation */
    union {
        Gnameset        compar;                 /* comparison
                                                   (EQUALS|CONTAINS|IS-IN) */;
        struct Gdyad_sel_crit {
                        Gsel_crit_ptr   left;   /* left selection operand */
                        Gsel_crit_ptr   right;  /* right selection operand */
        } dyad;         /* dyadic criterion (AND|OR) */
        Gsel_crit_ptr   monad;                  /* monadic criterion (NOT) */
    } value;
} Gsel_crit;
```

Gsel_crit_op selection criterion operation

```
typedef enum {
    GOP_CONTAINS,
    GOP_IS_IN,
    GOP_EQUALS,
    GOP_SEL_ALL,
    GOP_REJ_ALL,
    GOP_AND,
    GOP_OR,
    GOP_NOT
} Gsel_crit_op;
```

Gsel_crit_type selection criterion type

```
typedef enum {
    GSEL_DISP,
    GSEL_HIGHL,
    GSEL_DET
} Gsel_crit_type;
```

Gsel_table selection table

```
typedef struct {
    Gint   num_sels;   /* number of selections    */
    Gsel   *sels;      /* table of selections     */
} Gsel_table;
```

Gsense_flag sense flag

```
typedef enum {
    GFLAG_CLOCKWISE,
    GFLAG_ANTI_CLOCKWISE
} Gsense_flag;
```

Gseq_specif sequence specifier

```
typedef struct {
    Gint     inst_name;      /* instance name             */
    Gint     stencil_name;   /* existing stencil name     */
    Gpoint   ref_point;      /* reference point           */
} Gseq_specif;
```

Gseq_specif_list sequence specifier list

```
typedef struct {
    Gint           num_seq_specifs;   /* number of sequence specifiers   */
    Gseq_specif    *seq_specif;       /* list of sequence specifiers     */
} Gseq_specif_list;
```

Gshield_ind shielding indicator

```
typedef enum {
    GIND_NO_SHIELD,
    GIND_SHIELD
} Gshield_ind;
```

Gstencil_attr_name stencil attribute name

```
typedef enum {
    GNAME_TOP_Y,
    GNAME_CAP_Y,
    GNAME_HALF_Y,
    GNAME_BASE_Y,
    GNAME_BOTTOM_Y,
    GNAME_CTR_Y,
    GNAME_LEFT_X,
    GNAME_RIGHT_X,
    GNAME_CTR_X,
    GNAME_CTR,
    GNAME_ORIGIN,
    GNAME_CTR_TOP,
    GNAME_CTR_BOTTOM,
    GNAME_CTR_LEFT,
    GNAME_CTR_RIGHT,
    GNAME_TOP_LEFT,
    GNAME_TOP_RIGHT,
    GNAME_BOTTOM_LEFT,
    GNAME_BOTTOM_RIGHT
} Gstencil_attr_name;
```

Gstencil_attr_value stencil attribute value

```
typedef struct {
    Gstencil_attr_name   name;            /* attribute name    */
    union {
        Gfloat           top_y;           /* top y             */
        Gfloat           half_y;          /* half y            */
        Gfloat           base_y;          /* base y            */
        Gfloat           bottom_y;        /* bottom y          */
        Gfloat           ctr_y;           /* centre y          */
        Gfloat           left_x;          /* left x            */
        Gfloat           ctr_x;           /* centre x          */
        Gfloat           right_x;         /* right x           */
        Gpoint           ctr;             /* centre            */
        Gpoint           origin;          /* origin            */
        Gpoint           ctr_top;         /* centre top        */
        Gpoint           ctr_bottom;      /* centre bottom     */
        Gpoint           ctr_left;        /* centre left       */
        Gpoint           ctr_right;       /* centre right      */
        Gpoint           top_left;        /* top left          */
        Gpoint           top_right;       /* topr ight         */
        Gpoint           bottom_left;     /* bottom left       */
        Gpoint           bottom_right;    /* bottom right      */
    } value;
} Gstencil_attr_value;
```

Gstencil_attrs stencil attributes

```
typedef struct {
    Gfloat    top_y;           /* top y          */
    Gfloat    cap_y;           /* capy           */
    Gfloat    half_y;          /* half y         */
    Gfloat    base_y;          /* base y         */
    Gfloat    bottom_y;        /* bottom y       */
    Gfloat    ctr_y;           /* centre y       */
    Gfloat    left_x;          /* left x         */
    Gfloat    ctr_x;           /* centre x       */
    Gfloat    right_x;         /* right x        */
    Gpoint    ctr;             /* centre         */
    Gpoint    origin;          /* origin         */
    Gpoint    ctr_top;         /* centre top     */
    Gpoint    ctr_bottom;      /* centre bottom  */
    Gpoint    ctr_right;       /* centre right   */
    Gpoint    top_left;        /* top left       */
    Gpoint    top_right;       /* topr ight      */
    Gpoint    bottom_left;     /* bottom left    */
    Gpoint    bottom_right;    /* bottom right   */
} Gstencil_attrs;
```

`Gstencil_op_st` stencil operating state

```
typedef enum {
    GOP_ST_STCL,
    GOP_ST_STOP
  } Gstencil_op_st;
```

`Gstencil_sl` stencil state list

```
typedef struct {
    Gint             name;    /* stencil name         */
    Gstencil_attrs   attrs;   /* stencil attributes   */
} Gstencil_sl;
```

`Gstencil_sl_list` list of stencil state lists

```
typedef struct {
    Gint          num_stencil_sls;   /* number of stencil_sls in the list   */
    Gstencil_sl   *stencil_sls;      /* list of stencil_sls                 */
} Gstencil_sl_list;
```

`Gstore` store

```
typedef void *Gstore;
```

`Gstroke` stroke value

```
typedef struct {
    Gpoint_list   point_list;     /* stroke point list    */
    Gint          norm_tran_num;  /* norm. tran. number   */
} Gstroke;
```

`Gtext_align` text alignment

```
typedef struct {
    enum Ghor_text_align { /* horizontal text alignment */
        GHOR_NORM,
        GHOR_LEFT,
        GHOR_CTR,
        GHOR_RIGHT
    } hor;
    enum Gvert_text_align { /*  vertical text alignment */
        GVERT_NORM,
        GVERT_TOP,
        GVERT_CAP,
        GVERT_HALF,
        GVERT_BASE,
        GVERT_BOTTOM
    } vert;
} Gtext_align;
```

`Gtext_bundle` text bundle

```
typedef struct {
    Gtext_font_prec   text_font_prec;   /* text font and precision      */
    Gfloat            char_expan;       /* character expansion factor   */
    Gfloat            char_space;       /* character spacing            */
    Gcolr_kind        colr_kind;        /* colour kind                  */
    Gint              colr_ind;         /* colour index                 */
    Gcolr_rep         dir_colr;         /* direct colour                */
} Gtext_bundle;
```

`Gtext_bundle_table` text bundle table

```
typedef struct {
    Gint            num_inds;    /* number of character indices    */
    Gint            *inds;       /* list of character indices      */
    Gtext_bundle    *bundles;    /* list of text bundles           */
} Gtext_bundle_table;
```

`Gtext_extent` text extent

```
typedef struct {
    Gpoint    concat_point;    /* concatenation point          */
    Gpoint    paral[4];        /* text extent parallelogram    */
} Gtext_extent;
```

`Gtext_facs` text facilities

```
typedef struct {
    Gint              num_font_precs;    /* number of fonts and precisions   */
    Gtext_font_prec   *font_precs;       /* list of fonts and precisions     */
    Gint              num_char_hts;      /* number of character heights      */
    Gfloat            min_char_ht;       /* min. character height            */
    Gfloat            max_char_ht;       /* max. character height            */
    Gint              num_char_expans;   /* number of char. expansion
                                            factors                          */
    Gfloat            min_char_expan;    /* min. expansion factor            */
    Gfloat            max_char_expan;    /* max. expansion factor            */
    Gint              num_pred_inds;     /* number of predef. text indices   */
} Gtext_facs;
```

`Gtext_font_prec` text font [and] precision

```
typedef struct {
    Gint          font;    /* character font    */
    Gtext_prec    prec;    /* text precision    */
} Gtext_font_prec;
```

Gtext_path text path

```
typedef enum {
    GPATH_RIGHT,
    GPATH_LEFT,
    GPATH_UP,
    GPATH_DOWN
} Gtext_path;
```

Gtext_prec text precision

```
typedef enum {
    GPREC_STRING,
    GPREC_CHAR,
    GPREC_STROKE
} Gtext_prec;
```

Gtiling_op_st tiling operating state

```
typedef enum {
    GOP_ST_TLCL,
    GOP_ST_TLOP
  } Gtiling_op_st;
```

Gtiling_params tiling parameters

```
typedef struct {
    Gtiling_type                type;           /* tiling type */
    union {
        struct Gtiling_line_area { /* line or area primitives in tiling */
            Gcolr_specif        colr_specif;    /* colour specifier */
            union { /* geometric data of line or area primitive */
                Gpoint_list_list    polyline_set;   /* polyline set*/
                Gnurb_list          nurb_set;       /* NURB set*/
                Gconic_sec_list     conic_sec_set;  /* conic section set*/

                Gpoint_list_list    fill_area_set;  /* fill area set*/
                Gnurb_list          closed_nurb_set;/* closed NURB set*/
                Gconic_sec_list     ell_sec_set;    /* elliptic sector set*/
                Gconic_sec_list     ell_seg_set;    /* elliptic segment set*/
                Gell_disc_list      ell_disc_set;   /* elliptic disc set*/
            } geom_data;
        } tiling_line_area;
        Gdesign                 design;         /* design */
    } value;
} Gtiling_params;
```

`Gtiling_type` tiling type

```
typedef enum {
    GTILING_LINE_SET,
    GTILING_NURB_SET,
    GTILING_CONIC_SEC_SET,
    GTILING_FILL_SET,
    GTILING_CLOSED_NURB_SET,
    GTILING_ELL_SEC_SET,
    GTILING_ELL_SEG_SET,
    GTILING_ELL_DISC_SET,
    GTILING_DESIGN
} Gtiling_type;
```

`Gtran` transformation

```
typedef struct {
    Glimit   win;    /* window     */
    Glimit   vp;     /* viewport   */
} Gtran;
```

`Gtran_matrix` transformation matrix

```
typedef Gfloat Gtran_matrix[2][3];
```

`Gtran_mode` transformation mode

```
typedef enum {
    GTRAN_REPLACE,
    GTRAN_PRE,
    GTRAN_POST
} Gtran_mode;
```

`Gvec` vector

```
typedef struct {
    Gfloat   delta_x;    /* x coordinate   */
    Gfloat   delta_y;    /* y coordinate   */
} Gvec;
```

`Gview_rep` view representation

```
typedef struct {
    Gsel_crit      sel_crit;     /* view selection criterion   */
    Gtran_matrix   ori_matrix;   /* view orientation matrix    */
    Gtran_matrix   map_matrix;   /* view mapping matrix        */
    Gscissor       scissor;      /* view scissor               */
} Gview_rep;
```

Gvis visibility

```
typedef enum {
    GSEG_INVIS,
    GSEG_VIS
} Gvis;
```

Gvis_effects_st visual effects state

```
typedef enum {
    GST_SUPPR,
    GST_ALLOW
} Gvis_effects_st;
```

Gws_class workstation classification [display kind]

```
typedef enum {
    GCLASS_VEC,
    GCLASS_RASTER,
    GCLASS_OTHER
} Gws_class;
```

Gws_dt_entry workstation description table entry

```
typedef struct {
    Gws_dt_name                 name;                 /* entry name */
    union {
        Gws_type                type;                 /* workstation type     */
        Gws_status              ws_status;            /* ws. status           */
        Gdc_units               dc_units;             /* DC units             */
        Gdisp_space_size        disp_space_size;      /* displace
                                                         space size           */
        Gws_class               ws_class;             /* ws. classification   */

        Gline_facs              line_facs;            /* line facilities      */
        Gline_bundle_table      pred_line_bundles;    /* predefined line
                                                        bundle table          */

        Gmarker_facs            marker_facs;          /* marker
                                                         facilities           */
        Gmarker_bundle_table    pred_marker_bundles;  /* predefined marker
                                                         bundle table         */

        Garea_facs              area_facs;            /* area facilities      */
        Garea_bundle_table      pred_area_bundles;    /* predefined area
                                                         bundle table         */

        Gpat_table              pred_pat_reps;        /* predefined pattern
                                                         bundle table         */

        Gtext_facs              text_facs;            /* text facilities      */
        Gtext_bundle_table      pred_text_bundles;    /* predefined text
                                                         bundle table         */

        Gcolr_facs              colr_facs;            /* colour facilities    */
        Gcolr_table             pred_colr_reps;       /* predefined colour
                                                         bundle table         */
        Gint_list               gdp_ids;              /* list of available
                                                         GDPs                 */
        Gdev_id_list            dev_ids;              /* list of available
                                                         input devices        */
        Gin_dev_init_value_list in_init_values;       /* input device
                                                         initial values       */
        Gint_list               meas_ids;             /* list of available
                                                         measures             */
        Gint_list               trigger_ids;          /* list of available
                                                         triggers             */
        Gdev_comp_list          dev_comps;            /* device compositions  */
    } value;
} Gws_dt_entry;
```

Gws_dt_name workstation description table name

```
typedef enum {
    GWS_DT_TYPE,
    GWS_DT_STATUS,
    GWS_DT_DC_UNITS,
    GWS_DT_DISP_SPACE_SIZE,
    GWS_DT_CLASS,
    GWS_DT_LINE_FACS,
    GWS_DT_LINE_BUNDLE_TABLE,
    GWS_DT_MARKER_FACS,
    GWS_DT_MARKER_BUNDLE_TABLE,
    GWS_DT_AREA_FACS,
    GWS_DT_AREA_BUNDLE_TABLE,
    GWS_DT_PAT_TABLE,
    GWS_DT_TEXT_FACS,
    GWS_DT_TEXT_BUNDLE_TABLE,
    GWS_DT_COLR_FACS,
    GWS_DT_COLR_TABLE,
    GWS_DT_GDP_IDS,
    GWS_DT_DEV_IDS,
    GWS_DT_IN_DEV_INIT_VALUES,
    GWS_DT_MEAS_IDS,
    GWS_DT_TRIGGER_IDS,
    GWS_DT_DEV_COMPS
} Gws_dt_name;
```

Gws_sl_entry workstation state list entry

```
typedef struct {
    Gws_sl_name             name;                    /* entry name */
    union {
        Gws_st              ws_st;                   /* ws. state
                                                        (ACTIVE|INACTIVE)    */
        struct {
                            void                     *specif;
                            Gint                     type;
        } specif_type; /* specifier and type */
        Gline_bundle_table  line_bundle_table;       /* line bundle
                                                        table                */
        Gmarker_bundle_table marker_bundle_table;    /* marker
                                                        bundle table         */
        Gtext_bundle_table  text_bundle_table;       /* text
                                                        bundle table         */
        Garea_bundle_table  area_bundle_table;       /* area
                                                        bundle table         */
        Gpat_table          pat_table;               /* pattern
                                                        bundle table         */
        Gcolr_table         colr_table;              /* colour
                                                        bundle table         */
        Gtran               ws_win_vp;               /* window
                                                        and viewport         */
        Gvis_effects_st     vis_effects_st;          /* visual effects
                                                        state                */
        Gop_modes_init_values op_modes_init_values;  /* input device
                                                        operating modes
                                                        and initial values   */
        Gsel_table          sel_table;               /* selection table      */
        Gint_list           stored_segs;             /* set of
                                                        stored segments      */
        Gview_rep           view_rep[16];            /* view bundle table    */
        Gint                view_pri[16];            /* view priorities      */
    } entry;
} Gws_sl_entry;
```

Gws_sl_entry_list workstation state list entry value list

```
typedef struct {
    Gint            num_entries;    /* number of WSL entry values
                                       in list                      */
    Gws_sl_entry    *entries;       /* list of WSL entry values     */
} Gws_sl_entry_list;
```

Gws_sl_name workstation state list name

```
typedef enum{
    GWS_SL_ST,
    GWS_SL_CONN_TYPE,
    GWS_SL_LINE_BUNDLE_TABLE,
    GWS_SL_MARKER_BUNDLE_TABLE,
    GWS_SL_TEXT_BUNDLE_TABLE,
    GWS_SL_AREA_BUNDLE_TABLE,
    GWS_SL_PAT_TABLE,
    GWS_SL_COLR_TABLE,
    GWS_SL_WIN_VP,
    GWS_SL_VIS_EFFECTS_ST,
    GWS_SL_OP_MODES_INIT_VALS,
    GWS_SL_SEL_TABLE,
    GWS_SL_STORED_SEGS,
    GWS_SL_VIEW_TABLE,
    GWS_SL_VIEW_PRIS
} Gws_sl_name;
```

Gws_sl_name_list workstation state list entry name list

```
typedef struct {
    Gint          num_names;   /* number of WSL entry names in list   */
    Gws_sl_name   *names;      /* list of WSL entry names             */
} Gws_sl_name_list;
```

Gws_st workstation state

```
typedef enum {
    GWS_INACTIVE,
    GWS_ACTIVE
} Gws_st;
```

Gws_status workstation status

```
typedef enum {
    GWS_UP_TO_DATE,
    GWS_NOT_UP_TO_DATE
} Gws_status;
```

Gws_type workstation type

```
typedef struct {
    Gint   gen_type;       /* generic type          */
    void   *specif_info;   /* specific information  */
} Gws_type;
```

6 Macro definitions

6.1 Function identifiers

The error functions require a unique mapping of the GKS functions to a set of numbers. The names for these function identifiers are the same as the GKS function names except that the sentinel character has been replaced by "Gfn_".

Below, the function identifier macros are listed, with their numerical values. These values are identical with the values of the function identifiers in the previous edition of ISO/IEC 8651-4, if the corresponding functions were also defined in the 1991 edition.

6.1.1 In order of appearance

```
#define   Gfn_open_gks                                        (0)
#define   Gfn_close_gks                                       (1)
#define   Gfn_save_gks_sl                                     (200)
#define   Gfn_restore_gks_sl                                  (201)
#define   Gfn_escape                                          (11)
#define   Gfn_emergency_close_gks                             (153)
#define   Gfn_err_hand                                        (154)
#define   Gfn_err_log                                         (155)
#define   Gfn_route                                           (202)
#define   Gfn_create_out_prim                                 (203)
#define   Gfn_polyline                                        (12)
#define   Gfn_polyline_set                                    (204)
#define   Gfn_nurb_set                                        (205)
#define   Gfn_conic_sec_set                                   (206)
#define   Gfn_polymarker                                      (13)
#define   Gfn_fill_area                                       (15)
#define   Gfn_fill_area_set                                   (207)
#define   Gfn_ell_sec_set                                     (208)
#define   Gfn_ell_seg_set                                     (209)
#define   Gfn_ell_disc_set                                    (210)
#define   Gfn_closed_nurb_set                                 (211)
#define   Gfn_text                                            (14)
#define   Gfn_cell_array                                      (16)
#define   Gfn_design                                          (212)
#define   Gfn_gdp                                             (17)
#define   Gfn_open_stencil                                    (213)
#define   Gfn_close_stencil                                   (214)
#define   Gfn_rename_stencil                                  (215)
#define   Gfn_del_stencil                                     (216)
#define   Gfn_create_stencil_bndry                            (217)
#define   Gfn_create_stencil_conts                            (218)
#define   Gfn_set_cont_attr                                   (219)
#define   Gfn_set_stencil_attr                                (220)
#define   Gfn_get_stencil_attr                                (221)
#define   Gfn_inst_stencil                                    (222)
#define   Gfn_inst_stencil_path                               (223)
#define   Gfn_inst_stencil_seq_path                           (224)
#define   Gfn_open_tiling                                     (225)
```

```
#define   Gfn_close_tiling                          (226)
#define   Gfn_rename_tiling                         (227)
#define   Gfn_del_tiling                            (228)
#define   Gfn_create_tiling_comp                    (229)
#define   Gfn_set_prim_attr                         (230)
#define   Gfn_set_line_ind                          (18)
#define   Gfn_set_linetype                          (19)
#define   Gfn_set_linewidth                         (20)
#define   Gfn_set_line_colr_specif                  (231)
#define   Gfn_set_line_colr_ind                     (21)
#define   Gfn_set_marker_ind                        (22)
#define   Gfn_set_marker_type                       (23)
#define   Gfn_set_marker_size                       (24)
#define   Gfn_set_marker_colr_specif                (232)
#define   Gfn_set_marker_colr_ind                   (25)
#define   Gfn_set_text_ind                          (26)
#define   Gfn_set_text_font_prec                    (27)
#define   Gfn_set_char_expan                        (28)
#define   Gfn_set_char_space                        (29)
#define   Gfn_set_text_colr_specif                  (233)
#define   Gfn_set_text_colr_ind                     (30)
#define   Gfn_set_char_ht                           (31)
#define   Gfn_set_char_up_vec                       (32)
#define   Gfn_set_text_path                         (33)
#define   Gfn_set_text_align                        (34)
#define   Gfn_set_text_skew_angle                   (234)
#define   Gfn_set_area_ind                          (235)
#define   Gfn_set_int_style                         (236)
#define   Gfn_set_int_style_ind                     (237)
#define   Gfn_set_int_colr_specif                   (238)
#define   Gfn_set_int_colr_ind                      (239)
#define   Gfn_set_pat_size                          (39)
#define   Gfn_set_pat_ref_point                     (40)
#define   Gfn_set_global_tran_matrix                (240)
#define   Gfn_set_local_tran_matrix                 (241)
#define   Gfn_set_edge_flag                         (242)
#define   Gfn_set_edgetype                          (243)
#define   Gfn_set_edgewidth                         (244)
#define   Gfn_set_edge_colr_specif                  (245)
#define   Gfn_set_edge_colr_ind                     (246)
#define   Gfn_set_asfs                              (41)
#define   Gfn_set_pick_id                           (42)
#define   Gfn_set_nameset                           (247)
#define   Gfn_set_scissor_set                       (248)
#define   Gfn_add_set_names_nameset                 (249)
#define   Gfn_remove_set_names_nameset              (250)
#define   Gfn_add_set_scissors_scissor_set          (251)
#define   Gfn_remove_set_scissors_scissor_set       (252)
#define   Gfn_set_win_vp                            (49)
#define   Gfn_set_win                               (49)
#define   Gfn_set_vp                                (50)
#define   Gfn_set_vp_in_pri                         (50)
```

```
#define   Gfn_set_scissor_mode                       (253)
#define   Gfn_set_norm_tran_num                      (254)
#define   Gfn_sel_norm_tran                          (52)
#define   Gfn_del_prims_ndc_pic                      (255)
#define   Gfn_remove_set_names_ndc_pic               (256)
#define   Gfn_add_set_names_ndc_pic                  (257)
#define   Gfn_set_ndc_pic_prim_attr                  (258)
#define   Gfn_add_set_scissors_ndc_pic               (259)
#define   Gfn_remove_set_scissors_ndc_pic            (260)
#define   Gfn_reorder_ndc_pic                        (261)
#define   Gfn_copy_ndc_pic_ndc_mf                    (262)
#define   Gfn_copy_ndc_mf_pic_ndc_pic                (263)
#define   Gfn_open_pic_part                          (264)
#define   Gfn_close_pic_part                         (265)
#define   Gfn_ar_pic_part                            (266)
#define   Gfn_ret_pic_part_ar                        (267)
#define   Gfn_reopen_pic_part                        (268)
#define   Gfn_append_pic_part                        (269)
#define   Gfn_rename_pic_part                        (270)
#define   Gfn_del_pic_part                           (271)
#define   Gfn_copy_pic_part_pic_part_store           (272)
#define   Gfn_copy_ndc_pic_pic_part_store            (273)
#define   Gfn_set_in_dev_mode                        (274)
#define   Gfn_set_loc_mode                           (75)
#define   Gfn_set_stroke_mode                        (76)
#define   Gfn_set_val_mode                           (77)
#define   Gfn_set_choice_mode                        (78)
#define   Gfn_set_pick_mode                          (79)
#define   Gfn_set_string_mode                        (80)
#define   Gfn_req_in                                 (275)
#define   Gfn_req_loc                                (81)
#define   Gfn_req_stroke                             (82)
#define   Gfn_req_val                                (83)
#define   Gfn_req_choice                             (84)
#define   Gfn_req_pick                               (85)
#define   Gfn_req_string                             (86)
#define   Gfn_sample_in                              (276)
#define   Gfn_sample_loc                             (87)
#define   Gfn_sample_stroke                          (88)
#define   Gfn_sample_val                             (89)
#define   Gfn_sample_choice                          (90)
#define   Gfn_sample_pick                            (91)
#define   Gfn_sample_string                          (92)
#define   Gfn_await_in                               (277)
#define   Gfn_await_event                            (93)
#define   Gfn_flush_dev_events                       (278)
#define   Gfn_get_loc                                (95)
#define   Gfn_get_stroke                             (96)
#define   Gfn_get_val                                (97)
#define   Gfn_get_choice                             (98)
#define   Gfn_get_pick                               (99)
#define   Gfn_get_string                             (100)
```

```
#define   Gfn_set_font_ind_map                       (279)
#define   Gfn_get_glyph_name                         (280)
#define   Gfn_set_char_code                          (281)
#define   Gfn_audit                                  (282)
#define   Gfn_write_user_rec_audit                   (283)
#define   Gfn_playback                               (284)
#define   Gfn_read_item_func_name_audit              (285)
#define   Gfn_read_item_audit                        (286)
#define   Gfn_proc_audit_item                        (287)
#define   Gfn_eval_tran_matrix                       (105)
#define   Gfn_accum_tran_matrix                      (106)
#define   Gfn_eval_circle                            (288)
#define   Gfn_eval_ell                               (289)
#define   Gfn_eval_circ_arc_3_point                  (290)
#define   Gfn_eval_circ_arc_ctr                      (291)
#define   Gfn_eval_ell_arc                           (292)
#define   Gfn_eval_hyp_arc                           (293)
#define   Gfn_eval_par_arc                           (294)
#define   Gfn_eval_wc_ord                            (295)
#define   Gfn_set_err_hand                           (156)
#define   Gfn_create_store                           (296)
#define   Gfn_del_store                              (297)
#define   Gfn_open_ws                                (2)
#define   Gfn_close_ws                               (3)
#define   Gfn_save_ws_sl                             (298)
#define   Gfn_restore_ws_sl                          (299)
#define   Gfn_get_ws_status                          (300)
#define   Gfn_remove_backdrop                        (301)
#define   Gfn_set_rep                                (302)
#define   Gfn_set_line_rep                           (43)
#define   Gfn_set_marker_rep                         (44)
#define   Gfn_set_text_rep                           (45)
#define   Gfn_set_area_rep                           (303)
#define   Gfn_set_pat_rep                            (47)
#define   Gfn_set_colr_rep                           (48)
#define   Gfn_set_view_pri                           (304)
#define   Gfn_set_view_sel_crit                      (305)
#define   Gfn_set_view                               (306)
#define   Gfn_set_ws_vis_effects                     (307)
#define   Gfn_set_ws_win_vp                          (308)
#define   Gfn_set_ws_win                             (54)
#define   Gfn_set_ws_vp                              (55)
#define   Gfn_def_in_dev                             (309)
#define   Gfn_init_in_dev                            (310)
#define   Gfn_init_loc                               (69)
#define   Gfn_init_stroke                            (70)
#define   Gfn_init_val                               (71)
#define   Gfn_init_choice                            (72)
#define   Gfn_init_pick                              (73)
#define   Gfn_init_string                            (74)
#define   Gfn_set_ws_sel_crit                        (311)
#define   Gfn_copy_real_pic_real_mf                  (312)
```

```
#define   Gfn_copy_blank_real_pic_real_mf          (313)
#define   Gfn_copy_real_mf_pic_backdrop            (314)
#define   Gfn_message                              (10)
#define   Gfn_reset_specif_ws_dt_entry_st          (315)
#define   Gfn_get_text_extent                      (316)
#define   Gfn_eval_view_ori_matrix                 (317)
#define   Gfn_eval_view_map_matrix                 (318)
#define   Gfn_conv_colr                            (319)
#define   Gfn_create_seg                           (56)
#define   Gfn_close_seg                            (57)
#define   Gfn_rename_seg                           (58)
#define   Gfn_del_seg                              (59)
#define   Gfn_del_seg_ws                           (60)
#define   Gfn_assoc_seg_ws                         (61)
#define   Gfn_copy_seg_ws                          (62)
#define   Gfn_insert_seg                           (63)
#define   Gfn_set_seg_attr                         (320)
#define   Gfn_set_seg_tran                         (64)
#define   Gfn_set_vis                              (65)
#define   Gfn_set_highl                            (66)
#define   Gfn_set_det                              (68)
#define   Gfn_set_seg_pri                          (67)
#define   Gfn_activate_ws                          (4)
#define   Gfn_deactivate_ws                        (5)
#define   Gfn_clear_ws                             (6)
#define   Gfn_get_map_seg_name                     (321)
#define   Gfn_get_map_ws_id                        (322)
```

6.1.2 In alphabetical order

```
#define   Gfn_accum_tran_matrix                    (106)
#define   Gfn_activate_ws                          (4)
#define   Gfn_add_set_names_nameset                (249)
#define   Gfn_add_set_names_ndc_pic                (257)
#define   Gfn_add_set_scissors_ndc_pic             (259)
#define   Gfn_add_set_scissors_scissor_set         (251)
#define   Gfn_append_pic_part                      (269)
#define   Gfn_ar_pic_part                          (266)
#define   Gfn_assoc_seg_ws                         (61)
#define   Gfn_audit                                (282)
#define   Gfn_await_event                          (93)
#define   Gfn_await_in                             (277)
#define   Gfn_cell_array                           (16)
#define   Gfn_clear_ws                             (6)
#define   Gfn_close_gks                            (1)
#define   Gfn_close_pic_part                       (265)
#define   Gfn_close_seg                            (57)
#define   Gfn_close_stencil                        (214)
#define   Gfn_close_tiling                         (226)
#define   Gfn_close_ws                             (3)
#define   Gfn_closed_nurb_set                      (211)
#define   Gfn_conic_sec_set                        (206)
```

```
#define   Gfn_conv_colr                            (319)
#define   Gfn_copy_blank_real_pic_real_mf          (313)
#define   Gfn_copy_ndc_mf_pic_ndc_pic              (263)
#define   Gfn_copy_ndc_pic_ndc_mf                  (262)
#define   Gfn_copy_ndc_pic_pic_part_store          (273)
#define   Gfn_copy_pic_part_pic_part_store         (272)
#define   Gfn_copy_real_mf_pic_backdrop            (314)
#define   Gfn_copy_real_pic_real_mf                (312)
#define   Gfn_copy_seg_ws                          (62)
#define   Gfn_create_out_prim                      (203)
#define   Gfn_create_seg                           (56)
#define   Gfn_create_stencil_bndry                 (217)
#define   Gfn_create_stencil_conts                 (218)
#define   Gfn_create_store                         (296)
#define   Gfn_create_tiling_comp                   (229)
#define   Gfn_deactivate_ws                        (5)
#define   Gfn_def_in_dev                           (309)
#define   Gfn_del_pic_part                         (271)
#define   Gfn_del_prims_ndc_pic                    (255)
#define   Gfn_del_seg                              (59)
#define   Gfn_del_seg_ws                           (60)
#define   Gfn_del_stencil                          (216)
#define   Gfn_del_store                            (297)
#define   Gfn_del_tiling                           (228)
#define   Gfn_design                               (212)
#define   Gfn_ell_disc_set                         (210)
#define   Gfn_ell_sec_set                          (208)
#define   Gfn_ell_seg_set                          (209)
#define   Gfn_emergency_close_gks                  (153)
#define   Gfn_err_hand                             (154)
#define   Gfn_err_log                              (155)
#define   Gfn_escape                               (11)
#define   Gfn_eval_circ_arc_3_point                (290)
#define   Gfn_eval_circ_arc_ctr                    (291)
#define   Gfn_eval_circle                          (288)
#define   Gfn_eval_ell                             (289)
#define   Gfn_eval_ell_arc                         (292)
#define   Gfn_eval_hyp_arc                         (293)
#define   Gfn_eval_par_arc                         (294)
#define   Gfn_eval_tran_matrix                     (105)
#define   Gfn_eval_view_map_matrix                 (318)
#define   Gfn_eval_view_ori_matrix                 (317)
#define   Gfn_eval_wc_ord                          (295)
#define   Gfn_fill_area                            (15)
#define   Gfn_fill_area_set                        (207)
#define   Gfn_flush_dev_events                     (278)
#define   Gfn_gdp                                  (17)
#define   Gfn_get_choice                           (98)
#define   Gfn_get_glyph_name                       (280)
#define   Gfn_get_loc                              (95)
#define   Gfn_get_map_seg_name                     (321)
#define   Gfn_get_map_ws_id                        (322)
```

```
#define   Gfn_get_pick                              (99)
#define   Gfn_get_stencil_attr                      (221)
#define   Gfn_get_string                            (100)
#define   Gfn_get_stroke                            (96)
#define   Gfn_get_text_extent                       (316)
#define   Gfn_get_val                               (97)
#define   Gfn_get_ws_status                         (300)
#define   Gfn_init_choice                           (72)
#define   Gfn_init_in_dev                           (310)
#define   Gfn_init_loc                              (69)
#define   Gfn_init_pick                             (73)
#define   Gfn_init_string                           (74)
#define   Gfn_init_stroke                           (70)
#define   Gfn_init_val                              (71)
#define   Gfn_insert_seg                            (63)
#define   Gfn_inst_stencil                          (222)
#define   Gfn_inst_stencil_path                     (223)
#define   Gfn_inst_stencil_seq_path                 (224)
#define   Gfn_message                               (10)
#define   Gfn_nurb_set                              (205)
#define   Gfn_open_gks                              (0)
#define   Gfn_open_pic_part                         (264)
#define   Gfn_open_stencil                          (213)
#define   Gfn_open_tiling                           (225)
#define   Gfn_open_ws                               (2)
#define   Gfn_playback                              (284)
#define   Gfn_polyline                              (12)
#define   Gfn_polyline_set                          (204)
#define   Gfn_polymarker                            (13)
#define   Gfn_proc_audit_item                       (287)
#define   Gfn_read_item_audit                       (286)
#define   Gfn_read_item_func_name_audit             (285)
#define   Gfn_remove_backdrop                       (301)
#define   Gfn_remove_set_names_nameset              (250)
#define   Gfn_remove_set_names_ndc_pic              (256)
#define   Gfn_remove_set_scissors_ndc_pic           (260)
#define   Gfn_remove_set_scissors_scissor_set       (252)
#define   Gfn_rename_pic_part                       (270)
#define   Gfn_rename_seg                            (58)
#define   Gfn_rename_stencil                        (215)
#define   Gfn_rename_tiling                         (227)
#define   Gfn_reopen_pic_part                       (268)
#define   Gfn_reorder_ndc_pic                       (261)
#define   Gfn_req_choice                            (84)
#define   Gfn_req_in                                (275)
#define   Gfn_req_loc                               (81)
#define   Gfn_req_pick                              (85)
#define   Gfn_req_string                            (86)
#define   Gfn_req_stroke                            (82)
#define   Gfn_req_val                               (83)
#define   Gfn_reset_specif_ws_dt_entry_st           (315)
#define   Gfn_restore_gks_sl                        (201)
```

```
#define   Gfn_restore_ws_sl                                 (299)
#define   Gfn_ret_pic_part_ar                               (267)
#define   Gfn_route                                         (202)
#define   Gfn_sample_choice                                 (90)
#define   Gfn_sample_in                                     (276)
#define   Gfn_sample_loc                                    (87)
#define   Gfn_sample_pick                                   (91)
#define   Gfn_sample_string                                 (92)
#define   Gfn_sample_stroke                                 (88)
#define   Gfn_sample_val                                    (89)
#define   Gfn_save_gks_sl                                   (200)
#define   Gfn_save_ws_sl                                    (298)
#define   Gfn_sel_norm_tran                                 (52)
#define   Gfn_set_area_ind                                  (235)
#define   Gfn_set_area_rep                                  (303)
#define   Gfn_set_asfs                                      (41)
#define   Gfn_set_char_code                                 (281)
#define   Gfn_set_char_expan                                (28)
#define   Gfn_set_char_ht                                   (31)
#define   Gfn_set_char_space                                (29)
#define   Gfn_set_char_up_vec                               (32)
#define   Gfn_set_choice_mode                               (78)
#define   Gfn_set_colr_rep                                  (48)
#define   Gfn_set_cont_attr                                 (219)
#define   Gfn_set_det                                       (68)
#define   Gfn_set_edge_colr_ind                             (246)
#define   Gfn_set_edge_colr_specif                          (245)
#define   Gfn_set_edge_flag                                 (242)
#define   Gfn_set_edgetype                                  (243)
#define   Gfn_set_edgewidth                                 (244)
#define   Gfn_set_err_hand                                  (156)
#define   Gfn_set_font_ind_map                              (279)
#define   Gfn_set_global_tran_matrix                        (240)
#define   Gfn_set_highl                                     (66)
#define   Gfn_set_in_dev_mode                               (274)
#define   Gfn_set_int_colr_ind                              (239)
#define   Gfn_set_int_colr_specif                           (238)
#define   Gfn_set_int_style                                 (236)
#define   Gfn_set_int_style_ind                             (237)
#define   Gfn_set_line_colr_ind                             (21)
#define   Gfn_set_line_colr_specif                          (231)
#define   Gfn_set_line_ind                                  (18)
#define   Gfn_set_line_rep                                  (43)
#define   Gfn_set_linetype                                  (19)
#define   Gfn_set_linewidth                                 (20)
#define   Gfn_set_loc_mode                                  (75)
#define   Gfn_set_local_tran_matrix                         (241)
#define   Gfn_set_marker_colr_ind                           (25)
#define   Gfn_set_marker_colr_specif                        (232)
#define   Gfn_set_marker_ind                                (22)
#define   Gfn_set_marker_rep                                (44)
#define   Gfn_set_marker_size                               (24)
```

```
#define   Gfn_set_marker_type                      (23)
#define   Gfn_set_nameset                          (247)
#define   Gfn_set_ndc_pic_prim_attr                (258)
#define   Gfn_set_norm_tran_num                    (254)
#define   Gfn_set_pat_ref_point                    (40)
#define   Gfn_set_pat_rep                          (47)
#define   Gfn_set_pat_size                         (39)
#define   Gfn_set_pick_id                          (42)
#define   Gfn_set_pick_mode                        (79)
#define   Gfn_set_prim_attr                        (230)
#define   Gfn_set_rep                              (302)
#define   Gfn_set_scissor_mode                     (253)
#define   Gfn_set_scissor_set                      (248)
#define   Gfn_set_seg_attr                         (320)
#define   Gfn_set_seg_pri                          (67)
#define   Gfn_set_seg_tran                         (64)
#define   Gfn_set_stencil_attr                     (220)
#define   Gfn_set_string_mode                      (80)
#define   Gfn_set_stroke_mode                      (76)
#define   Gfn_set_text_align                       (34)
#define   Gfn_set_text_colr_ind                    (30)
#define   Gfn_set_text_colr_specif                 (233)
#define   Gfn_set_text_font_prec                   (27)
#define   Gfn_set_text_ind                         (26)
#define   Gfn_set_text_path                        (33)
#define   Gfn_set_text_rep                         (45)
#define   Gfn_set_text_skew_angle                  (234)
#define   Gfn_set_val_mode                         (77)
#define   Gfn_set_view                             (306)
#define   Gfn_set_view_pri                         (304)
#define   Gfn_set_view_sel_crit                    (305)
#define   Gfn_set_vis                              (65)
#define   Gfn_set_vp                               (50)
#define   Gfn_set_vp_in_pri                        (50)
#define   Gfn_set_win                              (49)
#define   Gfn_set_win_vp                           (49)
#define   Gfn_set_ws_sel_crit                      (311)
#define   Gfn_set_ws_vis_effects                   (307)
#define   Gfn_set_ws_vp                            (55)
#define   Gfn_set_ws_win                           (54)
#define   Gfn_set_ws_win_vp                        (308)
#define   Gfn_text                                 (14)
#define   Gfn_write_user_rec_audit                 (283)
```

6.2 Error codes

The error codes are represented by macros. The range of the numeric values of the macros is between 1000 and 2000.

Below, the error macros are listed.

```
#define  GE_NO_ERR                    (0)   /* No error */
#define  GE_GKS_NOT_OPEN              (1001)/* GKS not open */
```

```
#define  GE_GKS_OPEN                     (1002)/* GKS already open */
#define  GE_PP_NOT_OPEN                  (1003)/* Picture part not open */
#define  GE_PP_OPEN                      (1004)/* Picture part already open */
#define  GE_SEG_NOT_OPEN                 (1005)/* Segment not open */
#define  GE_SEG_OPEN                     (1006)/* Segment already open */
#define  GE_ST_NOT_OPEN                  (1007)/* Stencil not open */
#define  GE_ST_OPEN                      (1008)/* Stencil already open */
#define  GE_TL_NOT_OPEN                  (1009)/* Tiling not open */
#define  GE_TL_OPEN                      (1010)/* Tiling already open */
#define  GE_WS_TYPE_NOT_SUPP             (1011)/* Specified workstation type not sup-
                                               ported */
#define  GE_WS_NOT_OPEN                  (1012)/* Specified workstation not open */
#define  GE_WS_OPEN                      (1013)/* Specified workstation already open
                                               */
#define  GE_WS_NOT_ACTIVE                (1014)/* Specified workstation not active
                                               */
#define  GE_WS_ACTIVE                    (1015)/*  Specified workstation already
                                               active */
#define  GE_AUDIT_NOT_OPEN               (1016)/* Specified audit trail not open */
#define  GE_AUDIT_OPEN                   (1017)/* Specified audit trail already open
                                               */
#define  GE_PLAYBACK_NOT_OPEN            (1018)/* Specified playback not open */
#define  GE_PLAYBACK_OPEN                (1019)/* Specified playback already open */
#define  GE_PP_NOT_EXIST                 (1020)/* Specified picture part does not
                                               exist */
#define  GE_PP_NAME_USE                  (1021)/* Specified picture part name already
                                               in use */
#define  GE_OLD_PP_NOT_EXIST             (1022)/* Old picture part does not exist */
#define  GE_NEW_PP_NAME_USE              (1023)/* New picture part name already in
                                               use */
#define  GE_SRC_PP_NOT_EXIST             (1024)/* Source picture part does not exist
                                               */
#define  GE_SINK_PP_NOT_EXIST            (1025)/* Sink picture part does not exist */
#define  GE_SEG_NOT_EXIST                (1026)/* Segment does not exist */
#define  GE_SEG_NAME_IN_USE              (1027)/* Segment name already in use  */
#define  GE_ST_NOT_EXIST                 (1028)/* Specified stencil does not exist */
#define  GE_ST_NAME_USE                  (1029)/* Specified stencil name already in
                                               use */
#define  GE_OLD_ST_NOT_EXIST             (1030)/* Old stencil does not exist */
#define  GE_NEW_ST_NAME_USE              (1031)/* New stencil name already in use */
#define  GE_INST_NAME_EQ_ST_NAME_USE     (1032)/* Instance name same as stencil name
                                               already in use */
#define  GE_TL_NOT_EXIST                 (1033)/* Specified tiling does not exist */
#define  GE_TL_NAME_USE                  (1034)/* Specified tiling name already in
                                               use */
#define  GE_OLD_TL_NOT_EXIST             (1035)/* Old tiling does not exist */
#define  GE_NEW_TL_NAME_USE              (1036)/* New tiling name already in use */
#define  GE_IN_Q_OVERF                   (1037)/* Input queue has overflowed */
#define  GE_IN_Q_NOT_OVERF               (1038)/* Input queue has not overflowed ever
                                               or since the last invocation of
                                               INQUIRE INPUT QUEUE OVERFLOW */
#define  GE_IN_Q_OVERF_WS_CLOSED         (1039)/*  Input queue has overflowed but
```

```
                                                  associated workstation has been closed
                                                  */
#define  GE_ESCAPE_FUNC_NOT_SUPP        (1040)/* Specified escape function not sup-
                                                  ported */
#define  GE_GLYPH_NOT_AVAIL             (1041)/* Specified glyph not available */
#define  GE_FONT_NAME_NOT_AVAIL         (1042)/* Specified font name not available
                                                  */
#define  GE_LINETYPE_NOT_SUPP           (1043)/* Linetype not supported */
#define  GE_MARKER_TYPE_NOT_SUPP        (1044)/* Marker type not supported */
#define  GE_FONT_NOT_SUPP_PREC          (1045)/* Text font not supported for speci-
                                                  fied precision */
#define  GE_INT_STYLE_NOT_SUPP          (1046)/* Specified interior style not sup-
                                                  ported */
#define  GE_HATCH_STYLE_NOT_SUPP        (1047)/* Specified hatch style not supported
                                                  */
#define  GE_EDGETYPE_NOT_SUPP           (1048)/* Edgetype not supported */
#define  GE_COLR_MODEL_NOT_SUPP         (1049)/* Specified colour model not sup-
                                                  ported  */
#define  GE_GDP_NOT_SUPP                (1050)/* GDP not supported  */
#define  GE_IN_DEV_NOT_EXIST            (1051)/* Input device does not exist */
#define  GE_COMB_MEAS_TRIG_NOT_AVAIL    (1052)/* Specified combination of measures
                                                  and triggers cannot be provided */
#define  GE_SEG_NOT_EXIST_WS            (1053)/* Segment does not exist on specified
                                                  workstation */
#define  GE_PIC_NOT_EXIST_IN_MF         (1054)/* Specified picture does not exist in
                                                  specified metafile */
#define  GE_NO_ITEM_PLAYBACK            (1055)/* No item left on specified playback
                                                  */
#define  GE_PLAYBACK_ITEM_INVAL         (1056)/* Playback item invalid */
#define  GE_PP_NOT_EXIST_AR             (1057)/* Picture part does not exist in
                                                  archive */
#define  GE_EDGE_NOT_INTERSECT_PATH     (1058)/* Edge does not intersect path */
#define  GE_START_VEC_NOT_INTERSECT_HYP    (1059)/* Start vector does not inter-
                                                  sect hyperbola */
#define  GE_END_VEC_NOT_INTERSECT_HYP   (1060)/* End vector does not intersect
                                                  hyperbola */
#define  GE_OLD_PP_NAME_EQ_NEW_PP_NAME  (1061)/* Old picture part name and new pic-
                                                  ture part name are the same */
#define  GE_OLD_ST_NAME_EQ_NEW_ST_NAME  (1062)/* Old stencil name and new stencil
                                                  name are the same */
#define  GE_OLD_TL_NAME_EQ_NEW_TL_NAME  (1063)/* Old tiling name and new tiling name
                                                  are the same */


/*
 *      Binding specific errors
 */

#define  GE_START_IND_INVAL             (2201)/* Start index out of range */
#define  GE_APPL_LIST_LENGTH_LT_ZERO    (2202)/* Length of application list negative
                                                 */
#define  GE_ENUM_TYPE_INVAL             (2203)/* Enumeration type out of range */
#define  GE_ALLOC_STORE                 (2204)/* Error while allocating Store */
```

```
#define  GE_ALLOC_MEM_STORE             (2205)/*  Error  while  allocating  memory  for
                                             Store */

#define  GE_ERR_FILE_INVAL              (2250)/* Error file invalid */
#define  GE_INSUFF_STORE                (2251)/* Insufficient storage */
#define  GE_ENTRIES_INVAL               (2252)/* Entry values invalid */
#define  GE_ESC_FUNC_ID_INVAL           (2253)/*  Escape  function  identifier  invalid
                                             */
#define  GE_ESC_IN_DATA_INVAL           (2254)/* Escape input data record invalid */
#define  GE_ERR_NUM_INVAL               (2255)/* Error number invalid */
#define  GE_FUNC_NAME_INVAL             (2256)/* Function name invalid */
#define  GE_DIR_INVAL                   (2257)/* Direction invalid */
#define  GE_PRIM_TYPE_INVAL             (2258)/* Primitive type invalid */
#define  GE_NUM_POINTS_INVAL            (2259)/* Number of points invalid */
#define  GE_CODE_INVAL                  (2260)/* Code in string invalid */
#define  GE_DIMS_COLR_ARRAY_INVAL       (2261)/*  Dimensions  of  colour  array  invalid
                                             */
#define  GE_GDP_ID_INVAL                (2262)/* GDP identifier invalid */
#define  GE_GDP_DATA_INVAL              (2263)/* GDP data record invalid */
#define  GE_NURB_DEF_INVAL              (2264)/* NURB definition invalid */
#define  GE_CONIC_SEC_INVAL             (2265)/* Conic section invalid */
#define  GE_INSIDE_RULE_INVAL           (2266)/* Inside rule invalid */
#define  GE_SEQ_PATHS_INVAL             (2267)/* Sequence of paths invalid */
#define  GE_CONT_ATTR_NAME_INVAL        (2268)/* Contour attribute name invalid */
#define  GE_CONT_ATTR_VALUE_INVAL       (2269)/* Contour attribute value invalid */
#define  GE_ST_NAME_INVAL               (2270)/* Stencil name invalid */
#define  GE_OLD_NEW_ST_NAME_INVAL       (2271)/* Old or new stencil name invalid */
#define  GE_ST_ATTR_NAME_INVAL          (2272)/* Stencil attribute name invalid */
#define  GE_BNDRY_SPECIF_INVAL          (2273)/* Boundary specification invalid */
#define  GE_ST_ATTR_VALUE_INVAL         (2274)/* Stencil attribute value invalid */
#define  GE_INST_NAME_INVAL             (2275)/* Instance name invalid */
#define  GE_INST_TYPE_INVAL             (2276)/* Instance type invalid */
#define  GE_INST_SPECIF_INVAL           (2277)/* Instance specification invalid */
#define  GE_PATH_INVAL                  (2278)/* Path invalid */
#define  GE_TL_NAME_INVAL               (2279)/* Tiling name invalid */
#define  GE_REPL_TECH_INVAL             (2280)/* Replication technique invalid */
#define  GE_ATTR_NAME_INVAL             (2281)/* Attribute name invalid */
#define  GE_ATTR_VALUE_INVAL            (2282)/* Attribute value invalid */
#define  GE_SET_NAMES_INVAL             (2283)/* Set of names invalid */
#define  GE_SCISSOR_SPECIF_INVAL        (2284)/* Scissor specification invalid */
#define  GE_SCISSOR_IDS_INVAL           (2285)/* Scissor identifiers invalid */
#define  GE_SCISSOR_MODE_INVAL          (2286)/* Scissor mode invalid */
#define  GE_TRAN_NUM_INVAL              (2287)/* Transformation number invalid */
#define  GE_WIN_DEF_INVAL               (2288)/* Window definition invalid */
#define  GE_VP_DEF_INVAL                (2289)/* Viewport definition invalid */
#define  GE_SEL_CRIT_INVAL              (2290)/* Selection criterion invalid */
#define  GE_MF_SPECIF_INVAL             (2291)/* Metafile specifier invalid */
#define  GE_PIC_ID_INVAL                (2292)/* Picture identifier invalid */
#define  GE_PP_NAME_INVAL               (2293)/* Picture part name invalid */
#define  GE_AR_SPECIF_INVAL             (2294)/* Archive specifier invalid */
#define  GE_AR_NAME_PP_INVAL            (2295)/*   Archive   name   of   picture   part
                                             invalid */
```

```
#define  GE_SRC_PP_NAME_INVAL          (2296)/* Source picture part name invalid */
#define  GE_SINK_PP_NAME_INVAL         (2297)/* Sink picture part name invalid */
#define  GE_GLOBAL_TRAN_MODE_INVAL     (2298)/* Global transformation mode invalid
                                              */
#define  GE_LOCAL_TRAN_MODE_INVAL      (2299)/* Local transformation mode invalid
                                              */
#define  GE_OLD_PP_NAME_INVAL          (2300)/* Old picture part name invalid */
#define  GE_NEW_PP_NAME_INVAL          (2301)/* New picture part name invalid */
#define  GE_SCISSOR_SEL_INVAL          (2302)/* Scissor select invalid */
#define  GE_IN_DEV_INVAL               (2303)/* Input device identifier invalid */
#define  GE_OP_MODE_INVAL              (2304)/* Operating mode invalid */
#define  GE_TIMEOUT_INVAL              (2305)/* Timeout invalid */
#define  GE_FONT_IND_ZERO              (2306)/* Font index zero */
#define  GE_FONT_NAME_INVAL            (2307)/* Font name invalid */
#define  GE_CHAR_CODE_INVAL            (2308)/* Character code invalid */
#define  GE_AUDIT_ID_INVAL             (2309)/* Audit identifier invalid */
#define  GE_AUDIT_OP_INVAL             (2310)/* Audit operation invalid */
#define  GE_AUDIT_USER_DATA_INVAL      (2311)/* Audit user data record invalid */
#define  GE_PLAYBACK_OP_INVAL          (2312)/* Playback operation invalid */
#define  GE_PROC_OP_INVAL              (2313)/* Process operation invalid */
#define  GE_ENTRY_NAME_INVAL           (2314)/* Entry name invalid */
#define  GE_FIXED_POINT_INVAL          (2315)/* Fixed point invalid */
#define  GE_SHIFT_VEC_INVAL            (2316)/* Shift vector invalid */
#define  GE_RADIUS_INVAL               (2317)/* Radius or conjugate radius invalid
                                              */
#define  GE_RADII_ID                   (2318)/* Radii idenctical */
#define  GE_POINTS_COLLIN              (2319)/* Points collinear */
#define  GE_SENSE_FLAG_INVAL           (2320)/* Sense flag invalid */
#define  GE_ORD_SEL_INVAL              (2321)/* Ordinate selector invalid */
#define  GE_WS_ID_INVAL                (2322)/* Workskstation identifier invalid */
#define  GE_WS_TYPE_INVAL              (2323)/* Workskstation type invalid */
#define  GE_WS_SPECIF_INVAL            (2324)/* Workskstation specifier invalid */
#define  GE_REP_INVAL                  (2325)/* Representation invalid */
#define  GE_IND_INVAL                  (2326)/* Bundle ndex invalid */
#define  GE_TYPE_ZERO                  (2327)/* Linetype, marker type or edgetype
                                              zero */
#define  GE_SCALE_FAC_NEG              (2328)/* Linewidth, marker size or edgewidth
                                              scale factor negative */
#define  GE_CHAR_EXPAN_NOT_POS         (2329)/* Character expansion factor <= 0 */
#define  GE_EDGE_FLAG_INVAL            (2330)/* Edge flag invalid */
#define  GE_INT_STYLE_INVAL            (2331)/* Interior style invalid */
#define  GE_COLR_IND_INVAL             (2332)/* Colour index invalid */
#define  GE_PAT_IND_INVAL              (2333)/* Pattern index invalid */
#define  GE_COLR_SPECIF_INVAL          (2334)/* Colour specification invalid */
#define  GE_VIEW_IND_INVAL             (2335)/* View index invalid */
#define  GE_REF_IND_INVAL              (2336)/* Reference index invalid */
#define  GE_REL_PRI_INVAL              (2337)/* Relative priority invalid */
#define  GE_VIEW_SCISSOR_INVAL         (2338)/* View scissor invalid */
#define  GE_VIS_EFFECTS_ST_INVAL       (2339)/* Visual effects state invalid */
#define  GE_WS_WIN_LIMITS_INVAL        (2340)/* Workstation window limits invalid
                                              */
#define  GE_WS_VP_LIMITS_INVAL         (2341)/* Workstation viewport limits invalid
```

```
                                                  */
#define  GE_WS_VP_NOT_IN_DISP_SPACE    (2342)/* Workstation viewport not in display
                                              space */
#define  GE_MEAS_SEQ_INVAL             (2343)/* Measure sequence invalid */
#define  GE_TRIGGER_SET_INVAL          (2344)/* Trigger set invalid */
#define  GE_ECHO_SWITCH_INVAL          (2345)/* Echo switch invalid */
#define  GE_ECHO_AREA_RECT_INVAL       (2346)/* Echo area rectangle invalid */
#define  GE_SEL_TYPE_INVAL             (2347)/* Selection type invalid */
#define  GE_TYPE_RETURNED_VALUES_INVAL (2348)/* Type returned values invalid */
#define  GE_WS_GEN_TYPE_INVAL          (2349)/* Workstation generic type invalid */
#define  GE_COLR_MODEL_INVAL           (2350)/* Colour model invalid */
#define  GE_SEG_NAME_INVAL             (2351)/* Segment name invalid */
#define  GE_OLD_NEW_SEG_NAME_INVAL     (2352)/* Old or new segment name invalid */
#define  GE_SEG_ATTR_VALUE_INVAL       (2353)/* Segment attribute value invalid */
```

6.3 Miscellaneous

6.3.1 Linetypes

```
#define  GLINE_SOLID              (1)  /* Solid linetype */
#define  GLINE_DASH               (2)  /* Dashed linetype */
#define  GLINE_DOT                (3)  /* Dotted linetype */
#define  GLINE_DASH_DOT           (4)  /* Dashed-dotted linetype */
#define  GLINE_DASH_DOT_DOT       (5)  /* Dashed-dotted-dotted linetype */
```

6.3.2 Marker types

```
#define  GMARKER_DOT              (1)  /* Dotted marker type */
#define  GMARKER_PLUS             (2)  /* Plus (+) marker type */
#define  GMARKER_ASTERISK         (3)  /* Asterisk (*) marker type */
#define  GMARKER_CIRCLE           (4)  /* Circle (o) marker type */
#define  GMARKER_CROSS            (5)  /* Cross (X) marker type */
```

6.3.3 Hatch styles

```
#define  GHATCH_HOR               (1)  /* horizontal hatch lines */
#define  GHATCH_VERT              (2)  /* vertical hatch lines */
#define  GHATCH_DIAG_45           (3)  /* diagonal hatch lines 45 degree */
#define  GHATCH_DIAG_135          (4)  /* diagonal hatch lines 135 degree */
#define  GHATCH_HOR_VERT          (5)  /* hor. and vert. hatch lines */
#define  GHATCH_DIAG_45_135       (6)  /*  diag.  hatch  lines  45  and  135
                                       degree */
```

6.3.4 Colour models

```
#define  GMODEL_RGB               (1)  /* Red-Green-Blue */
#define  GMODEL_CIELUV            (2)  /* CIE Luv 1976 */
#define  GMODEL_HSV               (3)  /* Hue-Saturation-Value */
#define  GMODEL_HLS               (4)  /* Hue-Lightness-Saturation */
```

6.3.5 Prompt and echo types

```
#define  GLOC_DEF               (1)  /* Locator default */
#define  GLOC_CROSS_HAIR        (2)  /* Locator cross-hair */
#define  GLOC_TRACK_CROSS       (3)  /* Locator tracking cross */
#define  GLOC_RUB_BAND          (4)  /* Locator rubber band */
#define  GLOC_RECT              (5)  /* Locator rectangle */
#define  GLOC_DIGIT             (6)  /* Locator digital */

#define  GSTROKE_DEF            (1)  /* Stroke default */
#define  GSTROKE_DIGIT          (2)  /* Stroke digit */
#define  GSTROKE_MARKER         (3)  /* Stroke polymarker */
#define  GSTROKE_LINE           (4)  /* Stroke polyline */

#define  GVAL_DEF               (1)  /* Valuator default */
#define  GVAL_GRAPH             (2)  /* Valuator graphical */
#define  GVAL_DIGIT             (3)  /* Valuator digital */

#define  GCHOICE_DEF            (1)  /* Choice default */
#define  GCHOICE_PR_ECHO        (2)  /* Choice prompt and echo */
#define  GCHOICE_STRING_PR      (3)  /* Choice string and prompt */
#define  GCHOICE_STRING_IN      (4)  /* Choice string input */
#define  GCHOICE_SEG            (5)  /* Choice segment */

#define  GPICK_DEF              (1)  /* Pick default */
#define  GPICK_GROUP_HIGHL      (2)  /* Pick group highlighting */
#define  GPICK_SEG_HIGHL        (3)  /* Pick segment highlighting */
#define  GPICK_SET_NAMES_HIGHL (3)  /* Pick set of names highlighting */

#define  GSTRING_DEF            (1)  /* String default */
```

6.3.6 Default parameter of gopen_gks

```
#define  GDEF_ERR_FILE          (((char *)0))/* Default error file name */
```

7 C GKS function interface

7.1 Notational conventions

The binding of each GKS function follows the following template:

GKS Function Name

```
void gfunction(
     Gtype0   arg0,    /* argument 0 explanation   */
     Gtype1   arg1,    /* argument 1 explanation   */
     Gtype2   arg2     /* argument 2 explanation   */
);
```

"GKS Function Name" is the name of the function as listed in the GKS standard.

The C function name bound to the GKS function is **gfunction**. **arg0**, **arg1**, and **arg2** are the arguments to the function and correspond to the parameters of the GKS function definition. **Gtype0**, **Gtype1**, and **Gtype2** are the C data types of the arguments. The definitions of these types are listed in clause 5 of this part of ISO/IEC 8651.

To the right of each argument declaration is a C comment field which contains a brief explanation of the argument. If the comment begins with "OUT" it means the argument is used as an output parameter; the implementation returns data to the application through this argument. Arguments without "OUT" are input parameters.

All GKS/C functions have a **void** return type.

REMARK.

1) The C functions of 7.2 are C bindings of the functions of clause 12 of ISO/IEC 7942-1 (1994);

2) The C functions of 7.3 are C bindings of the functions of clause 13 of ISO/IEC 7942-1 (1994);

3) The C functions of 7.4 are C bindings of the functions of clause 14 of ISO/IEC 7942-1 (1994).

7.2 Workstation independent functions

7.2.1 Control functions

OPEN GKS

```
void gopen_gks(
     const char   *err_file,   /* name of error file                       */
     size_t       mem_units    /* number of units of memory available
                                  for buffer space                         */
);
```

REMARK.

1) If **err_file** has the value **GDEF_ERR_FILE**, the **stderr** stream is used.

2) The second parameter is unused.

CLOSE GKS

```
void gclose_gks(
     void
);
```

SAVE GKS STATE LIST

```
void gsave_gks_sl(
     const Ggks_sl_name_list    *entry_names,    /* GKS state list entry
                                                    names               */
     Gstore                     store,           /* handle to Store object   */
     Ggks_sl_entry_list         **entries        /* OUT GKS state
                                                    list entries */
);
```

REMARK. The memory referenced by `*entries` is managed by the parameter `store`.

RESTORE GKS STATE LIST

```
void grestore_gks_sl(
     const Ggks_sl_entry_list    *entries    /*  GKS state list entries */
);
```

ESCAPE

```
void gescape(
     Gint                      func_id,      /* escape function identifier   */
     const Gescape_in_data     *in_data,     /* escape input data record     */
     Gstore                    store,        /* handle to Store object       */
     Gescape_out_data          **out_data    /* OUT  escape output data
                                                record                       */
);
```

REMARK. The memory referenced by `*out_data` is managed by the parameter `store`.

EMERGENCY CLOSE GKS

```
void gemergency_close_gks(
     void
);
```

ERROR HANDLING

```
void gerr_hand(
     Gint          err_num,     /* error number                                */
     Gint          func_num,    /* number of function that detected the error  */
     const char    *err_file    /* name of error file                          */
);
```

ERROR LOGGING

```
void gerr_log(
     Gint         err_num,    /* error number                                 */
     Gint         func_num,   /* number of function that detected the error   */
     const char   *err_file   /* name of error file                           */
);
```

ROUTE

```
void groute(
     Groute_dir   dir   /* direction   */
);
```

7.2.2 Output functions

CREATE OUTPUT PRIMITIVE

```
void gcreate_out_prim(
     const Gprim_params   *prim_params   /* primitive parameters   */
);
```

[CREATE OUTPUT PRIMITIVE] POLYLINE

```
void gpolyline(
     const Gpoint_list   *point_list   /* list of points   */
);
```

[CREATE OUTPUT PRIMITIVE] POLYLINE SET

```
void gpolyline_set(
     const Gpoint_list_list   *point_list_list   /* list of point lists   */
);
```

[CREATE OUTPUT PRIMITIVE] NURB SET

```
void gnurb_set(
     const Gnurb_list   *nurb_set   /* NURB parameter lists   */
);
```

[CREATE OUTPUT PRIMITIVE] CONIC SECTION SET

```
void gconic_sec_set(
     const Gconic_sec_list   *conic_sec_set   /* Conic section set   */
);
```

[CREATE OUTPUT PRIMITIVE]
POLYMARKER

```
void gpolymarker(
     const Gpoint_list   *point_list   /* list of points   */
);
```

[CREATE OUTPUT PRIMITIVE]
FILL AREA

```
void gfill_area(
     const Gpoint_list   *point_list   /* list of points   */
);
```

[CREATE OUTPUT PRIMITIVE]
FILL AREA SET

```
void gfill_area_set(
     const Gpoint_list_list   *point_list_list  /* list of point lists   */
);
```

[CREATE OUTPUT PRIMITIVE]
CLOSED NURB SET

```
void gclosed_nurb_set(
     const Gnurb_list   *closed_nurb_set   /* NURB parameter lists   */
);
```

[CREATE OUTPUT PRIMITIVE]
ELLIPTIC SECTOR SET

```
void gell_sec_set(
     const Gconic_sec_list   *ell_sec_set   /* elliptic sector set   */
);
```

[CREATE OUTPUT PRIMITIVE]
ELLIPTIC SEGMENT SET

```
void gell_seg_set(
     const Gconic_sec_list   *ell_seg_set   /* elliptic segment set   */
);
```

[CREATE OUTPUT PRIMITIVE]
ELLIPTIC DISC SET

```
void gell_disc_set(
     const Gell_disc_list   *ell_disc_set   /* elliptic disc set   */
);
```

[CREATE OUTPUT PRIMITIVE]
TEXT

```
void gtext(
     const Gpoint   *text_pos,     /* text position      */
     const char     *char_string   /* character string   */
);
```

[CREATE OUTPUT PRIMITIVE]
CELL ARRAY

```
void gcell_array(
     const Grect       *rect,         /* cell rectangle   */
     const Gpat_rep    *colr_array    /* colour array     */
);
```

[CREATE OUTPUT PRIMITIVE]
DESIGN

```
void gdesign(
     const Gdesign    *design    /* design    */
);
```

[CREATE OUTPUT PRIMITIVE]
GENERALIZED DRAWING PRIMITIVE

```
void ggdp(
     const Gpoint_list    *point_list,    /* list of points    */
     Gint                 gdp_id,         /* gdp identifier    */
     const Ggdp_data      *gdp_data       /* gdp data record   */
);
```

7.2.3 Design output functions

OPEN STENCIL

```
void gopen_stencil(
     Gint            stencil_name,    /* stencil name   */
     Ginside_rule    inside_rule      /* inside rule    */
);
```

CLOSE STENCIL

```
void gclose_stencil(
     void
);
```

RENAME STENCIL

```
void grename_stencil(
     Gint    old_name,   /* old stencil name   */
     Gint    new_name    /* new stencil name   */
);
```

DELETE STENCIL

```
void gdel_stencil(
     Gint    name    /* stencil name   */
);
```

CREATE STENCIL FROM BOUNDARY

```
void gcreate_stencil_bndry(
     Gint                 stencil_name,    /* stencil name    */
     Ginside_rule         inside_rule,     /* inside rule     */
     const Gbndry_list    *bndry           /* boundary        */
);
```

CREATE STENCIL FROM CONTOURS

```
void gcreate_stencil_conts(
     Gint                stencil_name,    /* stencil name    */
     Ginside_rule        inside_rule,     /* inside rule     */
     const Gpath_list    *path_seq        /* path sequence   */
);
```

SET CONTOUR ATTRIBUTE

```
void gset_cont_attr(
     const Gcont_attr_value    *cont_attr    /* contour attribute   */
);
```

SET STENCIL ATTRIBUTE

```
void gset_stencil_attr(
     Gint                         stencil_name,    /* stencil name         */
     const Gstencil_attr_value    *stencil_attr    /* stencil attribute    */
);
```

GET STENCIL ATTRIBUTE

```
void gget_stencil_attr(
     Gint                   stencil_name,    /* stencil name             */
     Gstencil_attr_name     attr_name,       /* stencil attribute name   */
     Gint                   *err_ind,        /* OUT error indicator      */
     Gstencil_attr_value    *attr            /* OUT stencil attribute    */
);
```

INSTANCE STENCIL

```
void ginst_stencil(
     Gint                 stencil_name,   /* stencil name           */
     Gint                 inst_name,      /* instance name          */
     Gtran_matrix         stencil_tran,   /* stencil transformation */
     const Ginst_specif   *inst_specif    /* instance specifier     */
);
```

INSTANCE STENCIL ALONG PATH

```
void ginst_stencil_path(
     Gint            stencil_name,   /* stencil name       */
     Gint            inst_name,      /* instance name      */
     const Gpoint    *start_map,     /* start map          */
     const Gpoint    *end_map,       /* end map            */
     Gint            inst_num,       /* instance number    */
     const Gpath     *path_def       /* path definition    */
);
```

INSTANCE STENCIL SEQUENCE ALONG PATH

```
void ginst_stencil_seq_path(
     const Gseq_specif_list    *inst_seq_specif,   /* instance sequence
                                                      specifier              */
     const Gpath_specif        *path_specif,       /* path specifier         */
     Gtran_matrix              stencil_tran        /* stencil transformation */
);
```

OPEN TILING

```
void gopen_tiling(
     Gint   tiling_name   /* tiling name   */
);
```

CLOSE TILING

```
void gclose_tiling(
     void
);
```

RENAME TILING

```
void grename_tiling(
     Gint   old_name,   /* old tiling name   */
     Gint   new_name    /* new tiling name   */
);
```

DELETE TILING

```
void gdel_tiling(
     Gint   name   /* tiling name   */
);
```

CREATE TILING COMPONENT

```
void gcreate_tiling_comp(
     const Gtiling_params   *tiling_params,   /* tiling parameters      */
     const Gpoint           *origin,          /* origin                 */
     const Grepl_tech       *repl_tech        /* replication technique  */
);
```

7.2.4 Primitive attribute functions

SET PRIMITIVE ATTRIBUTE

```
void gset_prim_attr(
     const Gprim_attr_value   *attr_value   /* attr_value   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET PICK IDENTIFIER

```
void gset_pick_id(
     Gint   pick_id   /* pick identifier   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET NAMESET

```
void gset_nameset(
     const Gnameset   *nameset   /* nameset   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET SCISSOR SET

```
void gset_scissor_set(
     const Gscissor_list   *scissor_set   /* scissor_set   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET GLOBAL TRANSFORMATION MATRIX

```
void gset_global_tran_matrix(
     Gtran_matrix   global_tran_matrix   /* global transformation
                                            matrix                */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LOCAL TRANSFORMATION MATRIX

```
void gset_local_tran_matrix(
    Gtran_matrix   local_tran_matrix   /* local transformation
                                          matrix                */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET PATTERN SIZE

```
void gset_pat_size(
    const Gfloat_size   *pat_size   /* pattern size   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET PATTERN REFERENCE POINT

```
void gset_pat_ref_point(
    const Gpoint   *pat_ref_point   /* pattern reference point   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET CHARACTER HEIGHT

```
void gset_char_ht(
    Gfloat   char_ht   /* character height   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET CHARACTER UP VECTOR

```
void gset_char_up_vec(
    const Gvec   *char_up_vec   /* character up vector   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT SKEW ANGLE

```
void gset_text_skew_angle(
    Gfloat   text_skew_angle   /* text skew angle   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT PATH

```
void gset_text_path(
    Gtext_path   text_path   /* text path   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT ALIGNMENT

```
void gset_text_align(
     const Gtext_align   *text_align   /* text alignment   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LINE INDEX

```
void gset_line_ind(
     Gint   line_ind   /* line index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LINETYPE

```
void gset_linetype(
     Gint   linetype   /* linetype   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LINEWIDTH SCALE FACTOR

```
void gset_linewidth(
     Gfloat   linewidth   /* linewidth scale factor   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LINE COLOUR SPECIFIER

```
void gset_line_colr_specif(
     const Gcolr_specif   *line_colr_specif   /* line colour specifier   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET LINE COLOUR INDEX

```
void gset_line_colr_ind(
     Gint   line_colr_ind   /* line colour index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET MARKER INDEX

```
void gset_marker_ind(
     Gint   marker_ind   /* marker index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET MARKER TYPE

```
void gset_marker_type(
     Gint   marker_type   /* marker type   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET MARKER SIZE SCALE FACTOR

```
void gset_marker_size(
     Gfloat   marker_size   /* marker size scale factor   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET MARKER COLOUR SPECIFIER

```
void gset_marker_colr_specif(
     const Gcolr_specif   *marker_colr_specif   /* marker colour
                                                   specifier      */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET MARKER COLOUR INDEX

```
void gset_marker_colr_ind(
     Gint   marker_colr_ind   /* marker colour index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET AREA INDEX

```
void gset_area_ind(
     Gint   area_ind   /* area index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET INTERIOR STYLE

```
void gset_int_style(
     Gint_style   int_style   /* interior style   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET INTERIOR STYLE INDEX

```
void gset_int_style_ind(
     Gint   int_style_ind   /* interior style index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET INTERIOR COLOUR SPECIFIER

```
void gset_int_colr_specif(
     const Gcolr_specif   *int_colr_specif   /* interior colour specifier   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET INTERIOR COLOUR INDEX

```
void gset_int_colr_ind(
     Gint   int_colr_ind   /* interior colour index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET EDGE FLAG

```
void gset_edge_flag(
     Gedge_flag   edge_flag   /* edge flag   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET EDGETYPE

```
void gset_edgetype(
     Gint   edgetype   /* edgetype   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET EDGEWIDTH SCALE FACTOR

```
void gset_edgewidth(
     Gfloat   edgewidth   /* edgewidth scale factor   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET EDGE COLOUR SPECIFIER

```
void gset_edge_colr_specif(
     const Gcolr_specif   *edge_colr_specif   /* edge colour specifier   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET EDGE COLOUR INDEX

```
void gset_edge_colr_ind(
     Gint   edge_colr_ind   /* edge colour index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT INDEX

```
void gset_text_ind(
     Gint   text_ind   /* text index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT FONT AND PRECISION

```
void gset_text_font_prec(
     const Gtext_font_prec   *text_font_prec   /* text font and precision   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET CHARACTER EXPANSION FACTOR

```
void gset_char_expan(
     Gfloat   char_expan   /* character expansion factor   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET CHARACTER SPACING

```
void gset_char_space(
     Gfloat   char_space   /* character spacing   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT COLOUR SPECIFIER

```
void gset_text_colr_specif(
     const Gcolr_specif   *text_colr_specif   /* text colour specifier   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET TEXT COLOUR INDEX

```
void gset_text_colr_ind(
     Gint   text_colr_ind   /* text colour index   */
);
```

[SET PRIMITIVE ATTRIBUTE]
SET ATTRIBUTE SOURCE FLAGS

```
void gset_asfs(
     const Gasfs   *asfs   /* list of attribute source flags   */
);
```

ADD SET OF NAMES TO NAMESET

```
void gadd_set_names_nameset(
     const Gnameset   *set_names   /* set of names    */
);
```

REMOVE SET OF NAMES FROM NAMESET

```
void gremove_set_names_nameset(
     const Gnameset   *set_names   /* set of names    */
);
```

ADD SET OF SCISSORS TO SCISSOR SET

```
void gadd_set_scissors_scissor_set(
     const Gscissor_list   *set_scissors   /*set of scissors    */
);
```

REMOVE SET OF SCISSORS FROM SCISSOR SET

```
void gremove_set_scissors_scissor_set(
     const Gint_list   *scissor_ids   /* scissor identifiers    */
);
```

7.2.5 Normalization transformation functions

SET WINDOW AND VIEWPORT

```
void gset_win_vp(
     Gint           tran_num,    /* transformation number        */
     const Gtran    *win_vp      /* window and viewport limits   */
);
```

[SET WINDOW AND VIEWPORT]
SET WINDOW

```
void gset_win(
     Gint            tran_num,    /* transformation number    */
     const Glimit    *win         /* window limits            */
);
```

[SET WINDOW AND VIEWPORT]
SET VIEWPORT

```
void gset_vp(
     Gint            tran_num,    /* transformation number    */
     const Glimit    *vp          /* viewport limits          */
);
```

SET VIEWPORT INPUT PRIORITY

```
void gset_vp_in_pri(
     Gint       tran_num,       /* transformation number            */
     Gint       ref_tran_num,   /* reference transformation number  */
     Grel_pri   rel_pri         /* relative priority                */
);
```

SET SCISSOR MODE

```
void gset_scissor_mode(
     Gscissor_mode   scissor_mode   /* scissor mode    */
);
```

SET NORMALIZATION TRANSFORMATION NUMBER

```
void gset_norm_tran_num(
     Gint   tran_num   /* transformation number   */
);
```

[SET NORMALIZATION TRANSFORMATION NUMBER]
SELECT NORMALIZATION TRANSFORMATION

```
void gsel_norm_tran(
     Gint   tran_num   /* transformation number   */
);
```

7.2.6 NDC picture functions

DELETE PRIMITIVES FROM NDC PICTURE

```
void gdel_prims_ndc_pic(
     const Gsel_crit   *sel_crit   /* selection criterion   */
);
```

ADD SET OF NAMES TO NDC PICTURE

```
void gadd_set_names_ndc_pic(
     const Gnameset     *set_names,   /* set of names         */
     const Gsel_crit    *sel_crit     /* selection criterion  */
);
```

REMOVE SET OF NAMES FROM NDC PICTURE

```
void gremove_set_names_ndc_pic(
     const Gnameset     *set_names,   /* set of names         */
     const Gsel_crit    *sel_crit     /* selection criterion  */
);
```

SET NDC PICTURE PRIMITIVE ATTRIBUTE

```
void gset_ndc_pic_prim_attr(
     const Gsel_crit          *sel_crit,    /* selection criterion   */
     const Gprim_attr_value   *attr_value   /* attr_value            */
);
```

ADD SET OF SCISSORS TO NDC PICTURE

```
void gadd_set_scissors_ndc_pic(
     const Gscissor_list   *set_scissors,   /* scissor set           */
     const Gsel_crit       *sel_crit        /* selection criterion   */
);
```

REMOVE SET OF SCISSORS FROM NDC PICTURE

```
void gremove_set_scissors_ndc_pic(
     const Gint_list   *scissor_ids,   /* scissor identifiers   */
     const Gsel_crit   *sel_crit       /* selection criterion   */
);
```

REORDER NDC PICTURE

```
void greorder_ndc_pic(
     const Gsel_crit   *source_sel_crit,   /* source selection criterion      */
     const Gsel_crit   *ref_sel_crit,      /* reference selection criterion   */
     Grel_pos          rel_pos             /* relative position               */
);
```

7.2.7 Metafile functions

COPY NDC PICTURE TO NDC METAFILE

```
void gcopy_ndc_pic_ndc_mf(
     const void        *mf_specif,   /* metafile specifier    */
     Gint              pic_id,       /* picture identifier    */
     const Gsel_crit   *sel_crit,    /* selection criterion   */
     const Gnameset    *set_names    /* set of names          */
);
```

COPY NDC METAFILE PICTURE TO NDC PICTURE

```
void gcopy_ndc_mf_pic_ndc_pic(
     const void       *mf_specif,   /* metafile specifier   */
     Gint             pic_id,       /* picture identifier   */
     const Gnameset   *set_names    /* set of names         */
);
```

7.2.8 Picture part store functions

OPEN PICTURE PART

```
void gopen_pic_part(
     Gint    pic_part_name   /* picture part name   */
);
```

CLOSE PICTURE PART

```
void gclose_pic_part(
     void
);
```

ARCHIVE PICTURE PART

```
void gar_pic_part(
     Gint          pic_part_name,     /* picture part name              */
     const void    *ar_specif,        /* archive specifier              */
     Gint          ar_name_pic_part   /* archive name of picture part   */
);
```

RETRIEVE PICTURE PART FROM ARCHIVE

```
void gret_pic_part_ar(
     const void    *ar_specif,         /* archive specifier              */
     Gint          ar_name_pic_part,   /* archive name of picture part   */
     Gint          pic_part_name       /* picture part name              */
);
```

REOPEN PICTURE PART

```
void greopen_pic_part(
     Gint    pic_part_name   /* picture part name   */
);
```

APPEND PICTURE PART

```
void gappend_pic_part(
     Gint                source_pic_part_name,   /* source
                                                    picture part name      */
     Gint                sink_pic_part_name,     /* sink
                                                    picture part name      */
     const Gtran_matrix  global_tran_matrix,     /* global
                                                    transformation matrix  */
     Gtran_mode          global_tran_mode,       /* global
                                                    transformation mode    */
     const Gtran_matrix  local_tran_matrix,      /* local
                                                    transformation matrix  */
     Gtran_mode          local_tran_mode,        /* local
                                                    transformation mode    */
     const Gnameset      *set_names              /* set of names           */
);
```

RENAME PICTURE PART

```
void grename_pic_part(
     Gint   old_pic_part_name,    /* old picture part name   */
     Gint   new_pic_part_name     /* new picture part name   */
);
```

DELETE PICTURE PART

```
void gdel_pic_part(
     Gint   pic_part_name   /* picture part name   */
);
```

COPY PICTURE PART FROM PICTURE PART STORE

```
void gcopy_pic_part_pic_part_store(
     Gint                pic_part_name,        /* picture part name      */
     const Gsel_crit     *sel_crit,            /* selection criterion    */
     const Gtran_matrix  global_tran_matrix,   /* global
                                                  transformation matrix  */
     Gtran_mode          global_tran_mode,     /* global
                                                  transformation mode    */
     const Gtran_matrix  local_tran_matrix,    /* local
                                                  transformation matrix  */
     Gtran_mode          local_tran_mode,      /* local
                                                  transformation mode    */
     const Gnameset      *set_names,           /* set of names           */
     Gscissor_sel        scissor_sel           /* scissor select         */
);
```

COPY NDC PICTURE TO PICTURE PART STORE

```
void gcopy_ndc_pic_pic_part_store(
     const Gsel_crit   *sel_crit,       /* sel. criterion      */
     Gint              pic_part_name,   /* picture part name   */
     const Gnameset    *set_names       /* set of names        */
);
```

7.2.9 Input functions

SET LOGICAL INPUT DEVICE MODE

```
void gset_in_dev_mode(
     const Gdev_id    *dev_id,    /* input device number    */
     Gop_mode         op_mode     /* operating mode         */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET LOCATOR MODE

```
void gset_loc_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            loc_num,       /* locator device number    */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET STROKE MODE

```
void gset_stroke_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            stroke_num,    /* stroke device number     */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET VALUATOR MODE

```
void gset_val_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            val_num,       /* valuator device number   */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET CHOICE MODE

```
void gset_choice_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            choice_num,    /* choice device number     */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET PICK MODE

```
void gset_pick_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            pick_num,      /* pick device number       */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

[SET LOGICAL INPUT DEVICE MODE]
SET STRING MODE

```
void gset_string_mode(
     Gint            ws_id,         /* workstation identifier   */
     Gint            string_num,    /* string device number     */
     Gop_mode        op_mode,       /* operating mode           */
     Gecho_switch    echo_switch    /* echo switch              */
);
```

REQUEST INPUT

```
void greq_in(
     const Gdev_id    *dev_id,       /* input device number       */
     Gin_status       *in_status,    /* OUT input status          */
     Gin_value        *log_in_value  /* OUT logical input value   */
);
```

[REQUEST INPUT]
REQUEST LOCATOR

```
void greq_loc(
     Gint           ws_id,            /* workstation identifier   */
     Gint           loc_num,          /* locator device number    */
     Gin_status     *in_status,       /* OUT input status         */
     Gint           *norm_tran_num,   /* OUT normalization
                                         transformation number    */
     Gpoint         *loc_pos          /* OUT locator position     */
);
```

[REQUEST INPUT]
REQUEST STROKE

```
void greq_stroke(
     Gint            ws_id,             /* workstation identifier   */
     Gint            stroke_num,        /* stroke device number     */
     Gin_status      *in_status,        /* OUT input status         */
     Gint            *norm_tran_num,    /* OUT normalization
                                           transformation number    */
     Gpoint_list     *stroke            /* OUT stroke               */
);
```

REMARK. The application shall allocate the memory for the point list returned by this function. The maximum size of the returned **stroke** is specified by **ginit_stroke**. The maximum size of **stroke** supported by the implementation is returned by **ginq_def_stroke_data**.

[REQUEST INPUT]
REQUEST VALUATOR

```
void greq_val(
     Gint          ws_id,          /* workstation identifier  */
     Gint          val_num,        /* valuator device number  */
     Gin_status    *in_status,     /* OUT input status        */
     Gfloat        *value          /* OUT value               */
);
```

[REQUEST INPUT]
REQUEST CHOICE

```
void greq_choice(
     Gint          ws_id,          /* workstation identifier  */
     Gint          choice_num,     /* choice device number    */
     Gin_status    *in_status,     /* OUT input status        */
     Gint          *choice         /* OUT requested choice    */
);
```

[REQUEST INPUT]
REQUEST PICK

```
void greq_pick(
     Gint          ws_id,          /* workstation identifier    */
     Gint          pick_num,       /* pick device number        */
     Gin_status    *in_status,     /* OUT input status          */
     Gpick         *pick           /* OUT requested pick value  */
);
```

REMARK. The application shall allocate memory for the output parameter fields **pick->set_names.ws_ids.ints**, **pick->set_names.seg_names.ints**, **pick->set_names.names.ints**. If insufficient space has been allocated, an error message is issued.

[REQUEST INPUT]
REQUEST STRING

```
void greq_string(
     Gint        ws_id,         /* workstation identifier   */
     Gint        string_num,    /* string device number     */
     Gin_status  *in_status,    /* OUT input status         */
     char        *string        /* OUT requested string     */
);
```

REMARK. The application shall allocate the memory for the string returned by this function. The maximum size of the returned **string** is specified by **ginit_string.**

The maximum size of **string** supported by the implementation is returned by **ginq_def_string_data.**

SAMPLE INPUT

```
void gsample_in(
     const Gdev_id   *dev_id,        /* input device number       */
     Gin_value       *log_in_value   /* OUT logical input value   */
);
```

[SAMPLE INPUT]
SAMPLE LOCATOR

```
void gsample_loc(
     Gint     ws_id,          /* workstation identifier   */
     Gint     loc_num,        /* locator device number    */
     Gint     *norm_tran_num, /* OUT normalization
                                 transformation number    */
     Gpoint   *loc_pos        /* OUT locator position     */
);
```

[SAMPLE INPUT]
SAMPLE STROKE

```
void gsample_stroke(
     Gint          ws_id,           /* workstation identifier   */
     Gint          stroke_num,      /* stroke device number     */
     Gint          *norm_tran_num,  /* OUT normalization
                                       transformation number    */
     Gpoint_list   *stroke          /* OUT stroke               */
);
```

REMARK. The application shall allocate the memory for the point list returned by this function. The maximum size of the returned **stroke** is specified by **ginit_stroke.** The maximum size of **stroke** supported by the implementation is returned by **ginq_def_stroke_data.**

[SAMPLE INPUT]
SAMPLE VALUATOR

```
void gsample_val(
     Gint      ws_id,      /* workstation identifier   */
     Gint      val_num,    /* valuator device number   */
     Gfloat    *value      /* OUT value                */
);
```

[SAMPLE INPUT]
SAMPLE CHOICE

```
void gsample_choice(
     Gint         ws_id,         /* workstation identifier  */
     Gint         choice_num,    /* choice device number    */
     Gin_status   *in_status,    /* OUT input status        */
     Gint         *choice        /* OUT choice              */
);
```

[SAMPLE INPUT]
SAMPLE PICK

```
void gsample_pick(
     Gint         ws_id,         /* workstation identifier  */
     Gint         pick_num,      /* pick device number      */
     Gin_status   *in_status,    /* OUT input status        */
     Gpick        *pick          /* OUT pick value          */
);
```

REMARK. The application shall allocate memory for the output parameter fields **pick->set_names.ws_ids.ints**, **pick->set_names.seg_names.ints**, **pick->set_names.names.ints**. If insufficient space has been allocated, an error message is issued.

[SAMPLE INPUT]
SAMPLE STRING

```
void gsample_string(
     Gint   ws_id,         /* workstation identifier   */
     Gint   string_num,    /* string device number     */
     char   *string        /* OUT string               */
);
```

REMARK. The application shall allocate the memory for the string returned by this function. The maximum size of the returned **string** is specified by **ginit_string.** The maximum size of **string** supported by the implementation is returned by **ginq_def_string_data.**

AWAIT INPUT

```
void gawait_in(
     Gfloat                 timeout,            /* timeout (seconds)        */
     Gstore                 store,              /* handle to Store object   */
     Gdev_id_in_value_list  **dev_id_in_values  /* OUT logical
                                                  input devices and
                                                  logical input value      */
);
```

REMARK. The memory referenced by ***dev_id_in_values** is managed by the parameter **store**.

[AWAIT INPUT]
AWAIT EVENT

```
void gawait_event(
     Gfloat     timeout,   /* timeout (seconds)                */
     Gint       *ws_id,    /* OUT workstation identifier       */
     Gin_class  *class,    /* OUT device class                 */
     Gint       *in_num    /* OUT logical input device number  */
);
```

[AWAIT INPUT]
GET LOCATOR

```
void gget_loc(
     Gint     *norm_tran_num,   /* OUT normalization
                                   transformation number  */
     Gpoint   *loc_pos          /* OUT locator position   */
);
```

[AWAIT INPUT]
GET STROKE

```
void gget_stroke(
     Gint          *norm_tran_num,   /* OUT normalization
                                        transformation number  */
     Gpoint_list   *stroke           /* OUT stroke             */
);
```

REMARK. The application shall allocate the memory for the point list returned by this function. The maximum size of the returned **stroke** is specified by **ginit_stroke**. The maximum size of **stroke** supported by the implementation is returned by **ginq_def_stroke_data**.

[AWAIT INPUT]
GET VALUATOR

```
void gget_val(
     Gfloat   *value   /* OUT value   */
);
```

[AWAIT INPUT]
GET CHOICE

```
void gget_choice(
     Gin_status   *in_status,   /* OUT input status   */
     Gint         *choice       /* OUT choice         */
);
```

[AWAIT INPUT]
GET PICK

```
void gget_pick(
     Gin_status   *in_status,   /* OUT input status   */
     Gpick        *pick         /* OUT pick value     */
);
```

REMARK. The application shall allocate memory for the output parameter fields **pick->set_names.ws_ids.ints**, **pick->set_names.seg_names.ints**, **pick->set_names.names.ints**. If insufficient space has been allocated, an error message is issued.

[AWAIT INPUT]
GET STRING

```
void gget_string(
     char   *string   /* OUT string   */
);
```

REMARK. The application shall allocate the memory for the string returned by this function. The maximum size of the returned **string** is specified by **ginit_string**.

The maximum size of **string** supported by the implementation is returned by **ginq_def_string_data**.

FLUSH DEVICE EVENTS

```
void gflush_dev_events(
     Gdev_id   *dev_id   /* device identifier   */
);
```

7.2.10 Font and glyph functions

SET FONT INDEX MAPPING

```
void gset_font_ind_map(
     const Gfont_ind_map   *font_ind_map   /* font index mapping   */
);
```

GET GLYPH NAME

```
void gget_glyph_name(
     Gint    font_ind,      /* font index       */
     char    char_code,     /* character code   */
     Gint    *glyph_name    /* OUT glyph name   */
);
```

SET CHARACTER CODE

```
void gset_char_code(
     Gint    font_ind,     /* font index       */
     char    char_code,    /* character code   */
     Gint    glyph_name    /* glyph name       */
);
```

7.2.11 Audit and playback functions

AUDIT

```
void gaudit(
     Gint                 audit_id,    /* audit identifier   */
     const Gaudit_op      *audit_op    /* audit operation    */
);
```

WRITE USER RECORD TO AUDIT

```
void gwrite_user_rec_audit(
     Gint                        audit_id,     /* audit identifier   */
     const Gaudit_user_data      *user_data    /* user data record   */
);
```

PLAYBACK

```
void gplayback(
     Gint                    audit_id,       /* audit identifier     */
     const Gplayback_op      *playback_op    /* playback operation   */
);
```

READ ITEM FUNCTION NAME FROM AUDIT

```
void gread_item_func_name_audit(
     Gint   audit_id,    /* audit identifier    */
     Gint   *func_name   /* OUT function name   */
);
```

READ ITEM FROM AUDIT

```
void gread_item_audit(
     Gint           audit_id,        /* audit identifier          */
     Gstore         store,           /* handle to Store object    */
     Gfunc_params   **func_params    /* OUT function parameters   */
);
```

REMARK. The memory referenced by `*func_params` is managed by the parameter `store`.

PROCESS AUDIT ITEM

```
void gproc_audit_item(
     Gint       audit_id,   /* audit identifier   */
     Gproc_op   proc_op     /* process operation  */
);
```

7.2.12 Inquiry functions

INQUIRE OPERATING STATE ENTRY

```
void ginq_op_st_entry(
     Gop_st_name    op_st_name,     /* entry name              */
     Gint           *err_ind,       /* OUT error indicator     */
     Gop_st_entry   *op_st_entry    /* OUT operating state     */
);
```

INQUIRE GKS DESCRIPTION TABLE ENTRY

```
void ginq_gks_dt_entry(
     Ggks_dt_name    gks_dt_name,    /* entry name                   */
     Gstore          store,          /* handle to Store object       */
     Gint            *err_ind,       /* OUT error indicator          */
     Ggks_dt_entry   **gks_dt        /* OUT GKS
                                        description  table entry     */
);
```

REMARK. The memory referenced by `*gks_dt` is managed by the parameter `store`.

[INQUIRE GKS DESCRIPTION TABLE ENTRY]
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

```
void ginq_list_avail_ws_types(
     Gint        num_elems_appl_list,   /* length of application list        */
     Gint        start_ind,             /* starting index                    */
     Gint        *err_ind,              /* OUT error indicator               */
     Gint_list   *ws_gen_types,         /* OUT list of avalailable generic
                                           ws types                          */
     Gint        *num_elems_impl_list   /* OUT length of impl. list          */
);
```

[INQUIRE GKS DESCRIPTION TABLE ENTRY]
INQUIRE LIST OF AVAILABLE FONTS

```
void ginq_list_avail_fonts(
     Gint        num_elems_appl_list,   /* length of application list      */
     Gint        start_ind,             /* starting index                  */
     Gint        *err_ind,              /* OUT error indicator             */
     Gint_list   *fonts,                /* OUT list of avalailable fonts   */
     Gint        *num_elems_impl_list   /* OUT length of impl. list        */
);
```

[INQUIRE GKS DESCRIPTION TABLE ENTRY]
INQUIRE DEFAULT FONT INDEX MAPPING

```
void ginq_def_font_ind_map(
     Gint           *err_ind,        /* OUT error indicator      */
     Gfont_ind_map  *font_ind_map    /* OUT font index mapping   */
);
```

INQUIRE GKS STATE LIST ENTRY

```
void ginq_gks_sl_entry(
     Ggks_sl_name   gks_sl_name,     /* entry name               */
     Gstore         store,           /* handle to Store object   */
     Gint           *err_ind,        /* OUT error indicator      */
     Ggks_sl_entry  **gks_sl_entry   /* OUT GKS state
                                        list                     */
);
```

REMARK. The memory referenced by `*gks_sl_entry` is managed by the parameter `store`.

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE ROUTE DIRECTION

```
void ginq_route_dir(
     Gint         *err_ind,    /* OUT error indicator           */
     Groute_dir   *route_dir   /* OUT current route diretion    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SCISSOR MODE

```
void ginq_scissor_mode(
     Gint            *err_ind,       /* OUT error indicator        */
     Gscissor_mode   *scissor_mode   /* OUT current scissor mode   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF OPEN WORKSTATIONS

```
void ginq_set_open_wss(
     Gint        num_elems_appl_list,   /* length of application list   */
     Gint        start_ind,             /* starting index               */
     Gint        *err_ind,              /* OUT error indicator          */
     Gint_list   *open_ws,              /* OUT list of open ws ids      */
     Gint        *num_elems_impl_list   /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF ACTIVE WORKSTATIONS

```
void ginq_set_active_wss(
     Gint        num_elems_appl_list,   /* length of application list   */
     Gint        start_ind,             /* starting index               */
     Gint        *err_ind,              /* OUT error indicator          */
     Gint_list   *active_ws,            /* OUT list of active ws ids    */
     Gint        *num_elems_impl_list   /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF OPEN AUDITS

```
void ginq_set_open_audits(
     Gint               num_elems_appl_list,   /* length of application list   */
     Gint               start_ind,             /* starting index               */
     Gint               *err_ind,              /* OUT error indicator          */
     Gopen_audit_list   *open_audit_set,       /* OUT set of open audits       */
     Gint               *num_elems_impl_list   /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE SET OF OPEN PLAYBACKS

```
void ginq_set_open_playbacks(
     Gint        num_elems_appl_list,   /* length of application list   */
     Gint        start_ind,             /* starting index               */
     Gint        *err_ind,              /* OUT error indicator          */
     Gint_list   *open_playbacks,       /* OUT set of open playbacks    */
     Gint        *num_elems_impl_list   /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE INPUT QUEUE

```
void ginq_in_queue(
     Gstore      store,       /* handle to Store object  */
     Gint        *err_ind,    /* OUT error indicator     */
     Gin_queue   **in_queue   /* OUT input queue         */
);
```

REMARK. The memory referenced by `*in_queue` is managed by the parameter `store`.

[INQUIRE GKS STATE LIST ENTRY] INQUIRE FONT INDEX MAPPING

```
void ginq_font_ind_map(
     Gint           *err_ind,       /* OUT error indicator       */
     Gfont_ind_map  *font_ind_map   /* OUT font index mapping    */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE CURRENT PICK IDENTIFIER VALUE

```
void ginq_cur_pick_id(
     Gint   *err_ind,   /* OUT error indicator          */
     Gint   *pick_id    /* OUT current pick identifier  */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE NAMESET

```
void ginq_nameset(
     Gstore     store,      /* handle to Store object  */
     Gint       *err_ind,   /* OUT error indicator     */
     Gnameset   **nameset   /* nameset                 */
);
```

REMARK. The memory referenced by `*nameset` is managed by the parameter `store`.

[INQUIRE GKS STATE LIST ENTRY] INQUIRE SCISSOR SET

```
void ginq_scissor_set(
     Gstore          store,          /* handle to Store object    */
     Gint            *err_ind,       /* OUT error indicator       */
     Gscissor_list   **scissor_set   /* scissor                   */
);
```

REMARK. The memory referenced by `*scissor_set` is managed by the parameter `store`.

[INQUIRE GKS STATE LIST ENTRY] INQUIRE GLOBAL TRANSFORMATION MATRIX

```
void ginq_global_tran_matrix(
     Gtran_matrix   global_tran_matrix   /* global transformation
                                            matrix                  */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE LOCAL TRANSFORMATION MATRIX

```
void ginq_local_tran_matrix(
     Gtran_matrix   local_tran_matrix   /* local transformation
                                           matrix                  */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE PATTERN SIZE

```
void ginq_pat_size(
     Gint           *err_ind,    /* OUT error indicator          */
     Gfloat_size    *pat_size    /* OUT current pattern size     */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE PATTERN REFERENCE POINT

```
void ginq_pat_ref_point(
     Gint     *err_ind,        /* OUT error indicator                       */
     Gpoint   *pat_ref_point   /* OUT current pattern reference point      */
);
```

[INQUIRE GKS STATE LIST ENTRY] INQUIRE CHARACTER HEIGHT

```
void ginq_char_ht(
     Gint     *err_ind,   /* OUT error indicator            */
     Gfloat   *char_ht    /* OUT current character height   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE CHARACTER UP VECTOR

```
void ginq_char_up_vec(
     Gint    *err_ind,      /* OUT error indicator            */
     Gvec    *char_up_vec   /* OUT current character up vector  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT SKEW ANGLE

```
void ginq_text_skew_angle(
     Gint      *err_ind,          /* OUT error indicator            */
     Gfloat    *text_skew_angle   /* OUT current text skew angle    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT PATH

```
void ginq_text_path(
     Gint          *err_ind,    /* OUT error indicator      */
     Gtext_path    *text_path   /* OUT current text path    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT ALIGNMENT

```
void ginq_text_align(
     Gint           *err_ind,     /* OUT error indicator          */
     Gtext_align    *text_align   /* OUT current text alignment   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LINE INDEX

```
void ginq_line_ind(
     Gint    *err_ind,   /* OUT error indicator      */
     Gint    *line_ind   /* OUT current line index   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LINETYPE

```
void ginq_linetype(
     Gint    *err_ind,   /* OUT error indicator    */
     Gint    *linetype   /* OUT current linetype   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LINEWIDTH SCALE FACTOR

```
void ginq_linewidth(
     Gint     *err_ind,    /* OUT error indicator                 */
     Gfloat   *linewidth   /* OUT current linewidth scale factor  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LINE COLOUR SPECIFIER

```
void ginq_line_colr_specif(
     Gint           *err_ind,            /* OUT error indicator               */
     Gcolr_specif   *line_colr_specif    /* OUT current line colour specifier */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LINE COLOUR INDEX

```
void ginq_line_colr_ind(
     Gint   *err_ind,        /* OUT error indicator    */
     Gint   *line_colr_ind   /* OUT line colour index  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE MARKER INDEX

```
void ginq_marker_ind(
     Gint   *err_ind,     /* OUT error indicator         */
     Gint   *marker_ind   /* OUT current marker index    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE MARKER TYPE

```
void ginq_marker_type(
     Gint   *err_ind,      /* OUT error indicator         */
     Gint   *marker_type   /* OUT current marker type     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE MARKER SIZE SCALE FACTOR

```
void ginq_marker_size(
     Gint     *err_ind,      /* OUT error indicator                   */
     Gfloat   *marker_size   /* OUT current marker size scale factor  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE MARKER COLOUR SPECIFIER

```
void ginq_marker_colr_specif(
     Gint            *err_ind,              /* OUT error indicator   */
     Gcolr_specif    *marker_colr_specif    /* OUT current marker
                                               colour specifier      */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE MARKER COLOUR INDEX

```
void ginq_marker_colr_ind(
     Gint    *err_ind,            /* OUT error indicator        */
     Gint    *marker_colr_ind     /* OUT marker colour index    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE AREA INDEX

```
void ginq_area_ind(
     Gint    *err_ind,     /* OUT error indicator      */
     Gint    *area_ind     /* OUT current area index   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE INTERIOR STYLE

```
void ginq_int_style(
     Gint          *err_ind,      /* OUT error indicator            */
     Gint_style    *int_style     /* OUT current interior style     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE INTERIOR STYLE INDEX

```
void ginq_int_style_ind(
     Gint    *err_ind,          /* OUT error indicator                    */
     Gint    *int_style_ind     /* OUT current interior style index       */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE INTERIOR COLOUR SPECIFIER

```
void ginq_int_colr_specif(
     Gint            *err_ind,           /* OUT error indicator    */
     Gcolr_specif    *int_colr_specif    /* OUT current interior
                                            colour specifier       */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE INTERIOR COLOUR INDEX

```
void ginq_int_colr_ind(
     Gint   *err_ind,       /* OUT error indicator        */
     Gint   *int_colr_ind   /* OUT interior colour index  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE EDGE FLAG

```
void ginq_edge_flag(
     Gint         *err_ind,    /* OUT error indicator     */
     Gedge_flag   *edge_flag   /* OUT current edge flag   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE EDGETYPE

```
void ginq_edgetype(
     Gint   *err_ind,   /* OUT error indicator    */
     Gint   *edgetype   /* OUT current edgetype   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE EDGEWIDTH SCALE FACTOR

```
void ginq_edgewidth(
     Gint     *err_ind,    /* OUT error indicator                      */
     Gfloat   *edgewidth   /* OUT current edgewidth scale factor       */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE EDGE COLOUR SPECIFIER

```
void ginq_edge_colr_specif(
     Gint           *err_ind,           /* OUT error indicator                 */
     Gcolr_specif   *edge_colr_specif   /* OUT current edge colour specifier   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE EDGE COLOUR INDEX

```
void ginq_edge_colr_ind(
     Gint   *err_ind,        /* OUT error indicator    */
     Gint   *edge_colr_ind   /* OUT edge colour index  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT INDEX

```
void ginq_text_ind(
     Gint   *err_ind,   /* OUT error indicator      */
     Gint   *text_ind   /* OUT current text index   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT FONT AND PRECISION

```
void ginq_text_font_prec(
     Gint              *err_ind,     /* OUT error indicator                       */
     Gtext_font_prec   *font_prec    /* OUT current text font and precision       */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE CHARACTER EXPANSION FACTOR

```
void ginq_char_expan(
     Gint     *err_ind,     /* OUT error indicator                       */
     Gfloat   *char_expan   /* OUT current character expansion factor    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE CHARACTER SPACING

```
void ginq_char_space(
     Gint     *err_ind,     /* OUT error indicator              */
     Gfloat   *char_space   /* OUT current character spacing    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT COLOUR SPECIFIER

```
void ginq_text_colr_specif(
     Gint           *err_ind,            /* OUT error indicator                   */
     Gcolr_specif   *text_colr_specif    /* OUT current text colour specifier     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE TEXT COLOUR INDEX

```
void ginq_text_colr_ind(
     Gint   *err_ind,         /* OUT error indicator     */
     Gint   *text_colr_ind    /* OUT text colour index   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE ATTRIBUTE SOURCE FLAGS

```
void ginq_asfs(
    Gint     *err_ind,     /* OUT error indicator                 */
    Gasfs    *asfs         /* OUT current attribute source flags  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

```
void ginq_cur_norm_tran_num(
    Gint    *err_ind,        /* OUT error indicator        */
    Gint    *norm_tran_num   /* OUT current normalization
                                transformation number      */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LIST OF NORMALIZATION TRANSFORMATIONS

```
void ginq_list_norm_trans(
    Gint     *err_ind,        /* OUT error indicator        */
    Gtran    norm_tran[64]    /* OUT list of normalization
                                 transformations */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE NORMALIZATION TRANSFORMATION

```
void ginq_norm_tran(
    Gint     num,         /* normalization transformation number  */
    Gint     *err_ind,    /* OUT error indicator                  */
    Gtran    *norm_tran   /* OUT normalization transformation     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE LIST OF VIEWPORT INPUT PRIORITIES

```
void ginq_list_vp_in_pris(
    Gint    *err_ind,       /* OUT error indicator        */
    Gint    vp_in_pri[64]   /* OUT list of viewport input
                               priorities                 */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE NAME OF OPEN PICURE PART

```
void ginq_name_open_pic_part(
     Gint   *err_ind,            /* OUT  error indicator       */
     Gint   *name_open_pic_part  /* OUT name of open picture
                                    part                       */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF PICTURE PART NAMES IN USE

```
void ginq_set_pic_part_names(
     Gint        num_elems_appl_list,   /* length of application list      */
     Gint        start_ind,             /* starting index                  */
     Gint        *err_ind,              /* OUT error indicator             */
     Gint_list   *pic_part_names,       /* OUT list of picture part names  */
     Gint        *num_elems_impl_list   /* OUT length of impl. list        */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE NAME OF OPEN SEGMENT

```
void ginq_name_open_seg(
     Gint   *err_ind,        /* OUT  error indicator       */
     Gint   *name_open_seg   /* OUT name of open segment   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF SEGMENT NAMES IN USE

```
void ginq_set_seg_names(
     Gint        num_elems_appl_list,   /* length of application list   */
     Gint        start_ind,             /* starting index               */
     Gint        *err_ind,              /* OUT error indicator          */
     Gint_list   *seg_names,            /* OUT list of segment names    */
     Gint        *num_elems_impl_list   /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SEGMENT ATTRIBUTES

```
void ginq_seg_attrs(
     Gint         seg_name,     /* segment name              */
     Gint         *err_ind,     /* OUT error indicator       */
     Gseg_attrs   *seg_attrs    /* OUT segment attributes    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF ASSOCIATED WORKSTATIONS

```
void ginq_set_assoc_wss(
     Gint        seg_name,               /* segment name                 */
     Gint        num_elems_appl_list,    /* length of application list   */
     Gint        start_ind,              /* starting index               */
     Gint        *err_ind,               /* OUT error indicator          */
     Gint_list   *assoc_wss,             /* OUT list of
                                            asociated workstations       */
     Gint        *num_elems_impl_list    /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE NAME OF OPEN STENCIL

```
void ginq_name_open_stencil(
     Gint   *err_ind,            /* OUT  error indicator      */
     Gint   *name_open_stencil   /* OUT name of open stencil  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF STENCIL NAMES IN USE

```
void ginq_set_stencil_names(
     Gint        num_elems_appl_list,    /* length of application list   */
     Gint        start_ind,              /* starting index               */
     Gint        *err_ind,               /* OUT error indicator          */
     Gint_list   *stencil_names,         /* OUT list of stencil names    */
     Gint        *num_elems_impl_list    /* OUT length of impl. list     */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE STENCIL ATTRIBUTES

```
void ginq_stencil_attrs(
     Gint              stencil_name,     /* stencil name             */
     Gint              *err_ind,         /* OUT error indicator      */
     Gstencil_attrs    *stencil_attrs    /* OUT stencil attributes   */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE CURRENT CONTOUR ATTRIBUTES

```
void ginq_cur_cont_attrs(
     Gint           *err_ind,      /* OUT error indicator       */
     Gcont_attrs    *cont_attrs    /* OUT contour attributes    */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE NAME OF OPEN TILING

```
void ginq_name_open_tiling(
     Gint   *err_ind,            /* OUT  error indicator     */
     Gint   *name_open_tiling    /* OUT name of open tiling  */
);
```

[INQUIRE GKS STATE LIST ENTRY]
INQUIRE SET OF TILING NAMES IN USE

```
void ginq_set_tiling_names(
     Gint        num_elems_appl_list,   /* length of application list  */
     Gint        start_ind,             /* starting index              */
     Gint        *err_ind,              /* OUT error indicator         */
     Gint_list   *tiling_names,         /* OUT list of tiling names    */
     Gint        *num_elems_impl_list   /* OUT length of impl. list    */
);
```

INQUIRE INPUT QUEUE OVERFLOW

```
void ginq_in_overf(
     Gint        *err_ind,      /* OUT error indicator                */
     Gint        *ws_id,        /* OUT workstation identifier         */
     Gin_class   *meas_class,   /* OUT measure class                  */
     Gint        *in_num        /* OUT logical input device number    */
);
```

7.2.13 Utility functions

7.2.13.1 GKS Utilities

EVALUATE TRANSFORMATION MATRIX

```
void geval_tran_matrix(
     const Gpoint    *point,         /* fixed point                  */
     const Gvec      *shift,         /* shift vector                 */
     Gfloat          angle,          /* rotation angle               */
     const Gvec      *scale,         /* scale factors                */
     Gcoord_switch   coord_switch,   /* coordinate switch            */
     Gtran_matrix    tran_matrix     /* OUT transformation matrix    */
);
```

REMARK. The parameter `coord_switch` is unused.

ACCUMULATE TRANSFORMATION MATRIX

```
void gaccum_tran_matrix(
     const Gtran_matrix   matrix,          /* transformation matrix      */
     const Gpoint         *point,          /* fixed point                */
     const Gvec           *shift,          /* shift vector               */
     Gfloat               angle,           /* rotation angle             */
     const Gvec           *scale,          /* scale factors              */
     Gcoord_switch        coord_switch,    /* coordinate switch          */
     Gtran_matrix         tran_matrix      /* OUT transformation matrix  */
);
```

REMARK. The parameter `coord_switch` is unused.

EVALUATE CIRCLE

```
void geval_circle(
     const Gpoint     *circ_ctr,    /* circle centre     */
     Gfloat           radius,       /* radius            */
     Gconic_matrix    circle        /* OUT circle        */
);
```

EVALUATE ELLIPSE

```
void geval_ell(
     const Gpoint     *ell_ctr,                  /* ellipse centre       */
     const Gpoint     conj_radius_end_point[2],  /* conjugate
                                                    radius end points    */
     Gconic_matrix    ell                        /* OUT ellipse          */
);
```

EVALUATE CIRCULAR ARC 3 POINT

```
void geval_circ_arc_3_point(
     const Gpoint    circ_point[3],   /* 3 circle points
                                         (start, intermediate, end)   */
     Gconic_sec      *circ_arc        /* OUT circular arc             */
);
```

EVALUATE CIRCULAR ARC CENTRE

```
void geval_circ_arc_ctr(
     const Gpoint    *circ_ctr,     /* circle centre          */
     const Gvec      vec[2],        /* start and end vector   */
     Gsense_flag     sense_flag,    /* sense flag             */
     Gfloat          radius,        /* radius                 */
     Gconic_sec      *circ_arc      /* OUT circular arc       */
);
```

EVALUATE ELLIPTIC ARC

```
void geval_ell_arc(
     const Gpoint   *ell_ctr,                   /* ellipse centre         */
     const Gpoint   conj_radius_end_point[2],   /* conjugate
                                                   radius end points      */
     const Gvec     vec[2],                     /* start and end vector   */
     Gconic_sec     *ell_arc                    /* OUT elliptic arc       */
);
```

EVALUATE HYPERBOLIC ARC

```
void geval_hyp_arc(
     const Gpoint   *tran_radius_end_point,   /* transverse
                                                 radius end point       */
     const Gpoint   *hyp_ctr,                 /* hyperbola centre       */
     const Gpoint   *conj_radius_end_point,   /* conjugate
                                                 radius end point       */
     const Gvec     vec[2],                   /* start and end vector   */
     Gconic_sec     *hyp_arc                  /* OUT hyperbolic arc     */
);
```

EVALUATE PARABOLIC ARC

```
void geval_par_arc(
     const Gpoint   *start_point,       /* start point            */
     const Gpoint   *intersec_point,    /* tangential
                                           intersection point     */
     const Gpoint   *end_point,         /* end point              */
     Gconic_sec     *par_arc            /* OUT parabolic arc      */
);
```

EVALUATE WC ORDINATE

```
void geval_wc_ord(
     Gfloat     wc_ord,      /* WC ordinate          */
     Gord_sel   ord_sel,     /* ordinate selector    */
     Gfloat     *ndc_value   /* OUT NDC value        */
);
```

7.2.13.2 Binding Specific Utilities

SET ERROR HANDLING

```
void gset_err_hand(
     const void    (*new_hand)(Gint , Gint , const char *),
                   /* the application's error handling address */
     void          (**old_hand)(Gint , Gint , const char *)
                   /* OUT address of the error handling replaced */
);
```

Effect: See 3.11.

CREATE STORAGE

```
void gcreate_store(
     Gint     *err_ind,     /* OUT error indicator          */
     Gstore   *store        /* OUT handle to Store object   */
);
```

Effect: See 3.9.2

DELETE STORAGE

```
void gdel_store(
     Gint     *err_ind,     /* OUT error indicator              */
     Gstore   *store        /* IN/OUT storage to be deleted     */
);
```

Effect: See 3.9.2

7.3 Workstation functions

7.3.1 Control functions

OPEN WORKSTATION

```
void gopen_ws(
     Gint          ws_id,             /* workstation identifier     */
     const void    *ws_specif_info,   /* workstation specifier
                                         and specific information   */
     Gint          ws_gen_type        /* generic workstation type   */
);
```

CLOSE WORKSTATION

```
void gclose_ws(
     Gint    ws_id    /* workstation identifier   */
);
```

SAVE WORKSTATION STATE LIST

```
void gsave_ws_sl(
     Gint                       ws_id,          /* workstation identifier   */
     const Gws_sl_name_list     *entry_names,   /* list of entry
                                                   names                    */
     Gstore                     store,          /* handle to Store object   */
     Gws_sl_entry_list          **entry_list    /* OUT list of
                                                   entry values             */
);
```

REMARK. The memory referenced by `*entry_list` is managed by the parameter `store`.

RESTORE WORKSTATION STATE LIST

```
void grestore_ws_sl(
     Gint                        ws_id,      /* workstation identifier   */
     const Gws_sl_entry_list     *entries    /*  list of entries         */
);
```

GET WORKSTATION STATUS

```
void gget_ws_status(
     Gint          ws_id,        /* workstation identifier    */
     Gws_status    *ws_status    /*  OUT workstation status   */
);
```

REMOVE BACKDROP

```
void gremove_backdrop(
     Gint    ws_id    /* workstation identifier   */
);
```

SET REPRESENTATION

```
void gset_rep(
     Gint          ws_id,        /* workstation identifier   */
     const Grep    *rep_value    /* representation value     */
);
```

[SET REPRESENTATION]
SET LINE REPRESENTATION

```
void gset_line_rep(
     Gint                   ws_id,          /* workstation identifier   */
     Gint                   line_ind,       /* line index               */
     const Gline_bundle     *line_bundle    /* line representation      */
);
```

[SET REPRESENTATION]
SET MARKER REPRESENTATION

```
void gset_marker_rep(
     Gint                   ws_id,            /* workstation identifier   */
     Gint                   marker_ind,       /* marker index             */
     const Gmarker_bundle   *marker_bundle    /* marker representation    */
);
```

[SET REPRESENTATION]
SET TEXT REPRESENTATION

```
void gset_text_rep(
     Gint                 ws_id,          /* workstation identifier   */
     Gint                 text_ind,       /* text index               */
     const Gtext_bundle   *text_bundle    /* text representation      */
);
```

[SET REPRESENTATION]
SET AREA REPRESENTATION

```
void gset_area_rep(
     Gint                 ws_id,          /* workstation identifier   */
     Gint                 area_ind,       /* area index               */
     const Garea_bundle   *area_bundle    /* area representation      */
);
```

[SET REPRESENTATION]
SET PATTERN REPRESENTATION

```
void gset_pat_rep(
     Gint             ws_id,       /* workstation identifier    */
     Gint             pat_ind,     /* pattern index             */
     const Gpat_rep   *pat_rep     /* pattern representation    */
);
```

[SET REPRESENTATION]
SET COLOUR REPRESENTATION

```
void gset_colr_rep(
     Gint              ws_id,       /* workstation identifier   */
     Gint              colr_ind,    /* colour index             */
     const Gcolr_rep   *colr_rep    /* colour representation    */
);
```

SET VIEW PRIORITY

```
void gset_view_pri(
     Gint       ws_id,          /* workstation identifier   */
     Gint       view_ind,       /* view index               */
     Gint       ref_view_ind,   /* reference view index     */
     Grel_pri   rel_pri         /* relative priority        */
);
```

SET VIEW SELECTION CRITERION

```
void gset_view_sel_crit(
     Gint              ws_id,      /* workstation identifier  */
     Gint              view_ind,   /* view index              */
     const Gsel_crit   *sel_crit   /* selection criterion     */
);
```

SET VIEW

```
void gset_view(
     Gint                ws_id,        /* workstation identifier    */
     Gint                view_ind,     /* view index                */
     const Gtran_matrix  ori_matrix,   /* view orientation matrix   */
     const Gtran_matrix  map_matrix,   /* view mapping matrix       */
     const Gscissor      *scissor      /* view scissor              */
);
```

SET WORKSTATION VISUAL EFFECTS

```
void gset_ws_vis_effects(
     Gint               ws_id,            /* workstation identifier  */
     Gvis_effects_st    vis_effects_st    /* visual effects state    */
);
```

SET WORKSTATION WINDOW AND VIEWPORT

```
void gset_ws_win_vp(
     Gint           ws_id,        /* workstation identifier   */
     const Gtran    *ws_win_vp    /* workstation window
                                     and viewport limits      */
);
```

[SET WORKSTATION WINDOW AND VIEWPORT]
SET WORKSTATION WINDOW

```
void gset_ws_win(
     Gint           ws_id,     /* workstation identifier   */
     const Glimit   *ws_win    /* workstation window       */
);
```

[SET WORKSTATION WINDOW AND VIEWPORT]
SET WORKSTATION VIEWPORT

```
void gset_ws_vp(
     Gint           ws_id,   /* workstation identifier        */
     const Glimit   *ws_vp   /* workstation viewport limits   */
);
```

DEFINE LOGICAL INPUT DEVICE

```
void gdef_in_dev(
     const Gdev_id       *dev_id,       /* device identifier   */
     const Gint_list     *meas_seq,     /* measure sequence    */
     const Gint_list     *trigger_set,  /* trigger set         */
     const Ginit_value   *init_value    /* initial value       */
);
```

INITIALIZE LOGICAL INPUT DEVICE

```
void ginit_in_dev(
     const Gdev_id       *dev_id,     /* device identifier   */
     const Ginit_value   *init_value  /* initial value       */
);
```

[INITIALIZE LOGICAL INPUT DEVICE]
INITIALIZE LOCATOR

```
void ginit_loc(
     Gint              ws_id,               /* workstation identifier      */
     Gint              loc_num,             /* locator device number       */
     Gint              init_norm_tran_num,  /* initial normalization
                                               transformation number       */
     const Gpoint      *init_loc_pos,       /* initial locator position    */
     Gint              pet,                 /* prompt and echo type        */
     const Glimit      *echo_area,          /* echo area                   */
     const Gloc_data   *loc_data            /* locator data record         */
);
```

[INITIALIZE LOGICAL INPUT DEVICE] INITIALIZE STROKE

```
void ginit_stroke(
        Gint                  ws_id,                /* workstation identifier    */
        Gint                  stroke_num,           /* stroke device number      */
        Gint                  init_norm_tran_num,   /* initial normalization
                                                       transformation number     */
        const Gpoint_list     *init_stroke,         /* initial stroke            */
        Gint                  pet,                  /* prompt and echo type      */
        const Glimit          *echo_area,           /* echo area                 */
        const Gstroke_data    *stroke_data          /* stroke data record        */
);
```

[INITIALIZE LOGICAL INPUT DEVICE] INITIALIZE VALUATOR

```
void ginit_val(
        Gint               ws_id,         /* workstation identifier   */
        Gint               val_num,       /* valuator device number   */
        Gfloat             init_value,    /* initial value            */
        Gint               pet,           /* prompt and echo type     */
        const Glimit       *echo_area,    /* echo area                */
        const Gval_data    *val_data      /* valuator data record     */
);
```

[INITIALIZE LOGICAL INPUT DEVICE] INITIALIZE CHOICE

```
void ginit_choice(
        Gint                  ws_id,           /* workstation identifier    */
        Gint                  choice_num,      /* choice device number      */
        Gin_status            init_status,     /* initial status            */
        Gint                  init_choice,     /* initial choice            */
        Gint                  pet,             /* prompt and echo type      */
        const Glimit          *echo_area,      /* echo area                 */
        const Gchoice_data    *choice_data     /* choice data record        */
);
```

[INITIALIZE LOGICAL INPUT DEVICE]
INITIALIZE PICK

```
void ginit_pick(
     Gint               ws_id,         /* workstation identifier   */
     Gint               pick_num,      /* pick device number       */
     Gin_status         init_status,   /* initial status           */
     const Gpick        *init_pick,    /* initial pick value       */
     Gint               pet,           /* prompt and echo type     */
     const Glimit       *echo_area,    /* echo area                */
     const Gpick_data   *pick_data     /* pick data record         */
);
```

[INITIALIZE LOGICAL INPUT DEVICE]
INITIALIZE STRING

```
void ginit_string(
     Gint                 ws_id,           /* workstation identifier   */
     Gint                 string_num,      /* string device number     */
     const char           *init_string,    /* initial string           */
     Gint                 pet,             /* prompt and echo type     */
     const Glimit         *echo_area,      /* echo area                */
     const Gstring_data   *string_data     /* string data record       */
);
```

SET WORKSTATION SELECTION CRITERION

```
void gset_ws_sel_crit(
     Gint         ws_id,   /* workstation identifier         */
     const Gsel   *sel     /* selection type and criterion   */
);
```

COPY REALIZED PICTURE TO REALIZED METAFILE

```
void gcopy_real_pic_real_mf(
     Gint         ws_id,         /* workstation identifier   */
     const void   *mf_specif,    /* metafile specifier       */
     Gint         pic_id         /* picture identifier       */
);
```

COPY BLANK REALIZED PICTURE TO REALIZED METAFILE

```
void gcopy_blank_real_pic_real_mf(
     Gint         ws_id,         /* workstation identifier   */
     const void   *mf_specif,    /* metafile specifier       */
     Gint         pic_id         /* picture identifier       */
);
```

COPY REALIZED METAFILE PICTURE TO BACKDROP

```
void gcopy_real_mf_pic_backdrop(
     Gint          ws_id,         /* workstation identifier   */
     const void    *mf_specif,    /* metafile specifier       */
     Gint          pic_id         /* picture identifier       */
);
```

MESSAGE

```
void gmessage(
     Gint          ws_id,      /* workstation identifier   */
     const char    *message    /* message string           */
);
```

7.3.2 Inquiry functions

INQUIRE WORKSTATION STATE LIST ENTRY

```
void ginq_ws_sl_entry(
     Gint            ws_id,            /* workstation identifier    */
     Gws_sl_name     name,             /* entry name                */
     Ginq_type       type,             /* type of returned values   */
     Gstore          store,            /* handle to Store object    */
     Gint            *err_ind,         /* OUT error indicator       */
     Gws_sl_entry    **ws_sl_entry     /* OUT workstation
                                          state list entry          */
);
```

REMARK. The memory referenced by **`*ws_sl_entry`** is managed by the parameter **`store`**.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE WORKSTATION CONNECTION AND TYPE

```
void ginq_ws_conn_type(
     Gint     ws_id,        /* workstation identifier     */
     Gstore   store,        /* handle to Store object     */
     Gint     *err_ind,     /* OUT error indicator        */
     void     **conn_id,    /* OUT connection identifier  */
     Gint     *ws_type      /* OUT workstation type       */
);
```

REMARK. The memory referenced by **`*conn_id`** is managed by the parameter **`store`**.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE WORKSTATION STATE

```
void ginq_ws_st(
     Gint     ws_id,       /* workstation identifier   */
     Gint     *err_ind,    /* OUT error indicator      */
     Gws_st   *ws_st       /* OUT workstation state    */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE LIST OF LINE INDICES

```
void ginq_list_line_inds(
     Gint        ws_id,                  /* workstation identifier       */
     Gint        num_elems_appl_list,    /* length of application list   */
     Gint        start_ind,              /* starting index               */
     Gint        *err_ind,               /* OUT error indicator          */
     Gint_list   *def_line_inds,         /* OUT list of defined
                                            line indices                 */
     Gint        *num_elems_impl_list    /* OUT length of impl. list     */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE LINE REPRESENTATION

```
void ginq_line_rep(
     Gint           ws_id,       /* workstation identifier    */
     Gint           line_ind,    /* line index                */
     Ginq_type      type,        /* type of returned values   */
     Gint           *err_ind,    /* OUT error indicator       */
     Gline_bundle   *line_rep    /* OUT line representation   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE LIST OF MARKER INDICES

```
void ginq_list_marker_inds(
     Gint        ws_id,                  /* workstation identifier       */
     Gint        num_elems_appl_list,    /* length of application list   */
     Gint        start_ind,              /* starting index               */
     Gint        *err_ind,               /* OUT error indicator          */
     Gint_list   *def_marker_inds,       /* OUT list of defined
                                            marker indices               */
     Gint        *num_elems_impl_list    /* OUT length of impl. list     */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE MARKER REPRESENTATION

```
void ginq_marker_rep(
     Gint              ws_id,        /* workstation identifier      */
     Gint              marker_ind,   /* marker index                */
     Ginq_type         type,         /* type of returned values     */
     Gint              *err_ind,     /* OUT error indicator         */
     Gmarker_bundle    *marker_rep   /* OUT marker representation   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE LIST OF TEXT INDICES

```
void ginq_list_text_inds(
     Gint        ws_id,                /* workstation identifier       */
     Gint        num_elems_appl_list,  /* length of application list   */
     Gint        start_ind,            /* starting index               */
     Gint        *err_ind,             /* OUT error indicator          */
     Gint_list   *def_text_inds,       /* OUT list of defined
                                          text indices                 */
     Gint        *num_elems_impl_list  /* OUT length of impl. list     */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE TEXT REPRESENTATION

```
void ginq_text_rep(
     Gint            ws_id,       /* workstation identifier    */
     Gint            text_ind,    /* text index                */
     Ginq_type       type,        /* type of returned values   */
     Gint            *err_ind,    /* OUT error indicator       */
     Gtext_bundle    *text_rep    /* OUT text representation   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE LIST OF AREA INDICES

```
void ginq_list_area_inds(
     Gint        ws_id,                /* workstation identifier       */
     Gint        num_elems_appl_list,  /* length of application list   */
     Gint        start_ind,            /* starting index               */
     Gint        *err_ind,             /* OUT error indicator          */
     Gint_list   *def_area_inds,       /* OUT list of defined
                                          area indices                 */
     Gint        *num_elems_impl_list  /* OUT length of impl. list     */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE AREA REPRESENTATION

```
void ginq_area_rep(
    Gint           ws_id,      /* workstation identifier      */
    Gint           area_ind,   /* area index                  */
    Ginq_type      type,       /* type of returned values     */
    Gint           *err_ind,   /* OUT error indicator         */
    Garea_bundle   *area_rep   /* OUT area representation     */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE LIST OF PATTERN INDICES

```
void ginq_list_pat_inds(
    Gint        ws_id,                 /* workstation identifier          */
    Gint        num_elems_appl_list,   /* length of application list      */
    Gint        start_ind,             /* starting index                  */
    Gint        *err_ind,              /* OUT error indicator             */
    Gint_list   *def_pat_inds,         /* OUT list of defined
                                          pattern indices                 */
    Gint        *num_elems_impl_list   /* OUT length of impl. list        */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE PATTERN REPRESENTATION

```
void ginq_pat_rep(
    Gint        ws_id,      /* workstation identifier       */
    Gint        pat_ind,    /* pattern index                */
    Ginq_type   type,       /* type of returned values      */
    Gstore      store,      /* handle to Store object       */
    Gint        *err_ind,   /* OUT error indicator          */
    Gpat_rep    **pat_rep   /* OUT pattern representation   */
);
```

REMARK. The memory referenced by **`*pat_rep`** is managed by the parameter **`store`**.

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE LIST OF COLOUR INDICES

```
void ginq_list_colr_inds(
    Gint        ws_id,                 /* workstation identifier          */
    Gint        num_elems_appl_list,   /* length of application list      */
    Gint        start_ind,             /* starting index                  */
    Gint        *err_ind,              /* OUT error indicator             */
    Gint_list   *def_colr_inds,        /* OUT list of defined
                                          colour indices                  */
    Gint        *num_elems_impl_list   /* OUT length of impl. list        */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE COLOUR REPRESENTATION

```
void ginq_colr_rep(
     Gint        ws_id,      /* workstation identifier      */
     Gint        colr_ind,   /* colour index                */
     Ginq_type   type,       /* type of returned values     */
     Gint        *err_ind,   /* OUT error indicator         */
     Gcolr_rep   *colr_rep   /* OUT colour representation   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE WORKSTATION WINDOW AND VIEWPORT

```
void ginq_ws_win_vp(
     Gint    ws_id,       /* workstation identifier  */
     Gint    *err_ind,    /* OUT error indicator     */
     Gtran   *ws_win_vp   /* OUT workstation window
                             and viewport            */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE VISUAL EFFECTS STATE

```
void ginq_vis_effects_st(
     Gint             ws_id,             /* workstation identifier    */
     Gint             *err_ind,          /* OUT error indicator       */
     Gvis_effects_st  *vis_effects_st    /* OUT visual effects state  */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE TABLE OF SELECTION CRITERIA

```
void ginq_sel_table(
     Gint         ws_id,        /* workstation identifier            */
     Gstore       store,        /* handle to Store object            */
     Gint         *err_ind,     /* OUT error indicator               */
     Gsel_table   **sel_table   /* OUT table of selection criteria   */
);
```

REMARK. The memory referenced by `*sel_table` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

```
void ginq_set_seg_names_ws(
     Gint        ws_id,                /* workstation identifier     */
     Gint        num_elems_appl_list,  /* length of application list */
     Gint        start_ind,            /* starting index             */
     Gint        *err_ind,             /* OUT error indicator        */
     Gint_list   *seg_names,           /* OUT list of segment names  */
     Gint        *num_elems_impl_list  /* OUT length of impl. list   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE INPUT DEVICE OPERATING MODES AND INITIAL VALUES

```
void ginq_in_dev_op_modes_init_values(
     Gint                    ws_id,                  /* workstation identifier   */
     Gstore                  store,                  /* handle to Store object   */
     Gint                    *err_ind,               /* OUT error indicator      */
     Gop_modes_init_values   **op_modes_init_values  /* OUT table of
                                                        input device
                                                        operating modes and
                                                        initial values           */
);
```

REMARK. The memory referenced by `*in_table` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE VIEW REPRESENTATION

```
void ginq_view_rep(
     Gint        ws_id,        /* workstation identifier    */
     Gint        view_ind,     /* view index                */
     Ginq_type   type,         /* type of returned values   */
     Gstore      store,        /* handle to Store object    */
     Gint        *err_ind,     /* OUT error indicator       */
     Gview_rep   **view_rep    /* OUT view representation   */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE VIEW PRIORITIES

```
void ginq_view_pris(
     Gint        ws_id,          /* workstation identifier   */
     Ginq_type   type,           /* type of returned values  */
     Gint        *err_ind,       /* OUT error indicator      */
     Gint        view_pri[16]    /* OUT view priorities      */
);
```

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE LOCATOR DEVICE STATE

```
void ginq_loc_st(
     Gint           ws_id,                 /* workstation identifier          */
     Gint           loc_num,               /* locator device number           */
     Ginq_type      type,                  /* type of returned values         */
     Gstore         store,                 /* handle to Store object          */
     Gint           *err_ind,              /* OUT error indicator             */
     Gop_mode       *mode,                 /* OUT operating mode              */
     Gecho_switch   *esw,                  /* OUT echo switch                 */
     Gint           *init_norm_tran_num,   /* OUT initial normalization
                                              transformation number           */
     Gpoint         *init_loc_pos,         /* OUT initial locator position    */
     Gint           *pet,                  /* OUT prompt/echo type            */
     Glimit         *echo_area,            /* OUT echo area                   */
     Gloc_data      **loc_data             /* OUT locator data record         */
);
```

REMARK. The memory referenced by `*loc_data` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE STROKE DEVICE STATE

```
void ginq_stroke_st(
     Gint           ws_id,                 /* workstation identifier        */
     Gint           stroke_num,            /* stroke device number          */
     Ginq_type      type,                  /* type of returned values       */
     Gstore         store,                 /* handle to Store object        */
     Gint           *err_ind,              /* OUT error indicator           */
     Gop_mode       *mode,                 /* OUT operating mode            */
     Gecho_switch   *esw,                  /* OUT echo switch               */
     Gint           *init_norm_tran_num,   /* OUT initial normalization
                                              transformation number         */
     Gpoint_list    **init_stroke,         /* OUT initial stroke            */
     Gint           *pet,                  /* OUT prompt/echo type          */
     Glimit         *echo_area,            /* OUT echo area                 */
     Gstroke_data   **stroke_data          /* OUT stroke data record        */
);
```

REMARK. The memory referenced by `init_stroke` and `*stroke_data` is managed by `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE VALUATOR DEVICE STATE

```
void ginq_val_st(
    Gint            ws_id,           /* workstation identifier     */
    Gint            val_num,         /* valuator device number     */
    Gstore          store,           /* handle to Store object     */
    Gint            *err_ind,        /* OUT error indicator        */
    Gop_mode        *mode,           /* OUT operating mode         */
    Gecho_switch    *esw,            /* OUT echo switch            */
    Gfloat          *init_value,     /* OUT initial value          */
    Gint            *pet,            /* OUT prompt/echo type       */
    Glimit          *echo_area,      /* OUT echo area              */
    Gval_data       **val_data       /* OUT valuator data record   */
);
```

REMARK. The memory referenced by `*val_data` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY] INQUIRE CHOICE DEVICE STATE

```
void ginq_choice_st(
    Gint            ws_id,           /* workstation identifier       */
    Gint            choice_num,      /* choice device number         */
    Gstore          store,           /* handle to Store object       */
    Gint            *err_ind,        /* OUT error indicator          */
    Gop_mode        *mode,           /* OUT operating mode           */
    Gecho_switch    *esw,            /* OUT echo switch              */
    Gin_status      *init_status,    /* OUT initial choice status    */
    Gint            *init_choice,    /* OUT initial choice           */
    Gint            *pet,            /* OUT prompt/echo type         */
    Glimit          *echo_area,      /* OUT echo area                */
    Gchoice_data    **choice_data    /* OUT choice data record       */
);
```

REMARK. The memory referenced by `*choice_data` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE PICK DEVICE STATE

```
void ginq_pick_st(
     Gint            ws_id,          /* workstation identifier    */
     Gint            pick_num,       /* pick device number        */
     Ginq_type       type,           /* type of returned values   */
     Gstore          store,          /* handle to Store object    */
     Gint            *err_ind,       /* OUT error indicator       */
     Gop_mode        *mode,          /* OUT operating mode        */
     Gecho_switch    *esw,           /* OUT echo switch           */
     Gin_status      *init_status,   /* OUT initial pick status   */
     Gpick           *init_pick,     /* OUT initial pick value    */
     Gint            *pet,           /* OUT prompt/echo type      */
     Glimit          *echo_area,     /* OUT echo area             */
     Gpick_data      **pick_data     /* OUT pick data record      */
);
```

REMARK. The memory referenced by `*pick_data` is managed by the parameter `store`.

[INQUIRE WORKSTATION STATE LIST ENTRY]
INQUIRE STRING DEVICE STATE

```
void ginq_string_st(
     Gint            ws_id,           /* workstation identifier   */
     Gint            string_num,      /* string device number     */
     Gstore          store,           /* handle to Store object   */
     Gint            *err_ind,        /* OUT error indicator      */
     Gop_mode        *mode,           /* OUT operating mode       */
     Gecho_switch    *esw,            /* OUT echo switch          */
     char            **init_string,   /* OUT intial string        */
     Gint            *pet,            /* OUT prompt/echo type     */
     Glimit          *echo_area,      /* OUT echo area            */
     Gstring_data    **string_data    /* OUT string data record   */
);
```

REMARK. The memory referenced by `*init_string` and `*string_data` is managed by `store`.

INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY

```
void ginq_gen_ws_dt_entry(
     Gint            ws_gen_type,     /* workstation generic type      */
     Gws_dt_name     ws_dt_name,      /* workstation description table
                                         entry name                    */
     Gstore          store,           /* handle to Store object        */
     Gint            *err_ind,        /* OUT error indicator           */
     Gws_dt_entry    **ws_dt_entry    /* OUT workstation description
                                         table entry                   */
);
```

REMARK. The memory referenced by `*ws_dt_entry` is managed by the parameter `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE WORKSTATION CLASSIFICATION

```
void ginq_ws_class(
     Gint         ws_gen_type,    /* workstation generic type   */
     Gint         *err_ind,       /* OUT error indicator        */
     Gws_class    *class          /* OUT workstation class      */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DISPLAY SPACE SIZE

```
void ginq_disp_space_size(
     Gint               ws_gen_type,    /* workstation generic type  */
     Gint               *err_ind,       /* OUT error indicator       */
     Gdisp_space_size   *disp_size      /* OUT display [space] size  */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE LINE FACILITIES

```
void ginq_line_facs(
     Gint          ws_gen_type,            /* workstation generic type      */
     Gint          num_elems_appl_list,    /* length of application list    */
     Gint          start_ind,              /* starting index                */
     Gint          *err_ind,               /* OUT error indicator           */
     Gline_facs    *line_facs,             /* OUT line facilities           */
     Gint          *num_elems_impl_list    /* OUT length of linetype
                                              list in impl.                 */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY]
INQUIRE PREDEFINED LINE REPRESENTATION

```
void ginq_pred_line_rep(
     Gint            ws_gen_type,     /* workstation generic type   */
     Gint            ind,             /* predefined index           */
     Gint            *err_ind,        /* OUT error indicator        */
     Gline_bundle    *line_rep        /* OUT predefined line rep.   */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY]
INQUIRE MARKER FACILITIES

```
void ginq_marker_facs(
     Gint            ws_gen_type,          /* workstation generic type     */
     Gint            num_elems_appl_list,  /* length of application list   */
     Gint            start_ind,            /* starting index               */
     Gint            *err_ind,             /* OUT error indicator          */
     Gmarker_facs    *marker_facs,         /* OUT marker facilities        */
     Gint            *num_elems_impl_list  /* OUT length of marker type
                                              list in impl.                */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY]
INQUIRE PREDEFINED MARKER REPRESENTATION

```
void ginq_pred_marker_rep(
     Gint              ws_gen_type,     /* workstation generic type     */
     Gint              ind,             /* predefined index             */
     Gint              *err_ind,        /* OUT error indicator          */
     Gmarker_bundle    *marker_rep      /* OUT predefined marker rep.   */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY]
INQUIRE TEXT FACILITIES

```
void ginq_text_facs(
     Gint          ws_gen_type,          /* workstation generic type     */
     Gint          num_elems_appl_list,  /* length of application list   */
     Gint          start_ind,            /* starting index               */
     Gint          *err_ind,             /* OUT error indicator          */
     Gtext_facs    *text_facs,           /* OUT text facilities          */
     Gint          *num_elems_impl_list  /* OUT length of text font
                                            and precision list in impl.  */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE PREDEFINED TEXT REPRESENTATION

```
void ginq_pred_text_rep(
    Gint            ws_gen_type,    /* workstation generic type   */
    Gint            ind,            /* predefined index           */
    Gint            *err_ind,       /* OUT error indicator        */
    Gtext_bundle    *text_rep       /* OUT predefined text rep.   */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE AREA FACILITIES

```
void ginq_area_facs(
    Gint            ws_gen_type,    /* workstation generic type   */
    Gstore          store,          /* handle to Store object     */
    Gint            *err_ind,       /* OUT error indicator        */
    Garea_facs      **area_facs     /* OUT interior facilities    */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE PREDEFINED AREA REPRESENTATION

```
void ginq_pred_area_rep(
    Gint            ws_gen_type,    /* workstation generic type   */
    Gint            ind,            /* predefined index           */
    Gint            *err_ind,       /* OUT error indicator        */
    Garea_bundle    *area_rep       /* OUT predefined area rep.   */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE PATTERN FACILITIES

```
void ginq_pat_facs(
    Gint    ws_gen_type,      /* workstation generic type                */
    Gint    *err_ind,         /* OUT error indicator                     */
    Gint    *num_pred_inds    /* OUT num. of predef. pattern indices     */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE PREDEFINED PATTERN REPRESENTATION

```
void ginq_pred_pat_rep(
     Gint       ws_gen_type,   /* workstation generic type      */
     Gint       ind,           /* predefined index              */
     Gstore     store,         /* handle to Store object        */
     Gint       *err_ind,      /* OUT error indicator           */
     Gpat_rep   **pat_rep      /* OUT predefined pattern rep.   */
);
```

REMARK. The memory referenced by `*pat_rep` is managed by the parameter `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE COLOUR FACILITIES

```
void ginq_colr_facs(
     Gint         ws_gen_type,   /* workstation generic type   */
     Gint         *err_ind,      /* OUT error indicator        */
     Gcolr_facs   *colr_facs     /* OUT colour facilities      */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE PREDEFINED COLOUR REPRESENTATION

```
void ginq_pred_colr_rep(
     Gint        ws_gen_type,   /* workstation generic type     */
     Gint        ind,           /* predefined index             */
     Gint        *err_ind,      /* OUT error indicator          */
     Gcolr_rep   *colr_rep      /* OUT predefined colour rep.   */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

```
void ginq_list_avail_gdps(
     Gint        ws_gen_type,           /* workstation generic type    */
     Gint        num_elems_appl_list,   /* length of application list  */
     Gint        start_ind,             /* starting index              */
     Gint        *err_ind,              /* OUT error indicator         */
     Gint_list   *gdp,                  /* OUT list of GDPs            */
     Gint        *num_elems_impl_list   /* OUT length of impl. list    */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE SET OF AVAILABLE INPUT DEVICES

```
void ginq_set_avail_in_devs(
     Gint           ws_gen_type,          /* workstation generic type    */
     Gint           *err_ind,             /* OUT error indicator         */
     Gint           num_elems_appl_list,  /* length of application list  */
     Gdev_id_list   *dev_ids,             /* OUT set of input devices    */
     Gint           *num_elems_impl_list  /* OUT length of impl. list    */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE NUMBER OF AVAILABLE INPUT DEVICES

```
void ginq_num_avail_in(
     Gint      ws_gen_type,    /* workstation generic type     */
     Gint      *err_ind,       /* OUT error indicator          */
     Gnum_in   *num_in         /* OUT number of input devices  */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE INPUT DEVICE INITIAL VALUES

```
void ginq_in_dev_init_values(
     Gint                       ws_gen_type,          /* workstation generic type   */
     Gstore                     store,                /* handle to Store object     */
     Gint                       *err_ind,             /* OUT error indicator        */
     Gin_dev_init_value_list    **in_dev_init_values  /* OUT input device
                                                         initial values             */
);
```

REMARK. The memory referenced by `*in_dev_init_values` is managed by the parameter `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE SET OF AVAILABLE MEASURES

```
void ginq_set_avail_meass(
     Gint        ws_gen_type,          /* workstation generic type    */
     Gint        num_elems_appl_list,  /* length of application list  */
     Gint        *err_ind,             /* OUT error indicator         */
     Gint_list   *meas_ids,            /* OUT set of measure ids      */
     Gint        *num_elems_impl_list  /* OUT length of impl. list    */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE SET OF AVAILABLE TRIGGERS

```
void ginq_set_avail_triggers(
    Gint        ws_gen_type,          /* workstation generic type    */
    Gint        num_elems_appl_list,  /* length of application list  */
    Gint        *err_ind,             /* OUT error indicator         */
    Gint_list   *trigger_ids,         /* OUT set of trigger ids      */
    Gint        *num_elems_impl_list  /* OUT length of impl. list    */
);
```

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE INPUT DEVICE COMPOSITIONS

```
void ginq_in_dev_comps(
    Gint              ws_gen_type,   /* workstation generic type  */
    Gstore            store,         /* handle to Store object    */
    Gint              *err_ind,      /* OUT error indicator       */
    Gdev_comp_list    **dev_comps    /* OUT input device
                                        compositions              */
);
```

REMARK. The memory referenced by `*in_dev_comps` is managed by the parameter `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT LOCATOR DEVICE DATA

```
void ginq_def_loc_data(
    Gint        ws_gen_type,    /* workstation generic type         */
    Gint        loc_num,        /* logical input device number      */
    Gstore      store,          /* handle to Store object           */
    Gint        *err_ind,       /* OUT error indicator              */
    Gpoint      *loc_pos,       /* OUT default locator position     */
    Gint_list   **pet_list,     /* OUT list of prompt and echo types */
    Glimit      *echo_area,     /* OUT default echo area            */
    Gloc_data   **loc_data      /* OUT default data record          */
);
```

REMARK. The memory referenced by `*pet_list` and `*loc_data` is managed by `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT STROKE DEVICE DATA

```
void ginq_def_stroke_data(
    Gint           ws_gen_type,     /* workstation generic type         */
    Gint           stroke_num,      /* logical input device number      */
    Gstore         store,           /* handle to Store object           */
    Gint           *err_ind,        /* OUT error indicator              */
    Gint           *max_buf_size,   /* OUT max. input buffer size
                                          (nr. of points)               */
    Gint_list      **pet_list,      /* OUT list of prompt and echo types */
    Glimit         *echo_area,      /* OUT default echo area            */
    Gstroke_data   **stroke_data    /* OUT default data record          */
);
```

REMARK. The memory referenced by `*pet_list` and `*stroke_data` is managed by `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT VALUATOR DEVICE DATA

```
void ginq_def_val_data(
    Gint         ws_gen_type,   /* workstation generic type          */
    Gint         val_num,       /* logical input device number       */
    Gstore       store,         /* handle to Store object            */
    Gint         *err_ind,      /* OUT error indicator               */
    Gfloat       *def_val,      /* OUT default initial value         */
    Gint_list    **pet_list,    /* OUT list of prompt and echo types */
    Glimit       *echo_area,    /* OUT default echo area             */
    Gval_data    **val_data     /* OUT default data record           */
);
```

REMARK. The memory referenced by `*pet_list` and `*val_data` is managed by `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT CHOICE DEVICE DATA

```
void ginq_def_choice_data(
    Gint            ws_gen_type,        /* workstation generic type          */
    Gint            choice_num,         /* logical input device number       */
    Gstore          store,              /* handle to Store object            */
    Gint            *err_ind,           /* OUT error indicator               */
    Gint            *max_num_choices,   /* OUT max. num. of choices          */
    Gint_list       **pet_list,         /* OUT list of prompt and echo types */
    Glimit          *echo_area,         /* OUT default echo area             */
    Gchoice_data    **choice_data       /* OUT default data record           */
);
```

REMARK. The memory referenced by `*pet_list` and `*choice_data` is managed by `store`.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT PICK DEVICE DATA

```
void ginq_def_pick_data(
     Gint          ws_gen_type,    /* workstation generic type          */
     Gint          pick_num,       /* logical input device number       */
     Gstore        store,          /* handle to Store object            */
     Gint          *err_ind,       /* OUT error indicator               */
     Gint_list     **pet_list,     /* OUT list of prompt and echo types */
     Glimit        *echo_area,     /* OUT default echo area             */
     Gpick_data    **pick_data     /* OUT default data record           */
);
```

REMARK. The memory referenced by **`*pet_list`** and **`*pick_data`** is managed by **`store`**.

[INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE ENTRY] INQUIRE DEFAULT STRING DEVICE DATA

```
void ginq_def_string_data(
     Gint            ws_gen_type,      /* workstation generic type          */
     Gint            string_num,       /* logical input device number       */
     Gstore          store,            /* handle to Store object            */
     Gint            *err_ind,         /* OUT error indicator               */
     Gint            *max_buf_size,    /* OUT max. input buffer size
                                              (nr. of bytes)                */
     Gint_list       **pet_list,       /* OUT list of prompt and echo types */
     Glimit          *echo_area,       /* OUT default echo area             */
     Gstring_data    **string_data     /* OUT default data record           */
);
```

REMARK. The memory referenced by **`*pet_list`** and **`*string_data`** is managed by **`store`**.

INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY

```
void ginq_specif_ws_dt_entry(
     Gint            ws_id,           /* workstation identifier          */
     Gws_dt_name     ws_dt_name,      /* workstation description table
                                         entry name                      */
     Gstore          store,           /* handle to Store object          */
     Gint            *err_ind,        /* OUT error indicator             */
     Gws_dt_entry    **ws_dt_entry    /* OUT workstation description
                                         table entry                     */
);
```

REMARK. The memory referenced by **`*ws_dt_entry`** is managed by the parameter **`store`**.

INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE

```
void ginq_specif_ws_dt_entry_st(
    Gint            ws_id,          /* workstation identifier   */
    Gws_dt_name     ws_dt_name,     /* entry name               */
    Gint            *err_ind,       /* OUT error indicator      */
    Gchange_flag    *entry_st       /* OUT  entry state         */
);
```

RESET SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY STATE

```
void greset_specif_ws_dt_entry_st(
    Gint            ws_id,        /* workstation identifier   */
    Gws_dt_name     ws_dt_name    /* entry name               */
);
```

7.3.3 Retrieval functions

GET TEXT EXTENT

```
void gget_text_extent(
    Gint            ws_id,        /* workstation identifier         */
    const Gpoint    *pos,         /* text position                  */
    const char      *str,         /* character string               */
    Gint            *err_ind,     /* OUT error indicator            */
    Gtext_extent    *extent       /* OUT concatenation point and
                                     text extent parallelogram      */
);
```

[GET TEXT EXTENT]
INQUIRE TEXT EXTENT

```
void ginq_text_extent(
    Gint            ws_id,        /* workstation identifier         */
    const Gpoint    *pos,         /* text position                  */
    const char      *str,         /* character string               */
    Gint            *err_ind,     /* OUT error indicator            */
    Gtext_extent    *extent       /* OUT concatenation point and
                                     text extent parallelogram      */
);
```

7.3.4 Viewing utility functions

EVALUATE VIEW ORIENTATION MATRIX

```
void geval_view_ori_matrix(
     const Gpoint   *ref_point,      /* fixed reference point   */
     const Gvec     *view_up,        /* view up vector          */
     Gtran_matrix   view_ori_matrix  /* OUT view orientation
                                        matrix                  */
);
```

EVALUATE VIEW MAPPING MATRIX

```
void geval_view_map_matrix(
     const Gtran    *win_vp,         /* window/viewport limits    */
     Gtran_matrix   view_map_matrix  /* OUT view orientation
                                        matrix                    */
);
```

7.3.5 Colour utility functions

CONVERT COLOUR

```
void gconv_colr(
     Gint                ws_id,         /* workstation identifier   */
     const Gcolr_specif  *colr_specif,  /* colour specifier         */
     Gint                colr_model,    /* colour model             */
     Gcolr_specif        *colr_coords,  /* OUT colour coordinates   */
     Gconv_flag          *conv_flag     /* OUT conversion flag      */
);
```

7.4 Segment functions and workstation activation functions

REMARK. The segment facilties in ISO/IEC 7942- 1:1994 are identified as obsolescent features. It is anticipated that they will be considered for deletion for the next edition of ISO/IEC 7942-1. Consequently, the bindings of these functions, documented in this subclause, will be considered for deletion as well at the next edition of this part of ISO/IEC 8651.

7.4.1 Segment functions

CREATE SEGMENT

```
void gcreate_seg(
     Gint    seg_name   /* segment name   */
);
```

CLOSE SEGMENT

```
void gclose_seg(
     void
);
```

RENAME SEGMENT

```
void grename_seg(
     Gint   old_seg_name,    /* old segment name   */
     Gint   new_seg_name     /* new segment name   */
);
```

DELETE SEGMENT

```
void gdel_seg(
     Gint   seg_name    /* segment name   */
);
```

DELETE SEGMENT FROM WORKSTATION

```
void gdel_seg_ws(
     Gint   ws_id,      /* workstation identifier   */
     Gint   seg_name    /* segment name             */
);
```

ASSOCIATE SEGMENT WITH WORKSTATION

```
void gassoc_seg_ws(
     Gint   ws_id,      /* workstation identifier   */
     Gint   seg_name    /* segment name             */
);
```

COPY SEGMENT TO WORKSTATION

```
void gcopy_seg_ws(
     Gint   ws_id,      /* workstation identifier   */
     Gint   seg_name    /* segment name             */
);
```

INSERT SEGMENT

```
void ginsert_seg(
     Gint                 seg_name,     /* segment name             */
     const Gtran_matrix   tran_matrix   /* transformation matrix    */
);
```

SET SEGMENT ATTRIBUTE

```
void gset_seg_attr(
     Gint                    seg_name,          /* segment name              */
     const Gseg_attr_value   *seg_attr_value    /* segment attribute value   */
);
```

[SET SEGMENT ATTRIBUTE]
SET SEGMENT TRANSFORMATION

```
void gset_seg_tran(
     Gint           seg_name,     /* segment name              */
     Gtran_matrix   tran_matrix   /* transformation matrix     */
);
```

[SET SEGMENT ATTRIBUTE]
SET VISIBILITY

```
void gset_vis(
     Gint    seg_name,     /* segment name   */
     Gvis    vis           /* visibility     */
);
```

[SET SEGMENT ATTRIBUTE]
SET HIGHLIGHTING

```
void gset_highl(
     Gint      seg_name,    /* segment name   */
     Ghighl    highl        /* highlighting   */
);
```

[SET SEGMENT ATTRIBUTE]
SET DETECTABILITY

```
void gset_det(
     Gint    seg_name,     /* segment name    */
     Gdet    det           /* detectability   */
);
```

[SET SEGMENT ATTRIBUTE]
SET SEGMENT PRIORITY

```
void gset_seg_pri(
     Gint      seg_name,    /* segment name       */
     Gfloat    seg_pri      /* segment priority   */
);
```

7.4.2 Workstation activation functions

ACTIVATE WORKSTATION

```
void gactivate_ws(
     Gint   ws_id   /* workstation identifier   */
);
```

DEACTIVATE WORKSTATION

```
void gdeactivate_ws(
     Gint   ws_id   /* workstation identifier   */
);
```

CLEAR WORKSTATION

```
void gclear_ws(
     Gint         ws_id,      /* workstation identifier    */
     Gctrl_flag   ctrl_flag   /* control flag              */
);
```

REMARK. The second parameter is unused.

7.4.3 Utility functions

GET MAPPED SEGMENT NAME

```
void gget_map_seg_name(
     Gint   seg_name,        /* segment name              */
     Gint   *map_seg_name    /* OUT mapped segment name   */
);
```

GET MAPPED WORKSTATION IDENTIFIER

```
void gget_map_ws_id(
     Gint   ws_id,        /* workstation identifier            */
     Gint   *map_ws_id    /* OUT mapped workstation identifier */
);
```

Annex A
(informative)

Compiled GKS/C specification

This annex contains a print of the header **gks.h**. This header should be included in any GKS/C application program and contains the following information:

1) The data types from clause 5 in compilation order.
2) The macros from clause 6.
3) The specification of the GKS functions as **extern void**.
4) The header **gks_compat.h;** see annex E for a description of this header.
5) Additional implementation-dependent items; they should preferably have the sentinels "g" or "G".

This annex only contains the parts 1)-4).

A.1 Data types in compilation order

```
/* Data Types in Compilation Order*/
/* Basic Types*/
/* Gfloat  floating point number */
typedef     float Gfloat;
/* Gint  integer */
typedef     int         Gint;
/* Gstore store */
typedef void *Gstore;
/* Gtran_matrix transformation matrix */
typedef Gfloat Gtran_matrix[2][3];
/* Gconic_matrix conic matrix */
typedef Gfloat Gconic_matrix[3][3];
/* Gdata data */
typedef struct {
 size_t     size; /* size of data*/
 void       *data;      /* pointer to data*/
} Gdata;
/* Enumeration Types*/
/* Gabut_specif ABUT specifier */
typedef enum {
 GABUT_RIGHT,
 GABUT_LEFT,
 GABUT_TOP,
 GABUT_BOTTOM,
 GABUT_CTR_VERT,
 GABUT_CTR_HOR,
 GABUT_CAP_LINE,
 GABUT_BASE_LINE
} Gabut_specif;
/* Gasf attribute source flag */
typedef enum {
 GASF_BUNDLED,
 GASF_INDIV
} Gasf;
/* Gattr_ctrl_flag attribute control flag */
```

```
typedef enum {
 GFLAG_CUR,
 GFLAG_SPECIF
} Gattr_ctrl_flag;
/* Gprim_attr_name primitive attribute name */
typedef enum {
 GNAME_PICK_ID,   /* pick identifier*/
 GNAME_NAMESET,   /* nameset  */
 GNAME_SCISSOR_SET,     /* scissor set*/
 GNAME_GLOBAL_TRAN_MATRIX,     /* global transformation matrix */
 GNAME_LOCAL_TRAN_MATRIX,      /* local transformation matrix */
 GNAME_PAT_SIZE,  /* pattern size*/
 GNAME_PAT_REF_POINT,    /* pattern reference point*/
 GNAME_CHAR_HT,   /* character height*/
 GNAME_CHAR_UP_VEC,      /* character up vector*/
 GNAME_TEXT_SKEW_ANGLE, /* text skew angle*/
 GNAME_TEXT_PATH, /* text path*/
 GNAME_TEXT_ALIGN,       /* text alignment*/
 GNAME_LINE_IND,  /* line index*/
 GNAME_LINETYPE,  /* linetype */
 GNAME_LINEWIDTH, /* linewidth scale factor*/
 GNAME_LINE_COLR_SPECIF,       /* line colour specifier*/
 GNAME_MARKER_IND,       /* marker index*/
 GNAME_MARKER_TYPE,      /* marker type*/
 GNAME_MARKER_SIZE,      /* marker size scale factor*/
 GNAME_MARKER_COLR_SPECIF,     /* marker colour specifier*/
 GNAME_AREA_IND,  /* area index*/
 GNAME_INT_STYLE, /* interior style*/
 GNAME_INT_STYLE_IND,    /* interior style index*/
 GNAME_INT_COLR_SPECIF, /* interior colour specifier*/
 GNAME_EDGE_FLAG, /* edge flag */
 GNAME_EDGETYPE,  /* edgetype */
 GNAME_EDGEWIDTH, /* edgewidth scale factor */
 GNAME_EDGE_COLR_SPECIF,       /* edge colour specifier */
 GNAME_TEXT_IND,  /* text index*/
 GNAME_TEXT_FONT_PREC,   /* text font and precision*/
 GNAME_CHAR_EXPAN,       /* character expansion factor*/
 GNAME_CHAR_SPACE,       /* character spacing*/
 GNAME_TEXT_COLR_SPECIF,       /* text colour specifier*/
 GNAME_ASFS /* asfs      */
} Gprim_attr_name;
/* Gaudit_flag audit flag */
typedef enum {
 GAUDIT_OFF,
 GAUDIT_ON
} Gaudit_flag;
/* Gaudit_op audit operation */
typedef struct {
  enum Gaudit_op_type {
                GAUDIT_OPEN, GAUDIT_CLOSE, GAUDIT_BEGIN, GAUDIT_END
  } type;
  void       *specif;    /* audit specifier*/
```

```
} Gaudit_op;
/* Gcap cap */
typedef enum {
 GCAP_BUT,
 GCAP_ROUND,
 GCAP_SQUARE
} Gcap;
/* Gchange_flag change flag */
typedef enum {
 GFLAG_CHANGED,
 GFLAG_NOT_CHANGED
} Gchange_flag;
/* Gconv_flag conversion flag */
typedef enum {
 GCONV_DONE,
 GCONV_APPROX,
 GCONV_NOT_CONV
} Gconv_flag;
/* Gchoice_status choice status */
typedef enum {
 GCHOICE_OK,
 GCHOICE_NO_CHOICE
} Gchoice_status;
/* Gclip_ind clipping indicator */
typedef enum {
 GIND_NO_CLIP,
 GIND_CLIP
} Gclip_ind;
/* Gclosed_path_op_st closed path operating state */
typedef enum {
 GCP_ST_CPCL,
 GCP_ST_CPOP
} Gclosed_path_op_st;
/* Gcolr_avail colour available */
typedef enum {
 GAVAIL_MONOCHR,
 GAVAIL_COLR
} Gcolr_avail;
/* Gcolr_kind colour kind */
typedef enum {
 GCOLR_INDIR,
 GCOLR_RGB,
 GCOLR_CIELUV,
 GCOLR_HLS,
 GCOLR_HSV
} Gcolr_kind;
/* Gcont_attr_name contour attribute name */
typedef enum {
 GNAME_STYLE,
 GNAME_WIDTH,
 GNAME_CAP,
 GNAME_JOIN
```

```
} Gcont_attr_name;
/* Gcoord_switch coordinate switch */
typedef enum {
 GCOORD_WC,
 GCOORD_NDC
} Gcoord_switch;
/* Gctrl_flag control flag */
typedef enum {
 GFLAG_COND,
 GFLAG_ALWAYS
} Gctrl_flag;
/* Gdc_units device coordinate units */
typedef enum {
 GDC_METRES,
 GDC_OTHER
} Gdc_units;
/* Gdet detectability */
typedef enum {
 GSEG_UNDET,
 GSEG_DET
} Gdet;
/* Groute_dir direction */
typedef enum {
 GDIR_NDC,
 GDIR_BACKDROP
} Groute_dir;
/* Gecho_switch echo switch */
typedef enum {
 GSWITCH_NO_ECHO,
 GSWITCH_ECHO
} Gecho_switch;
/* Gedge_flag edge flag */
typedef enum {
 GEDGE_OFF,
 GEDGE_ON
} Gedge_flag;
/* Ggks_dt_name GKS description table name */
typedef enum {
 GGKS_DT_AVAIL_WS_GEN_TYPES,  /* list of available ws. types*/
 GGKS_DT_AVAIL_FONTS,    /* list of available fonts*/
 GGKS_DT_DEF_FONT_IND_MAP     /* default font index mapping*/
} Ggks_dt_name;
/* Ggks_op_st GKS operating state */
typedef enum {
 GOP_ST_GKCL,
 GOP_ST_GKOP
} Ggks_op_st;
/* Ggks_sl_name GKS state list name */
typedef enum {
 GGKS_SL_ROUTE_DIR,      /* route direction*/
 GGKS_SL_SCISSOR_MODE,   /* scissor mode*/
 GGKS_SL_OPEN_WSS,       /* list of open workstations.*/
```

```
GGKS_SL_ACTIVE_WSS,     /* list of active workstations*/
GGKS_SL_OPEN_AUDITS,    /* set of open audits*/
GGKS_SL_OPEN_PLAYBACKS,       /* set of open playbacks*/
GGKS_SL_IN_QUEUE,       /* input queue*/
GGKS_SL_FONT_IND_MAP,   /* font index mapping*/
GGKS_SL_PICK_ID, /* pick identifier*/
GGKS_SL_NAMESET, /* nameset  */
GGKS_SL_GLOBAL_TRAN_MATRIX,   /* global
             transformation matrix */
GGKS_SL_LOCAL_TRAN_MATRIX,    /* local
             transformation matrix */
GGKS_SL_PAT_SIZE,       /* pattern size*/
GGKS_SL_PAT_REF_POINT, /* Pattern reference point*/
GGKS_SL_CHAR_HT, /* Character height*/
GGKS_SL_CHAR_UP_VEC,    /* character up vector*/
GGKS_SL_TEXT_SKEW_ANGLE,      /* text skew angle*/
GGKS_SL_TEXT_PATH,      /* text path*/
GGKS_SL_TEXT_ALIGN,     /* text alignment*/
GGKS_SL_SCISSOR_SET,    /* scissor set*/
GGKS_SL_LINE_IND,       /* line index*/
GGKS_SL_LINETYPE,       /* linetype*/
GGKS_SL_LINEWIDTH,      /* linewidth scale factor*/
GGKS_SL_LINE_COLR_SPECIF,     /* line colour specifier*/
GGKS_SL_MARKER_IND,     /* marker index*/
GGKS_SL_MARKER_TYPE,    /* marker type*/
GGKS_SL_MARKER_SIZE,    /* marker size scale factor*/
GGKS_SL_MARKER_COLR_SPECIF,   /* marker colour specifier*/
GGKS_SL_AREA_IND,       /* area ind*/
GGKS_SL_INT_STYLE,      /* interior style*/
GGKS_SL_INT_STYLE_IND, /* interior style index*/
GGKS_SL_INT_COLR_SPECIF,      /* interior colour specifier*/
GGKS_SL_EDGE_FLAG,      /* edge flag */
GGKS_SL_EDGETYPE,       /* edgetype */
GGKS_SL_EDGEWIDTH,      /* edgewidth scale factor */
GGKS_SL_EDGE_COLR_SPECIF,     /* edge colour specifier */
GGKS_SL_TEXT_IND,       /* text index*/
GGKS_SL_TEXT_FONT_PREC,       /* text font
             and precision    */
GGKS_SL_CHAR_EXPAN,     /* character expansion factor*/
GGKS_SL_CHAR_SPACE,     /* character spacing*/
GGKS_SL_TEXT_COLR_SPECIF,     /* text colour specifier*/
GGKS_SL_ASFS,     /* asfs      */
GGKS_SL_NORM_TRAN_NUM, /* current norm. tran. number */
GGKS_SL_NORM_TRANS,     /* list of norm. tran. */
GGKS_SL_VP_IN_PRI,      /* list of vp input pris*/
GGKS_SL_NAME_OPEN_PIC_PART,  /* name of open pic. part*/
GGKS_SL_PIC_PART_NAMES,       /* list of picture part names in use*/
GGKS_SL_CONT_ATTRS,     /* contour attributes*/
GGKS_SL_NAME_OPEN_SEG, /* nme of open segment*/
GGKS_SL_SEG_NAMES,      /* list of segments in use*/
GGKS_SL_SEG_SLS, /* list of segment
             state lists     */
```

```
 GGKS_SL_NAME_OPEN_STENCIL,   /* name of open stencil*/
 GGKS_SL_STENCIL_NAMES, /* list of stencils in use*/
 GGKS_SL_STENCIL_SLS,   /* list of stencil
             state lists in use*/
 GGKS_SL_NAME_OPEN_TILING,    /* name of open tiling*/
 GGKS_SL_TILING_NAMES   /* list of tilings in use*/
} Ggks_sl_name;
/* Gop_st_name operating state entry name */
typedef enum {
 GOP_ST_GKS,      /* GKS operating state*/
 GOP_ST_PIC_PART, /* picture part operating state*/
 GOP_ST_STENCIL,  /* stencil operating state*/
 GOP_ST_SEG,      /* segment operating state*/
 GOP_ST_CLOSED_PATH,    /* closed path operating state */
 GOP_ST_TILING    /* tiling operating state*/
} Gop_st_name;
/* Ghighl highlighting */
typedef enum {
 GSEG_NORM,
 GSEG_HIGHL
} Ghighl;
/* Gin_status [input] status */
typedef enum {
 GIN_STATUS_NONE,
 GIN_STATUS_OK ,
 GIN_STATUS_NO_IN
} Gin_status;
/* Ginit_sel initial selector */
typedef enum {
 GSEL_INIT_VALUE, /* initial value*/
 GSEL_PET,  /* prompt and echo type */
 GSEL_ECHO_SWITCH,      /* echo switch */
 GSEL_ECHO_AREA,  /* echo area*/
 GSEL_IN_DATA,    /* input data*/
 GSEL_ALL   /* all initial input*/
} Ginit_sel;
/* Ginq_type inquire type */
typedef enum {
 GINQ_SET,
 GINQ_REALIZED
} Ginq_type;
/* Ginside_rule inside rule */
typedef enum {
 GRULE_EVEN_ODD,
 GRULE_WINDING
} Ginside_rule;
/* Gint_style interior style */
typedef enum {
 GSTYLE_HOLLOW,
 GSTYLE_SOLID,
 GSTYLE_PAT,
 GSTYLE_HATCH,
 GSTYLE_EMPTY
} Gint_style;
/* Gproc_op process operation */
typedef enum {
 GPROC_SKIP,
 GPROC_DO
} Gproc_op;
/* Gline_fill_ctrl_flag line/fill control flag */
typedef enum {
 GFLAG_LINE,
 GFLAG_FILL
} Gline_fill_ctrl_flag;
/* Gin_class input [measure] class */
typedef enum {
 GIN_NONE,
 GIN_LOC,
 GIN_STROKE,
 GIN_VAL,
 GIN_CHOICE,
 GIN_PICK,
 GIN_STRING,
 GIN_COMP
} Gin_class;
/* Gop_mode operating mode */
typedef enum {
 GOP_REQ,
 GOP_SAMPLE,
 GOP_EVENT
} Gop_mode;
/* Gord_sel ordinate selector */
typedef enum {
 GORD_X,
 GORD_Y
} Gord_sel;
/* Gtiling_op_st tiling operating state */
typedef enum {
 GOP_ST_TLCL,
 GOP_ST_TLOP
} Gtiling_op_st;
/* Gpic_part_op_st picture part operating state */
typedef enum {
 GOP_ST_PPCL,
 GOP_ST_PPOP
} Gpic_part_op_st;
/* Gpick_status pick status */
typedef enum {
 GPICK_OK,
 GPICK_NO_PICK
} Gpick_status;
/* Gplayback_op playback operation */
typedef struct {
 enum Gplayback_op_type {
```

```
             GPLAYBACK_OPEN, GPLAYBACK_CLOSE
 } type;
 void      *specif;    /* audit specifier*/
} Gplayback_op;
/* Gpr_flag prompt flag */
typedef enum {
 GPR_OFF,
 GPR_ON
} Gpr_flag;
/* Gprim_type primtive type */
typedef enum {
 GPRIM_LINE_SET,
 GPRIM_NURB_SET,
 GPRIM_CONIC_SEC_SET,
 GPRIM_MARKER,
 GPRIM_FILL_SET,
 GPRIM_CLOSED_NURB_SET,
 GPRIM_ELL_SEC_SET,
 GPRIM_ELL_SEG_SET,
 GPRIM_ELL_DISC_SET,
 GPRIM_TEXT,
 GPRIM_CELL_ARRAY,
 GPRIM_DESIGN,
 GPRIM_GDP
} Gprim_type;
/* Grel_pos relative position */
typedef enum {
 GPOS_FRONT,
 GPOS_BACK
} Grel_pos;
/* Grel_pri relative priority */
typedef enum {
 GPRI_HIGHER,
 GPRI_LOWER
} Grel_pri;
/* Gscissor_mode scissor mode */
typedef enum {
 GSCISSOR_LOCUS,
 GSCISSOR_SHAPE
} Gscissor_mode;
/* Gscissor_sel scissor select */
typedef enum {
 GSCISSOR_CHANGE,
 GSCISSOR_LEAVE
} Gscissor_sel;
/* Gseg_attr_name segment attribute name */
typedef enum {
 GNAME_TRAN_MATRIX,     /* transformation matrix*/
 GNAME_VIS, /* visibility     */
 GNAME_HIGHL,     /* hightlighting*/
 GNAME_PRI, /* segment priority*/
 GNAME_DET  /* detectability  */
```

```
} Gseg_attr_name;
/* Gsel_crit_op selection criterion operation */
typedef enum {
 GOP_CONTAINS,
 GOP_IS_IN,
 GOP_EQUALS,
 GOP_SEL_ALL,
 GOP_REJ_ALL,
 GOP_AND,
 GOP_OR,
 GOP_NOT
} Gsel_crit_op;
/* Gsel_crit_type selection criterion type */
typedef enum {
 GSEL_DISP,
 GSEL_HIGHL,
 GSEL_DET
} Gsel_crit_type;
/* Gsense_flag sense flag */
typedef enum {
 GFLAG_CLOCKWISE,
 GFLAG_ANTI_CLOCKWISE
} Gsense_flag;
/* Gshield_ind shielding indicator */
typedef enum {
 GIND_NO_SHIELD,
 GIND_SHIELD
} Gshield_ind;
/* Gstencil_attr_name stencil attribute name */
typedef enum {
 GNAME_TOP_Y,
 GNAME_CAP_Y,
 GNAME_HALF_Y,
 GNAME_BASE_Y,
 GNAME_BOTTOM_Y,
 GNAME_CTR_Y,
 GNAME_LEFT_X,
 GNAME_RIGHT_X,
 GNAME_CTR_X,
 GNAME_CTR,
 GNAME_ORIGIN,
 GNAME_CTR_TOP,
 GNAME_CTR_BOTTOM,
 GNAME_CTR_LEFT,
 GNAME_CTR_RIGHT,
 GNAME_TOP_LEFT,
 GNAME_TOP_RIGHT,
 GNAME_BOTTOM_LEFT,
 GNAME_BOTTOM_RIGHT
} Gstencil_attr_name;
/* Gstencil_op_st stencil operating state */
typedef enum {
 GOP_ST_STCL,
 GOP_ST_STOP
} Gstencil_op_st;
/* Gseg_op_st segment operating state */
typedef enum {
 GOP_ST_SGCL,
 GOP_ST_SGOP
} Gseg_op_st;
/* Gtext_path text path */
typedef enum {
 GPATH_RIGHT,
 GPATH_LEFT,
 GPATH_UP,
 GPATH_DOWN
} Gtext_path;
/* Gtext_prec text precision */
typedef enum {
 GPREC_STRING,
 GPREC_CHAR,
 GPREC_STROKE
} Gtext_prec;
/* Gtiling_type tiling type */
typedef enum {
 GTILING_LINE_SET,
 GTILING_NURB_SET,
 GTILING_CONIC_SEC_SET,
 GTILING_FILL_SET,
 GTILING_CLOSED_NURB_SET,
 GTILING_ELL_SEC_SET,
 GTILING_ELL_SEG_SET,
 GTILING_ELL_DISC_SET,
 GTILING_DESIGN
} Gtiling_type;
/* Gtran_mode transformation mode */
typedef enum {
 GTRAN_REPLACE,
 GTRAN_PRE,
 GTRAN_POST
} Gtran_mode;
/* Gvis visibility */
typedef enum {
 GSEG_INVIS,
 GSEG_VIS
} Gvis;
/* Gvis_effects_st visual effects state */
typedef enum {
 GST_SUPPR,
 GST_ALLOW
} Gvis_effects_st;
/* Gws_class workstation classification [display kind] */
typedef enum {
 GCLASS_VEC,
 GCLASS_RASTER,
 GCLASS_OTHER
} Gws_class;
/* Gws_dt_name workstation description table name */
typedef enum {
 GWS_DT_TYPE,
 GWS_DT_STATUS,
 GWS_DT_DC_UNITS,
 GWS_DT_DISP_SPACE_SIZE,
 GWS_DT_CLASS,
 GWS_DT_LINE_FACS,
 GWS_DT_LINE_BUNDLE_TABLE,
 GWS_DT_MARKER_FACS,
 GWS_DT_MARKER_BUNDLE_TABLE,
 GWS_DT_AREA_FACS,
 GWS_DT_AREA_BUNDLE_TABLE,
 GWS_DT_PAT_TABLE,
 GWS_DT_TEXT_FACS,
 GWS_DT_TEXT_BUNDLE_TABLE,
 GWS_DT_COLR_FACS,
 GWS_DT_COLR_TABLE,
 GWS_DT_GDP_IDS,
 GWS_DT_DEV_IDS,
 GWS_DT_IN_DEV_INIT_VALUES,
 GWS_DT_MEAS_IDS,
 GWS_DT_TRIGGER_IDS,
 GWS_DT_DEV_COMPS
} Gws_dt_name;
/* Gws_sl_name workstation state list name */
typedef enum{
 GWS_SL_ST,
 GWS_SL_CONN_TYPE,
 GWS_SL_LINE_BUNDLE_TABLE,
 GWS_SL_MARKER_BUNDLE_TABLE,
 GWS_SL_TEXT_BUNDLE_TABLE,
 GWS_SL_AREA_BUNDLE_TABLE,
 GWS_SL_PAT_TABLE,
 GWS_SL_COLR_TABLE,
 GWS_SL_WIN_VP,
 GWS_SL_VIS_EFFECTS_ST,
 GWS_SL_OP_MODES_INIT_VALS,
 GWS_SL_SEL_TABLE,
 GWS_SL_STORED_SEGS,
 GWS_SL_VIEW_TABLE,
 GWS_SL_VIEW_PRIS
} Gws_sl_name;
/* Gws_status workstation status */
typedef enum {
 GWS_UP_TO_DATE,
 GWS_NOT_UP_TO_DATE
} Gws_status;
/* Gws_st workstation state */
```

```
typedef enum {
 GWS_INACTIVE,
 GWS_ACTIVE
} Gws_st;
/* Simple Structures*/
/* Gasfs attribute source flags  */
typedef struct {
 Gasf        linetype;   /* linetype ASF */
 Gasf        linewidth;  /* linewidth scale factor ASF */
 Gasf        line_colr_ind;    /* line colour ASF */
 Gasf        marker_type;      /* marker type ASF */
 Gasf        marker_size;      /* marker size scale factor ASF */
 Gasf        marker_colr_ind;  /* marker colour ASF */
 Gasf        text_font_prec;   /* text font and precision ASF */
 Gasf        char_expan; /* character expansion factor ASF */
 Gasf        char_space; /* character spacing ASF */
 Gasf        text_colr_ind;    /* text colour ASF */
 Gasf        int_style;  /* interior style ASF */
 Gasf        int_style_ind;    /* interior style index ASF */
 Gasf        int_colr_ind;     /* interior colour ASF */
 Gasf        edge_flag;  /* edge flag ASF */
 Gasf        edgetype;   /* edgetype ASF */
 Gasf        edgewidth;  /* edgewidth scale factor ASF */
 Gasf        edge_colr_ind;    /* edge colour ASF */
} Gasfs;
/* Gchoice choice [value] */
typedef struct {
 Gchoice_status   status;     /* status*/
 Gint        num;   /* choice number*/
} Gchoice;
/* Gcieluv CIE L*u*v* [colour specification] */
typedef struct {
 Gfloat      u_prime;  /* u' coefficient*/
 Gfloat      v_prime;  /* v' coefficient*/
 Gfloat      y;    /* y luminance*/
} Gcieluv;
/* Gdev_id device identifier */
typedef struct {
 Gint        ws_id;      /* workstation identifier*/
 Gint        num;   /* device number*/
 Gin_class   in_class;   /* input [measure] class*/
} Gdev_id;
/* Gdev_id_list device identifier list */
typedef struct {
 Gint        num_dev_ids;      /* number of device identifiers in list*/
 Gdev_id     *dev_ids;   /* list of device identifiers*/
} Gdev_id_list;
/* Gfloat_list float list */
typedef struct {
 Gint        num_floats; /* number of floats in list*/
 Gint        *floats;    /* list of floats*/
} Gfloat_list;
```

```
/* Gfloat_size float size */
typedef struct {
 Gfloat     size_x;     /* x size*/
 Gfloat     size_y;     /* y size*/
} Gfloat_size;
/* Ghls Hue Luminance Saturation [colour specification] */
typedef struct {
 Gfloat     hue;  /* hue      */
 Gfloat     lightness;  /* lightness*/
 Gfloat     satur;      /* saturation*/
} Ghls;
/* Ghomo_point homogeneous point */
typedef struct {
 Gfloat     xw;   /* x coordinate*/
 Gfloat     yw;   /* y coordinate*/
 Gfloat     w;    /* w coordinate*/
} Ghomo_point;
/* Ghsv Hue Saturation Value [colour specification] */
typedef struct {
 Gfloat     hue;  /* hue      */
 Gfloat     satur;      /* saturation*/
 Gfloat     value;      /* value*/
} Ghsv;
/* Gint_list integer list */
typedef struct {
 Gint       num_ints;   /* number of integers in list*/
 Gint       *ints;      /* list of integers*/
} Gint_list;
/* Gint_size integer size */
typedef struct {
 Gint       size_x;     /* x size*/
 Gint       size_y;     /* y size*/
} Gint_size;
/* Gjoin join  */
typedef struct {
 enum Gjoin_name {/* name (ROUND|BEVEL|MITRED) */
            GJOIN_ROUND,
            GJOIN_BEVEL,
            GJOIN_MITRED
  } name;
 Gfloat     mitred;     /* mitred value */
} Gjoin;
/* Glimit limit */
typedef struct {
 Gfloat     x_min;      /* x min*/
 Gfloat     x_max;      /* x max*/
 Gfloat     y_min;      /* y min*/
 Gfloat     y_max;      /* y max*/
} Glimit;
/* Gnum_in number [of] input [devices] */
typedef struct {
 Gint       loc;  /* number of locator devices*/
 Gint       stroke;     /* number of stroke devices*/
 Gint       val;  /* number of valuator devices*/
 Gint       choice;     /* number of choice devices*/
 Gint       pick; /* number of pick devices*/
 Gint       string;     /* number of string devices*/
 Gint       comp; /* number of composed devices*/
} Gnum_in;
/* Gpoint point */
typedef struct {
 Gfloat     x;    /* x coordinate*/
 Gfloat     y;    /* y coordinate*/
} Gpoint;
/* Grgb Red Green Blue [colour specification] */
typedef struct {
 Gfloat     red;  /* red intensity*/
 Gfloat     green;      /* green intensity*/
 Gfloat     blue; /* blue intensity*/
} Grgb;
/* Gseg_attr_value segment attribute value */
typedef struct {
 Gseg_attr_name name; /* segment attribute name */
 union {
           Gtran_matrix      tran_matrix;/* transformation matrix*/
           Gvis   vis;  /* visibility*/
           Ghighl      highl;/* hightlighting*/
           Gfloat      pri;  /* segment priority*/
           Gdet   det;  /* detectability*/
 } value;
} Gseg_attr_value;
/* Gseg_attrs segment attributes */
typedef struct {
 Gtran_matrix     tran_matrix;/* transformation matrix*/
 Gvis       vis;  /* visibility*/
 Ghighl     highl;      /* hightlighting*/
 Gfloat     pri;  /* segment priority*/
 Gdet       det;  /* detectability*/
} Gseg_attrs;
/* Gseg_sl segment state list */
typedef struct {
 Gint       name; /* segment name*/
 Gint_list  assoc_wss;  /* associated workstations*/
 Gseg_attrs attrs;      /* segment attributes*/
} Gseg_sl;
/* Gseg_sl_list list of segment state lists */
typedef struct {
 Gint       num_seg_sls;      /* number of segment state lists in list*/
 Gseg_sl    *seg_sls;   /* list of segment state lists*/
} Gseg_sl_list;
/* Gtext_align text alignment */
typedef struct {
 enum Ghor_text_align { /* horizontal text alignment */
           GHOR_NORM,
             GHOR_LEFT,
             GHOR_CTR,
             GHOR_RIGHT
 } hor;
 enum Gvert_text_align { /*  vertical text alignment */
             GVERT_NORM,
             GVERT_TOP,
             GVERT_CAP,
             GVERT_HALF,
             GVERT_BASE,
             GVERT_BOTTOM
 } vert;
} Gtext_align;
/* Gtext_font_prec text font [and] precision */
typedef struct {
 Gint        font; /* character font*/
 Gtext_prec prec; /* text precision*/
} Gtext_font_prec;
/* Gvec vector */
typedef struct {
 Gfloat      delta_x;    /* x coordinate*/
 Gfloat      delta_y;    /* y coordinate*/
} Gvec;
/* Nested Structures */
/* Gcolr_specif colour specifier */
typedef struct {
 Gcolr_kind kind; /* colour kind */
 union {
             Gint   ind;  /* colour index*/
             Grgb   rgb;  /* Red Green Blue colour specification */
             Gcieluv     cieluv;/* CIE L*u*v* 1976 colour specification */
             Ghls   hls;  /* Hue Luminance Saturation colour specification */
             Ghsv   hsv;  /* Hue Saturation Value colour specification */
 } value;
} Gcolr_specif;
/* Gconic_sec conic section  */
typedef struct {
 Gconic_matrix    matrix;     /* conic matrix */
 Gpoint      point_1;    /* point # 1 */
 Gpoint      point_2;    /* point # 2 */
 Gsense_flag      sense_flag; /* sense flag */
} Gconic_sec;
/* Gconic_sec_list conic primitive list */
typedef struct {
 Gint        num_conic_prims;  /* number of conic primitives in list*/
 Gconic_sec *conic_prims;      /* list of conic primitives*/
} Gconic_sec_list;
/* Gcont_attr_value contour attribute value */
typedef struct {
 Gcont_attr_name  name; /* attribute name */
 union {
             Gint  style;      /* style*/
```

```
            Gfloat      width;/* width */
            Gcap  cap;  /* cap*/
            Gjoin join; /* join */
 } value;
} Gcont_attr_value;
/* Gcont_attrs contour attributes */
typedef struct {
 Gint       style;      /* style*/
 Gfloat     width;      /* width */
 Gcap       cap;  /* cap       */
 Gjoin      join; /* join      */
} Gcont_attrs;
/* Gdesign design  */
typedef struct {
 Gint       stencil_name;     /* stencil name */
 Gpoint     stencil_origin;   /* stencil origin*/
 Gtran_matrix     stencil_tran;/* stencil transformation */
 Gint       tiling_name;      /* tiling name*/
 Gpoint     tiling_origin;    /* tiling origin */
 Gtran_matrix     tiling_tran;/* tiling transformation */
} Gdesign;
/* Gdev_comp device composition */
typedef struct {
 Gdev_id    id;   /* device id*/
 Gint_list  meas_set;   /* measure set*/
 Gint_list  trigger_set;      /* triger set*/
} Gdev_comp;
/* Gdev_comp_list device composition list */
typedef struct {
 Gint       num_devs;   /* number of [basic] devices in list*/
 Gdev_comp  *dev_comps; /* list of device compositions*/
} Gdev_comp_list;
/* Gell_disc_list elliptic disc list */
typedef struct {
 Gint       num_ell_discs;    /* number of elliptic discs in list*/
 Gconic_matrix     *ell_discs; /* list of elliptic discs*/
} Gell_disc_list;
/* Gfont_ind_map font index mapping */
typedef struct {
 Gint       ind;  /* font index */
 Gint       iso9541_name;     /* ISO 9541 font name*/
} Gfont_ind_map;
/* Gop_st_entry operating state entry */
typedef struct {
 Gop_st_name      name; /* GKS operating state name */
 union {
            Ggks_op_st  gks;  /* GKS operating state*/
            Gpic_part_op_st   pic_part;/* picture part operating state*/
            Gstencil_op_st    stencil;/* stencil operating state*/
            Gseg_op_st  seg;  /* segment operating state*/
            Gclosed_path_op_stclosed_path;/* closed path operating
                               state*/
            Gtiling_op_st     tiling;/* tiling operating state*/
 } value;
} Gop_st_entry;
/* Ginst_specif instance specifier */
typedef struct {
 enum Ginst_type {/*  instance type */
            GINST_PUT,
            GINST_ALIGN,
            GINST_MAP
 } type;
 union {
            struct Gput_inst {
                  Gpoint      ref_point_inst;/* ref. point on instance */
                  Gpoint      ref_point_open_stencil;/* ref. point on
                                 open stencil */
            } put;
            struct Galign_inst{
                  Gpoint      ref_point_inst;/* ref. point on instance */
                  Gpoint      dir_point_inst;/* direction point on instance */
                  Gpoint      ref_point_open_stencil;/* ref. point on
                                 open stencil */
                  Gpoint      dir_point_open_stencil;/* direction point on
                                 open stencil */
            } align;
            struct Gmap_inst {
                  Gpoint      ref_point_inst[3];/* 3 ref. points on instance */
                  Gpoint      corr_point_open_stencil[3];/* 3 corr. points on
                                 open stencil */
            } map;
 } specif;
} Ginst_specif;
/* Gopen_audit_list open audit list */
typedef struct {
 Gint       num_audits; /* number of audits */
 Gint       *audit_ids; /* list of audit identifiers */
 Gaudit_flag      *audit_flags;/* list of audit flags */
} Gopen_audit_list;
/* Gseq_specif sequence specifier */
typedef struct {
 Gint       inst_name;  /* instance name*/
 Gint       stencil_name;     /* existing stencil name */
 Gpoint     ref_point;  /* reference point */
} Gseq_specif;
/* Gseq_specif_list sequence specifier list */
typedef struct {
 Gint       num_seq_specifs;  /* number of sequence specifiers*/
 Gseq_specif      *seq_specif;/* list of sequence specifiers*/
} Gseq_specif_list;
/* Gstencil_attr_value stencil attribute value */
typedef struct {
 Gstencil_attr_name     name; /* attribute name*/
 union {
```

```
            Gfloat      top_y;/* top y*/
            Gfloat      half_y;/* half y*/
            Gfloat      base_y;/* base y*/
            Gfloat      bottom_y;/* bottom y*/
            Gfloat      ctr_y;/* centre y*/
            Gfloat      left_x;/* left x*/
            Gfloat      ctr_x;/* centre x*/
            Gfloat      right_x;/* right x*/
            Gpoint      ctr;  /* centre*/
            Gpoint      origin;/* origin*/
            Gpoint      ctr_top;/* centre top*/
            Gpoint      ctr_bottom;/* centre bottom*/
            Gpoint      ctr_left;/* centre left*/
            Gpoint      ctr_right;/* centre right*/
            Gpoint      top_left;/* top left*/
            Gpoint      top_right;/* topr ight*/
            Gpoint      bottom_left;/* bottom left*/
            Gpoint      bottom_right;/* bottom right*/
 } value;
} Gstencil_attr_value;
/* Gstencil_attrs stencil attributes */
typedef struct {
 Gfloat     top_y;      /* top y*/
 Gfloat     cap_y;      /* capy */
 Gfloat     half_y;     /* half y*/
 Gfloat     base_y;     /* base y*/
 Gfloat     bottom_y;   /* bottom y*/
 Gfloat     ctr_y;      /* centre y*/
 Gfloat     left_x;     /* left x*/
 Gfloat     ctr_x;      /* centre x*/
 Gfloat     right_x;    /* right x*/
 Gpoint     ctr;  /* centre   */
 Gpoint     origin;     /* origin*/
 Gpoint     ctr_top;    /* centre top*/
 Gpoint     ctr_bottom; /* centre bottom*/
 Gpoint     ctr_right;  /* centre right*/
 Gpoint     top_left;   /* top left*/
 Gpoint     top_right;  /* topr ight*/
 Gpoint     bottom_left;      /* bottom left*/
 Gpoint     bottom_right;     /* bottom right*/
} Gstencil_attrs;
/* Gstencil_sl stencil state list */
typedef struct {
 Gint       name; /* stencil name*/
 Gstencil_attrs  attrs;     /* stencil attributes*/
} Gstencil_sl;
/* Gstencil_sl_list list of stencil state lists */
typedef struct {
 Gint        num_stencil_sls;  /* number of stencil_sls in the list*/
 Gstencil_sl      *stencil_sls;/* list of stencil_sls*/
} Gstencil_sl_list;
/* Gctrl_point_list control point list */
```

```
typedef struct {
 enum Grat { /* rationality (RATIONAL|POLYNOMIAL) */
           GNURB_RAT,
           GNURB_POL
 } rat;
 Gint      num_points; /* number of points in the list */
 union {
           Ghomo_point *homo_points;/* list of homg. points*/
           Gpoint      *points;/* list of points*/
  } ctrl_points;
} Gctrl_point_list;
/* Gdisp_space_size display space size */
typedef struct {
 Gdc_units  dc_units;   /* device coordinate units*/
 Gfloat_size      size_dc;    /* display space size [in] dc [units]*/
 Gint_size  size_raster;      /* display space size [in]
                  raster [units]*/
} Gdisp_space_size;
/* Garea_facs area facilities */
typedef struct {
 Gint       num_int_styles;   /* number of interior styles*/
 Gint_style int_styles[5];    /* list of available interior styles*/
 Gint_list  hatch_styles;     /* list of available hatch styles*/
 Gint       num_pred_inds;    /* number of predef. area indices*/
 Gint_list  edgetypes;   /* list of edgetypes*/
 Gint       num_edgewidths;   /* number of available edgewidths*/
 Gfloat     nom_edgewidth;    /* nominal edgewidth*/
 Gfloat     min_edgewidth;    /* min. edgewidth*/
 Gfloat     max_edgewidth;    /* max. edgewidth*/
} Garea_facs;
/* Glimit_list limit list */
typedef struct {
 Gint       num_limits; /* number of limits in list*/
 Glimit     *limits;    /* list of limits*/
} Glimit_list;
/* Gline_facs line facilities */
typedef struct {
 Gint_list  types;      /* list of linetypes*/
 Gint       num_widths; /* number of available linewidths*/
 Gfloat     nom_width;  /* nominal linewidth*/
 Gfloat     min_width;  /* min. linewidth*/
 Gfloat     max_width;  /* max. linewidth*/
 Gint       num_pred_inds;    /* number of predef. line indices*/
} Gline_facs;
/* Gloc locator value */
typedef struct {
 Gpoint     point;      /* loc point */
 Gint       norm_tran_num;    /* norm. tran. number */
} Gloc;
/* Gmarker_facs marker facilities */
typedef struct {
 Gint_list  types;      /* list of marker types*/
 Gint       num_sizes;  /* number of available marker sizes*/
 Gfloat     nom_size;   /* nominal marker size*/
 Gfloat     min_size;   /* min. marker size*/
 Gfloat     max_size;   /* max. marker size*/
 Gint       num_pred_inds;    /* number of predef. marker indices*/
} Gmarker_facs;
/* Gnameset nameset */
typedef struct {
 Gint_list  ws_ids;     /* list of ws idenfiers*/
 Gint_list  seg_names;  /* list of segment names*/
 Gint_list  names;      /* list of names*/
} Gnameset;
/* Gnurb NURB  */
typedef struct {
 Gint       order;      /* order */
 Gctrl_point_list ctrl_point_list;/* list of control points */
 Gfloat_list      weights;    /* list of weight factors */
 Gfloat_list      knots;      /* list of knots */
 Gfloat     t_min;      /* t_min */
 Gfloat     t_max;      /* t_max */
} Gnurb;
/* Gnurb_list NURB list */
typedef struct {
 Gint       num_nurbs;  /* number of NURBs in list*/
 Gnurb      *nurbs;     /* list of NURBs*/
} Gnurb_list;
/* Gpat_rep pattern representation */
typedef struct {
 Gint_size  dims; /* colour array's dimensions*/
 Gcolr_kind colr_kind;  /* colour kind*/
 Gint       *colr_array;      /* colour array*/
 union {
            Grgb   *rgb;  /* RGB colours */
            Gcieluv      *cieluv;/* CIELUV colours */
            Ghls   *hls;  /* HLS colours */
            Ghsv   *hsv;  /* HSV colours */
  } dir_colr;
} Gpat_rep;
/* Gpat_table pattern table */
typedef struct {
 Gint       num_inds;   /* number of pattern indices*/
 Gint       *inds;      /* list of pattern indices*/
 Gpat_rep   *pat_reps;  /* list of pattern representations*/
} Gpat_table;
/* Gpick pick [value] */
typedef struct {
 Gpick_status     status;     /* status*/
 Gnameset   set_names;  /* set of names*/
 Gint       seg_name;   /* segment name*/
 Gint       pick_id;    /* pick identifier*/
} Gpick;
/* Gpoint_list point list */
```

```
typedef struct {
 Gint       num_points; /* number of points in the list*/
 Gpoint     *points;    /* list of points*/
} Gpoint_list;
/* Gpoint_list_list list of point lists */
typedef struct {
 Gint       num_point_lists;  /* number of point lists in the list*/
 Gpoint_list      *point_lists;/* list of point lists*/
} Gpoint_list_list;
/* Gpath path */
typedef struct {
  enum Gpath_type { /* path type */
           GPATH_LINE,
           GPATH_NURB,
           GPATH_CONIC_SEC
 } type;
 union {
           Gpoint_list polyline;/* polyline */
           Gnurb nurb; /* NURB*/
           Gconic_sec  conic_sec;/* conic section */
 } value;
} Gpath;
/* Gpath_list path list */
typedef struct {
 Gint       num_paths;  /* number of paths in list*/
 Gpath      *paths;     /* list of paths*/
} Gpath_list;
/* Gpath_specif path specifier */
typedef struct {
 Gabut_specif     abut_specif;/* Abut specifier*/
 Gpath      path; /* path      */
 Gpoint     ref_point;  /* reference point */
} Gpath_specif;
/* Gbndry boundary */
typedef struct {
 enum Gbndry_type {
           GBNDRY_CLOSED_PATH,
           GBNDRY_ELL_DISC
 } type;
 union {
           Gpath_list  closed_path_seq;/* closed path sequence*/
           Gconic_matrix     ell_disc;/* elliptic disc */
 } value;
} Gbndry;
/* Gbndry_list boundary list */
typedef struct {
 Gint       num_bndries;      /* number of boundaries in list*/
 Gbndry     *bndries;   /* list of boundaries*/
} Gbndry_list;
/* Grect rectangle */
typedef struct {
 Gpoint     p;    /* point p  */
 Gpoint     q;    /* point q  */
} Grect;
/* Gscissor scissor */
typedef struct {
 Gclip_ind  clip_ind;   /* clipping indicator*/
 Glimit_list     clip_rect_set;/* clipping rectangle set*/
 Gshield_ind     shield_ind; /* shielding indicator*/
 Glimit_list     shield_rect_set;/* shielding rectangle set*/
} Gscissor;
/* Gscissor_list scissor list */
typedef struct {
 Gint       num_scissors;     /* number of scissors in list*/
 Gint       *scissor_ids;     /* list of scissor identifications*/
 Gscissor   *scissors;  /* list of scissors*/
} Gscissor_list;
/* Gsel_crit selection criterion */
typedef struct GSEL_CRIT *Gsel_crit_ptr;
typedef struct GSEL_CRIT {
 Gsel_crit_op     op;   /* operation */
 union {
           Gnameset    compar;/* comparison
                          (EQUALS|CONTAINS|IS-IN) */;
           struct Gdyad_sel_crit {
                 Gsel_crit_ptrleft;/* left selection operand */
                 Gsel_crit_ptrright;/* right selection operand */
           } dyad;     /* dyadic criterion (AND|OR) */
           Gsel_crit_ptr     monad;/* monadic criterion (NOT) */
 } value;
} Gsel_crit;
/* Ggks_dt_entry GKS description table entry */
typedef struct {
 Ggks_dt_name     name;  /* entry name */
 union {
           Gint_list   avail_ws_gen_types;/* list of available ws. types*/
           Gint_list   avail_fonts;/* list of available fonts*/
           Gfont_ind_map     def_font_ind_map;/* default font index mapping*/
 } value;
} Ggks_dt_entry;
/* Gsel selection  */
typedef struct {
 Gsel_crit_type   type; /* selection type */
 Gsel_crit  crit; /* selection criterion */
} Gsel;
/* Gsel_table selection table */
typedef struct {
 Gint       num_sels;   /* number of selections */
 Gsel       *sels;      /* table of selections */
} Gsel_table;
/* Gstroke stroke value */
typedef struct {
 Gpoint_list      point_list; /* stroke point list */
 Gint       norm_tran_num;    /* norm. tran. number */
```

```
} Gstroke;
/* Gtext_extent text extent */
typedef struct {
 Gpoint     concat_point;     /* concatenation point*/
 Gpoint     paral[4];   /* text extent parallelogram*/
} Gtext_extent;
/* Gtext_facs text facilities */
typedef struct {
 Gint       num_font_precs;   /* number of fonts and precisions*/
 Gtext_font_prec  *font_precs;/* list of fonts and precisions*/
 Gint       num_char_hts;     /* number of character heights*/
 Gfloat     min_char_ht;      /* min. character height*/
 Gfloat     max_char_ht;      /* max. character height*/
 Gint       num_char_expans;  /* number of char. expansion
                     factors  */
 Gfloat     min_char_expan;   /* min. expansion factor*/
 Gfloat     max_char_expan;   /* max. expansion factor*/
 Gint       num_pred_inds;    /* number of predef. text indices*/
} Gtext_facs;
/* Gtran transformation */
typedef struct {
 Glimit     win;   /* window   */
 Glimit     vp;    /* viewport */
} Gtran;
/* Gview_rep view representation */
typedef struct {
 Gsel_crit  sel_crit;   /* view selection criterion*/
 Gtran_matrix     ori_matrix; /* view orientation matrix*/
 Gtran_matrix     map_matrix; /* view mapping matrix*/
 Gscissor   scissor;    /* view scissor*/
} Gview_rep;
/* Gprim_attr_value primitive attribute value */
typedef struct {
 Gprim_attr_name name; /* primitive attribute name */
 union {
            Gint  pick_id;    /* pick identifier*/
            Gnameset    nameset;/* nameset */
            Gscissor_list     scissor_set;/* scissor set*/
            Gtran_matrix      global_tran_matrix;/* global
                           transformation matrix */
            Gtran_matrix      local_tran_matrix;/* local
                           transformation matrix */
            Gfloat_size pat_size;/* pattern size*/
            Gpoint      pat_ref_point;/* pattern
                           reference point*/
            Gfloat      char_ht;/* character height*/
            Gvec  char_up_vec;/* character up vector*/
            Gfloat      text_skew_angle;/* text skew angle*/
            Gtext_path  text_path;/* text path*/
            Gtext_align text_align;/* text alignment*/
            Gint  line_ind;   /* line index*/
            Gint  linetype;   /* linetype*/
            Gfloat      linewidth;/* linewidth
                           scale factor*/
            Gcolr_specif      line_colr_specif;/* line colour
                           specifier*/
            Gint   marker_ind; /* marker index*/
            Gint   marker_type;/* marker type*/
            Gfloat      marker_size;/* marker size
                           scale factor*/
            Gcolr_specif      marker_colr_specif;/* marker colour
                           specifier*/
            Gint   area_ind;   /* area ind*/
            Gint_style  int_style;/* interior style*/
            Gint   int_style_ind;/* interior style index*/
            Gcolr_specif      int_colr_specif;/* interior colour
                           specifier*/
            Gedge_flag  edge_flag;/* edge flag */
            Gint   edgetype;   /* edgetype */
            Gfloat      edgewidth;/* edgewidth
                           scale factor */
            Gcolr_specif      edge_colr_specif;/* edge colour
                           specifier */
            Gint   text_ind;   /* text index*/
            Gtext_font_prec   text_font_prec;/* text font
                           and precision*/
            Gfloat      char_expan;/* character
                           expansion factor*/
            Gfloat      char_space;/* character spacing*/
            Gcolr_specif      text_colr_specif;/* text colour
                           specifier*/
            Gasfs asfs; /* asfs*/
 }value;
} Gprim_attr_value;
/* Gbasic_in_value basic input value */
typedef struct{
 Gin_class class; /* basic input [measure] class */
 union {
            Gloc   loc;  /* locator value */
            Gstroke     stroke;/* stroke value */
            Gfloat      val;  /* valuator value */
            Gchoice     choice;/* choice value */
            Gpick pick; /* pick value */
            char   *string;    /* string value */
 } in_value; /* input value */
} Gbasic_in_value;
/* Gcomp_value composite value */
typedef struct {
 Gint        num_devs;   /* number of [basic] devices */
 Gbasic_in_value  *basic_in_value;/* list of basic input values*/
} Gcomp_value;
/* Gin_value input value */
typedef union {
 Gloc        loc;  /* locator value*/
 Gstroke    stroke;     /* strokevalue*/
 Gfloat     val;  /* valuator value*/
 Gchoice    choice;     /* choice value*/
 Gpick      pick; /* pick value*/
 char       *string;    /* string value*/
 Gcomp_value      comp; /* composite value*/
} Gin_value;
/* Gin_queue_item input item [in queue] */
typedef struct {
 Gdev_id    dev_id;     /* device identifier*/
 Gin_value  in_value;   /* input value*/
} Gin_queue_item;
/* Gin_queue input queue */
typedef struct {
 Gint       num_items;  /* number of input items*/
 Gin_queue_item   *in_items;  /* input items*/
} Gin_queue;
/* Ggks_sl_entry GKS state list entry */
typedef struct {
 Ggks_sl_name     name; /* entry name */
 union {
            Groute_dir  route_dir;/* route direction*/
            Gscissor_mode     scissor_mode;/* scissor mode*/
            Gint_list   open_wss;/* list of open workstations.*/
            Gint_list   active_wss;/* list of active workstations*/
            Gopen_audit_list  open_audits;/* set of open audits*/
            Gint_list   open_playbacks;/* set of open playbacks*/
            Gin_queue   in_queue;/* input queue*/
            Gfont_ind_map     font_ind_map;/* font index mapping*/
            Gint  pick_id;    /* pick identifier*/
            Gnameset    nameset;/* nameset */
            Gtran_matrix      global_tran_matrix;/* global
                        transformation matrix */
            Gtran_matrix      local_tran_matrix;/* local
                        transformation matrix */
            Gfloat_size pat_size;/* pattern size*/
            Gpoint      pat_ref_point;/* pattern reference point*/
            Gfloat      char_ht;/* character height*/
            Gvec  char_up_vec;/* character up vector*/
            Gfloat      text_skew_angle;/* text skew angle*/
            Gtext_path  text_path;/* text path*/
            Gtext_align text_align;/* text alignment*/
            Gscissor_list     scissor_set;/* scissor set*/
            Gint  line_ind;   /* line index*/
            Gint  linetype;   /* linetype*/
            Gfloat      linewidth;/* linewidth scale factor*/
            Gcolr_specif      line_colr_specif;/* line colour specifier*/
            Gint  marker_ind; /* marker index*/
            Gint  marker_type;/* marker type*/
            Gfloat      marker_size;/* marker size scale factor*/
            Gcolr_specif      marker_colr_specif;/* marker colour specifier*/
            Gint  area_ind;   /* area ind*/
            Gint_style  int_style;/* interior style*/
            Gint  int_style_ind;/* interior style index*/
            Gcolr_specif      int_colr_specif;/* interior colour specifier*/
            Gedge_flag  edge_flag;/* edge flag */
            Gint  edgetype;   /* edgetype */
            Gfloat      edgewidth;/* edgewidth scale factor */
            Gcolr_specif      edge_colr_specif;/* edge colour specifier */
            Gint  text_ind;   /* text index*/
            Gtext_font_prec   text_font_prec;/* text font
                        and precision*/
            Gfloat      char_expan;/* character expansion factor*/
            Gfloat      char_space;/* character spacing*/
            Gcolr_specif      text_colr_specif;/* text colour specifier*/
            Gasfs asfs; /* asfs*/
            Gint  norm_tran_num;/* current norm. tran.
                  number   */
            Gtran norm_tran[64];/* list of norm. tran. */
            Gint  vp_in_pri[64];/* list of vp input pris*/
            Gint  name_open_pic_part;/* name of open pic. part*/
            Gint_list   pic_part_names;/* list of picture parts
                  in use   */
            Gint  name_open_seg;/* name of open segment*/
            Gint_list   seg_names;/* list of segments
                  in use   */
            Gseg_sl_list      seg_sls;/* list of segment
                        state lists */
            Gint  name_open_stencil;/* name of open stencil*/
            Gint_list   stencil_names;/* list of stencils
                  in use   */
            Gstencil_sl_list  stencil_sls;/* list of stencil
                        state lists in use*/
            Gcont_attrs cont_attrs;/* contour attributes*/
            Gint  name_open_tiling;/* name of open tiling*/
            Gint_list   tiling_names;/* list of tilings in use*/

 } value;
} Ggks_sl_entry;
/* Ggks_sl_entry_list GKS state list entry list */
typedef struct {
 Gint        num_gks_sl_entries;/* number of entries in list*/
 Ggks_sl_entry     *gks_sl_entries;/* list of entries*/
} Ggks_sl_entry_list;
/* Ggks_sl_name_list GKS state list name list */
typedef struct {
 Gint        num_gks_sl_names; /* number of names in list*/
 Ggks_sl_name      *gks_sl_names;/* list of names*/
} Ggks_sl_name_list;
/* Gcolr_chars  colour characteristics */
typedef struct {
 enum Gcolr_chars_type {GTYPE_CIELUV, GTYPE_OTHER } type;
 union {
           Gcieluv     prim_colr[3];/* CIELUV primary colours */
```

```
            Gdata impl_dep;   /* impl.dep. */
 } chars;
} Gcolr_chars;
/* Gcolr_facs colour facilities */
typedef struct {
 Gint        num_colr_models;  /* number of colour models*/
 Gint        colr_models[4];   /* list of available colour models*/
 Gint        num_colrs;  /* number of colours*/
 Gcolr_chars       chars;      /* colour characteristics*/
 Gcolr_avail       colr_avail; /* colour availability*/
 Gint        num_pred_inds;    /* number of predef. colour indices*/
} Gcolr_facs;
/* Gcolr_rep colour representation */
typedef union {
 Grgb        rgb;  /* Red Green Blue colour specification */
 Gcieluv     cieluv;     /* CIE L*u*v* 1976 colour specification */
 Ghls        hls;  /* Hue Luminance Saturation colour specification */
 Ghsv        hsv;  /* Hue Saturation Value colour specification */
                   /* etc. */
} Gcolr_rep;
/* Garea_bundle area bundle */
typedef struct {
 Gint_style int_style;  /* interior style*/
 Gint        style_ind;  /* interior style index*/
 Gcolr_kind colr_kind;  /* interior colour kind*/
 Gint        colr_ind;   /* interior colour index*/
 Gcolr_rep  int_dir_colr;     /* direct interior colour*/
 Gedge_flag edge_flag;  /* edge flag*/
 Gint        edgetype;   /* edgetype*/
 Gfloat      edgewidth;  /* edgewidth scale factor*/
 Gcolr_kind edge_colr_kind;   /* edge colour kind*/
 Gint        edge_colr_ind;    /* edge colour index*/
 Gcolr_rep  edge_dir_colr;    /* direct edge colour*/
} Garea_bundle;
/* Garea_attrs area attributes */
typedef struct {
 Gasf        int_style_asf;    /* interior style ASF */
 Gasf        style_ind_asf;    /* style index ASF */
 Gasf        colr_ind_asf;     /* colour specifier ASF */
 Gasf        edge_flag_asf;    /* edge flag ASF */
 Gasf        edgetype_asf;     /* edgetype ASF */
 Gasf        edgewidth_asf;    /* edgewidth scale factor ASF */
 Gasf        edge_colr_ind_asf;/* edge colour specifier ASF */
 Gint        ind;  /* area index*/
 Garea_bundle      bundle;     /* area bundle*/
} Garea_attrs;
/* Gline_bundle line bundle */
typedef struct {
 Gint        type; /* linetype */
 Gfloat      width;      /* linewidth scale factor*/
 Gcolr_kind colr_kind;  /* colour kind*/
 Gint        colr_ind;   /* colour index*/
 Gcolr_rep  dir_colr;   /* direct colour*/
} Gline_bundle;
/* Gline_attrs line attributes */
typedef struct {
 Gasf       type_asf;   /* linetype ASF */
 Gasf       width_asf;  /* linewidth scale factor ASF */
 Gasf       colr_ind_asf;     /* line colour specifier ASF */
 Gint       ind;  /* line index*/
 Gline_bundle     bundle;     /* line bundle*/
} Gline_attrs;
/* Gmarker_bundle marker bundle */
typedef struct {
 Gint       type; /* marker type*/
 Gfloat     size; /* marker size scale factor*/
 Gcolr_kind colr_kind;  /* colour kind*/
 Gint       colr_ind;   /* colour index*/
 Gcolr_rep  dir_colr;   /* direct colour*/
} Gmarker_bundle;
/* Gmarker_attrs marker attributes */
typedef struct {
 Gasf       type_asf;   /* marker type ASF */
 Gasf       size_asf;   /* marker size scale factor ASF */
 Gasf       colr_ind_asf;     /* marker colour specifier ASF */
 Gint       ind;  /* marker index*/
 Gmarker_bundle   bundle;     /* marker bundle*/
} Gmarker_attrs;
/* Garea_bundle_table area bundle table */
typedef struct {
 Gint       num_inds;   /* number of area indices*/
 Gint       *inds;      /* list of area indices*/
 Garea_bundle     *bundles;   /* list of area bundles*/
} Garea_bundle_table;
/* Gline_bundle_table line bundle table */
typedef struct {
 Gint       num_inds;   /* number of line indices*/
 Gint       *inds;      /* list of line indices*/
 Gline_bundle     *bundles;   /* list of line bundles*/
} Gline_bundle_table;
/* Gmarker_bundle_table marker bundle table */
typedef struct {
 Gint       num_inds;   /* number of marker indices*/
 Gint       *inds;      /* list of marker indices*/
 Gmarker_bundle   *bundles;   /* list of marker bundles*/
} Gmarker_bundle_table;
/* Gtext_bundle text bundle */
typedef struct {
 Gtext_font_prec  text_font_prec;/* text font and precision*/
 Gfloat     char_expan; /* character expansion factor*/
 Gfloat     char_space; /* character spacing*/
 Gcolr_kind colr_kind;  /* colour kind*/
 Gint       colr_ind;   /* colour index*/
 Gcolr_rep  dir_colr;   /* direct colour*/
```

```
} Gtext_bundle;
/* Gtext_bundle_table text bundle table */
typedef struct {
 Gint         num_inds;    /* number of character indices*/
 Gint         *inds;       /* list of character indices*/
 Gtext_bundle      *bundles;    /* list of text bundles*/
} Gtext_bundle_table;
/* Grep representation  */
typedef struct {
 Gint ind; /* index */
 enum Grep_design { /* representation designation */
           GREP_LINE,  /* line bundle*/
           GREP_MARKER,      /* marker bundle*/
           GREP_AREA,  /* area bundle*/
           GREP_PAT,   /* pattern representation*/
           GREP_TEXT,  /* text bundle*/
           GREP_COLR,  /* colr specification*/
           GREP_UNDEF  /* undefined*/
  } design;
 union {
           Gline_bundle       line_bundle;/* line bundle*/
           Gmarker_bundle     marker_bundle;/* marker bundle*/
           Gtext_bundle       text_bundle;/* text bundle*/
           Garea_bundle       area_bundle;/* area bundle*/
           Gpat_rep     pat_rep;/* pattern representation*/
           Gcolr_specif       colr_specif;/* colour specification*/
 } value;
} Grep;
/* Grepl_tech replication technique */
typedef struct {
 enum Grepl_dir {
           GREPL_DX,
           GREPL_DY,
           GREPL_DXY,
           GREPL_DYX
 } dir;
 union {
           Gfloat      dx;    /* dx*/
           Gfloat      dy;    /* dy*/
           Gvec  dxy;  /* dxy */
           Gvec  dyx;  /* dyx */
 } value;
} Grepl_tech;
/* Gcolr_table colour table */
typedef struct {
 Gint         num_inds;    /* number of colour indices*/
 Gint         *inds;       /* list of colour indices*/
 Gcolr_specif      *colr_specifs;/* list of colour specifications*/
} Gcolr_table;
/* Implementation Dependent Types*/
/* Gchoice_data  choice data   */
typedef struct {
 union Gchoice_pets {
           struct Gchoice_pet_r1 {
                 Gdata impl_dep;/* impl. dep. */
           } pet_r1;   /* pet 1 data */
           struct Gchoice_pet_r2 {
                 Gint  num_prs;/* number of prompts */
                 Gpr_flag    *prs;/* prompt array */
                 Gdata impl_dep;/* impl. dep. */
           } pet_r2;   /* pet 2 data */
           struct Gchoice_pet_r3 {
                 Gint  num_strings;/* number of choice strings */
                 char  **strings;/* array of choice strings */
                 Gdata impl_dep;/* impl. dep. */
           } pet_r3;   /* pet 3 data */
           struct Gchoice_pet_r4 {
                 Gint  num_strings;/* number of choice strings */
                 char  **strings;/* array of choice strings */
                 Gdata impl_dep;/* impl. dep. */
           } pet_r4;   /* pet 4 data */
           struct Gchoice_pet_r5 {
                 Gint  seg_name;/* segment name */
                 Gint  num_pick_ids;/* number of pick identifiers */
                 Gint  *pick_ids;/* array of pick identifiers */
                 Gdata impl_dep;/* impl. dep. */
           } pet_r5;   /* pet 5 data */
           struct Gchoice_pet_u1 {
                 Gdata impl_dep;/* impl. dep. */
           } pet_u1;   /* pet -1 data */
           /*. . . impl. defined PET's */
            } pets;
} Gchoice_data;
/* Gescape_in_data  escape input data record */
typedef union {
 struct Gescape_in_r1 {
           Gdata impl_dep;   /* impl. dep. */
 } escape_r1;     /* escape 1 data */
 struct Gescape_in_u1 {
           Gdata impl_dep;   /* impl. dep. */
 } escape_u1;     /* escape -1 data */
 /* etc. */
} Gescape_in_data;
/* Gescape_out_data  escape output data record */
typedef union {
 struct Gescape_out_r1 {
           Gdata impl_dep;   /* impl. dep. */
 } escape_r1;     /* escape 1 data */
 struct Gescape_out_u1 {
           Gdata impl_dep;   /* impl. dep. */
 } escape_u1;     /* escape -1 data */
 /* etc. */
} Gescape_out_data;
/* Ggdp_data  gdp data record */
```

```
typedef union {
 struct Ggdp_r1 {
            Gdata impl_dep;   /* impl. dep. */
 } gdp_r1;  /* gdp 1 data */
 struct Ggdp_u1 {
            Gdata impl_dep;   /* impl. dep. */
 } gdp_u1;  /* gdp -1 data */
 /* etc. */
} Ggdp_data;
/* Gloc_data  locator  data record */
typedef struct {
 union Gloc_pets {
            struct Gloc_pet_r1 {
                  Gdata impl_dep;/* impl. dep. */
            } pet_r1;    /* pet 1 data */
            struct Gloc_pet_r2 {
                  Gdata impl_dep;/* impl. dep. */
            } pet_r2;    /* pet 2 data */
            struct Gloc_pet_r3 {
                  Gdata impl_dep;/* impl. dep. */
            } pet_r3;    /* pet 3 data */
            struct Gloc_pet_r4 {
                  Gattr_ctrl_flagattr_ctrl_flag;
                  /* attributecontrol flag */
                  Gline_attrs line_attrs;
                  /* line      attributes */
                  Gdata impl_dep;/* impl. dep. */
            } pet_r4;    /* pet 4 data */
            struct Gloc_pet_r5 {
                  Gattr_ctrl_flagattr_ctrl_flag;
                  /* attributecontrol flag */
                  Gline_fill_ctrl_flagline_fill_ctrl_flag;
                  /* line/area control flag */
                  union Gloc_attrs {
                        Gline_attrsline_attrs;
                        /* lineattrs. */
                        Garea_attrsarea_attrs;
                        /* interiorattrs. */
                  } attrs;
                  Gdata impl_dep;/* impl. dep. */
            } pet_r5;    /* pet 5 data */
            struct Gloc_pet_r6 {
                  Gdata impl_dep;/* impl. dep. */
            } pet_r6;    /* pet 6 data */
            struct Gloc_pet_u1 {
                  Gdata impl_dep;/* impl. dep. */
            } pet_u1;    /* pet -1 data */
            /*. . . impl. defined PET's */
 } pets;
} Gloc_data;
/* Gpick_data  pick data  */
typedef struct {
 union Gpick_pets {
           struct Gpick_pet_r1 {
                Gdata impl_dep;/* impl. dep. */
           } pet_r1;    /* pet 1 data */
           struct Gpick_pet_r2 {
                Gint  pick_id;/* pick identifier*/
                Gint  seg_name;/* segment name*/
                Gnameset    set_names;/* set of names*/
                Gdata impl_dep;/* impl. dep. */
           } pet_r2;    /* pet 2 data */
           struct Gpick_pet_r3 {
                Gnameset    set_names;/* set of names*/
                Gdata impl_dep;/* impl. dep. */
           } pet_r3;    /* pet 3 data */
           struct Gpick_pet_u1 {
                Gdata data; /* data */
           } pet_u1;    /* pet -1 data */
           /* etc. */
  } pets;
} Gpick_data;
/* Gstring_data  string data record */
typedef struct {
 Gint      in_buf_size;      /* input buffer size (nr. of bytes) */
 Gint      init_pos;    /* initial [cursor] position */
 union Gstring_pets {
           struct Gstring_pet_r1 {
                Gdata impl_dep;/* impl. dep. */
           } pet_r1;    /* pet 1 data */
           struct Gstring_pet_u1 {
                Gdata impl_dep;/* impl. dep. */
           } pet_u1;    /* pet -1 data */
           /* etc. */
  } pets;
} Gstring_data;
/* Gstroke_data  stroke data  */
typedef struct {
 Gint      init_pos;    /* initial buffer editing position */
 Gint      in_buf_size;      /* input buffer size (nr. of points) */
 Gfloat    x_interval; /* X interval */
 Gfloat    y_interval; /* Y interval */
 Gfloat    time_interval;    /* time interval */
 union Gstroke_pets {
           struct Gstroke_pet_r1 {
                Gdata impl_dep;/* impl. dep. */
           } pet_r1;    /* pet 1 data */
           struct Gstroke_pet_r2 {
                Gdata impl_dep;/* impl. dep. */
           } pet_r2;    /* pet 2 data */
           struct Gstroke_pet_r3 {
                Gattr_ctrl_flagattr_ctrl_flag;
                /* attributecontrol flag */
                Gmarker_attrsmarker_attrs;
                /* marker   attrs. */
                Gdata impl_dep;/* impl. dep. */
          } pet_r3;   /* pet 3 data */
          struct Gstroke_pet_r4 {
                Gattr_ctrl_flagattr_ctrl_flag;
                /* attributecontrol flag */
                Gline_attrs line_attrs;
                /* line     attrs. */
                Gdata impl_dep;/* impl. dep. */
          } pet_r4;   /* pet 4 data */
          struct Gstroke_pet_u1 {
                Gdata impl_dep;/* impl. dep. */
          } pet_u1;   /* pet -1 data */
          /* etc. */
 } pets;
} Gstroke_data;
/* Gval_data  valuator data record */
typedef struct {
 Gfloat     low_value;  /* low value */
 Gfloat     high_value; /* high value */
 union Gval_pets {
          struct Gval_pet_r1 {
                Gdata impl_dep;/* impl. dep. */
          } pet_r1;   /* pet 1 data */
          struct Gval_pet_r2 {
                Gdata impl_dep;/* impl. dep. */
          } pet_r2;   /* pet 2 data */
          struct Gval_pet_r3 {
                Gdata impl_dep;/* impl. dep. */
          } pet_r3;   /* pet 3 data */
          struct Gval_pet_u1 {
                Gdata impl_dep;/* impl. dep. */
          } pet_u1;   /* pet -1 data */
          /* etc. */
  } pets;
} Gval_data;
/* Gaudit_user_data  audut user data record */
typedef struct {
 Gint       id;   /* identifier*/
 union {
          Gdata impl_dep;    /* impl. dependent*/
 } data;
} Gaudit_user_data;
/* Gbasic_in_data basic input data */
typedef struct{
 Gin_class class; /* basic input [measure] class */
 union {
          Gloc_data   loc_data;/* locator data */
          Gstroke_data      stroke_data;/* stroke data */
          Gval_data   val_data;/* valuator data */
          Gchoice_data      choice_data;/* choice data */
          Gpick_data  pick_data;/* pick data */
```

```
            Gstring_data      string_data;/* string data */
 } in_data; /* input data */
} Gbasic_in_data;
/* Gcomp_data composite data */
typedef struct {
 Gint        num_devs;   /* number of [basic] devices */
 Gbasic_in_data   *basic_in_data;/* list of basic input data */
} Gcomp_data;
/* Gdev_id_in_value_list device identifier and input value list */
typedef struct {
 Gint        num_devs;   /* number of [basic] devices in list*/
 Gdev_id     *dev_ids;   /* list of device id values*/
 Gin_value   *in_values; /* list of device input values*/
} Gdev_id_in_value_list;
/* Gin_data input data */
typedef union {
 Gloc_data   loc_data;   /* locator data*/
 Gstroke_data      stroke_data;/* strokedata*/
 Gval_data   val_data;   /* valuator data*/
 Gchoice_data      choice_data;/* choice data*/
 Gpick_data pick_data;   /* pick data*/
 Gstring_data      string_data;/* string data*/
 Gcomp_data comp_data;   /* composite data*/
} Gin_data;
/* Ginit_in initial input */
typedef struct {
 Gin_value   in_value;   /* input value*/
 Gint        pet;   /* prompt and echo type */
 Gecho_switch      echo_switch;/* echo switch */
 Glimit      echo_area;   /* echo area*/
 Gin_data    in_data;    /* input data*/
} Ginit_in;
/* Ginit_value initial value */
typedef struct {
 Ginit_sel sel;  /* initial selector */
 union {
            Gin_value   init_value;/* input value*/
            Gint  pet;  /* prompt and echo type */
            Gecho_switch      echo_switch;/* echo switch */
            Glimit      echo_area;/* echo area*/
            Gin_data    in_data;/* input data*/
            Ginit_in    all;  /* all initial input*/
 } value;
} Ginit_value;
/* Gin_dev_init_value input device initial value */
typedef struct {
 Gdev_id     dev_id;     /* device identifier*/
 Ginit_value      init_value; /* initial value*/
} Gin_dev_init_value;
/* Gin_dev_init_value_list input initial value list */
typedef struct {
 Gint        num_devs;   /* number of input devices*/
 Gin_dev_init_value     *init_values;/* list of input initial values*/
} Gin_dev_init_value_list;
/* Gop_modes_init_values operating modes and initial values */
typedef struct {
 Gint       num_in_devs;      /* number of input devices*/
 Gdev_id    dev_ids;     /* device identifiers*/
 Gop_mode   op_modes;    /* operating modes*/
 Ginit_in   init_ins;    /* initial input values*/
} Gop_modes_init_values;
/* Gprim_params primitive parameters */
typedef struct {
 Gprim_type type; /* primitive type */
 union {
           Gpoint_list_list  polyline_set;/* polyline set*/
           Gnurb_list  nurb_set;/* NURB set*/
           Gconic_sec_list   conic_sec_set;/* conic section set*/

           Gpoint_list polymarker;/* polymarker */
           Gpoint_list_list  fill_area_set;/* fill area set*/
           Gnurb_list  closed_nurb_set;/* closed NURB set*/
           Gconic_sec_list   ell_sec_set;/* elliptic sector set*/
           Gconic_sec_list   ell_seg_set;/* elliptic segment set*/
           Gell_disc_list    ell_disc_set;/* elliptic disc set*/
           struct Gtext {
                 char   *string;/* character string */
                 Gpoint      pos;/* text position */
           } text;
           struct Gcell_array {
                 Grect rect; /* cell rectangle */
                 Gpat_rep    array;/* cell array */
           } cell_array;
           Gdesign     design;/* design */
           struct Ggdp {
                 Gpoint_list point_list;/* point_list */
                 Gint   id;    /* gdp identifier */
                 Ggdp_data   gdp_data;/* gdp_data */
           } gdp;
 } value;
} Gprim_params;
/* Gtiling_params tiling parameters */
typedef struct {
 Gtiling_type     type; /* tiling type */
 union {
           struct Gtiling_line_area { /* line or area primitives in tiling */
                 Gcolr_specifcolr_specif;/* colour specifier */
                 union { /* geometric data of line or area primitive */
                       Gpoint_list_listpolyline_set;/* polyline set*/
                       Gnurb_listnurb_set;/* NURB set*/
                       Gconic_sec_listconic_sec_set;/* conic section set*/
                       Gpoint_list_listfill_area_set;/* fill area set*/
                       Gnurb_listclosed_nurb_set;/* closed NURB set*/
                       Gconic_sec_listell_sec_set;/* elliptic sector set*/
```

```
                        Gconic_sec_listell_seg_set;/* elliptic segment set*/
                        Gell_disc_listell_disc_set;/* elliptic disc set*/
                  } geom_data;
            } tiling_line_area;
            Gdesign     design;/* design */
 } value;
} Gtiling_params;
/* Gws_type workstation type */
typedef struct {
 Gint       gen_type;   /* generic type*/
 void       *specif_info;     /* specific information*/
} Gws_type;
/* Gws_dt_entry workstation description table entry */
typedef struct {
 Gws_dt_name      name; /* entry name */
 union {
            Gws_type    type; /* workstation type */
            Gws_status  ws_status;/* ws. status*/
            Gdc_units   dc_units;/* DC units*/
            Gdisp_space_size  disp_space_size;/* displace
                           space size*/
            Gws_class   ws_class;/* ws. classification*/

            Gline_facs  line_facs;/* line facilities*/
            Gline_bundle_tablepred_line_bundles;/* predefined line
                          bundle table*/

            Gmarker_facs      marker_facs;/* marker
                           facilities*/
            Gmarker_bundle_tablepred_marker_bundles;/* predefined marker
                           bundle table*/

            Garea_facs  area_facs;/* area facilities*/
            Garea_bundle_tablepred_area_bundles;/* predefined area
                           bundle table*/

            Gpat_table  pred_pat_reps;/* predefined pattern
                           bundle table*/

            Gtext_facs  text_facs;/* text facilities*/
            Gtext_bundle_tablepred_text_bundles;/* predefined text
                           bundle table*/

            Gcolr_facs  colr_facs;/* colour facilities*/
            Gcolr_table pred_colr_reps;/* predefined colour
                           bundle table*/
            Gint_list   gdp_ids;/* list of available
                           GDPs*/
            Gdev_id_list      dev_ids;/* list of available
                           input devices*/
            Gin_dev_init_value_listin_init_values;/* input device
                           initial values*/
            Gint_list   meas_ids;/* list of available
                              measures*/
            Gint_list   trigger_ids;/* list of available
                              triggers*/
            Gdev_comp_list    dev_comps;/* device compositions*/
 } value;
} Gws_dt_entry;
/* Gws_sl_entry workstation state list entry */
typedef struct {
 Gws_sl_name      name; /* entry name */
 union {
            Gws_st      ws_st;/* ws. state
                              (ACTIVE|INACTIVE) */
            struct {
                  void  *specif;
                  Gint  type;
            } specif_type; /* specifier and type */
            Gline_bundle_tableline_bundle_table;/* line bundle
                              table*/
            Gmarker_bundle_tablemarker_bundle_table;/* marker
                              bundle table*/
            Gtext_bundle_tabletext_bundle_table;/* text
                              bundle table*/
            Garea_bundle_tablearea_bundle_table;/* area
                              bundle table*/
            Gpat_table  pat_table;/* pattern
                              bundle table*/
            Gcolr_table colr_table;/* colour
                              bundle table*/
            Gtran ws_win_vp;  /* window
                              and viewport*/
            Gvis_effects_st   vis_effects_st;/* visual effects
                              state*/
            Gop_modes_init_valuesop_modes_init_values;/* input device
                              operating modes
                              and initial values*/
            Gsel_table  sel_table;/* selection table*/
            Gint_list   stored_segs;/* set of
                              stored segments*/
            Gview_rep   view_rep[16];/* view bundle table */
            Gint  view_pri[16];/* view priorities*/
 } entry;
} Gws_sl_entry;
/* Gws_sl_entry_list workstation state list entry value list */
typedef struct {
 Gint       num_entries;      /* number of WSL entry values
                     in list  */
 Gws_sl_entry     *entries;   /* list of WSL entry values*/
} Gws_sl_entry_list;
/* Gws_sl_name_list workstation state list entry name list */
typedef struct {
 Gint       num_names;  /* number of WSL entry names in list*/
 Gws_sl_name      *names;     /* list of WSL entry names*/
} Gws_sl_name_list;
/* Gfunc_params function parameters */
typedef struct {
 Gint        func_id;
 union {
struct Gopen_gks {
 char        *err_file;
} open_gks;
struct Gsave_gks_sl {
 Ggks_sl_name_list      entry_names;
 Ggks_sl_entry_list     entries;
} save_gks_sl;
struct Grestore_gks_sl {
 Ggks_sl_entry_list     entries;
} restore_gks_sl;
struct Gescape {
 Gint        func_id;
 Gescape_in_data  in_data;
 Gescape_out_data out_data;
} escape;
struct Groute {
 Groute_dir dir;
} route;
struct Gcreate_out_prim {
 Gprim_params     prim_params;
} create_out_prim;
struct Gopen_stencil {
 Gint        stencil_name;
 Ginside_rule     inside_rule;
} open_stencil;
struct Grename_stencil {
 Gint        old_name;
 Gint        new_name;
} rename_stencil;
struct Gdel_stencil {
 Gint        name;
} del_stencil;
struct Gcreate_stencil_bndry {
 Gint        stencil_name;
 Ginside_rule     inside_rule;
 Gbndry_list      bndry;
} create_stencil_bndry;
struct Gcreate_stencil_conts {
 Gint        stencil_name;
 Ginside_rule     inside_rule;
 Gpath_list path_seq;
} create_stencil_conts;
struct Gset_cont_attr {
 Gcont_attr_value cont_attr;
} set_cont_attr;
struct Gset_stencil_attr {
 Gint        stencil_name;
 Gstencil_attr_value    stencil_attr;
} set_stencil_attr;
struct Gget_stencil_attr {
 Gint        stencil_name;
 Gstencil_attr_name     attr_name;
 Gstencil_attr_value    attr;
} get_stencil_attr;
struct Ginst_stencil {
 Gint        stencil_name;
 Gint        inst_name;
 Gtran_matrix     stencil_tran;
 Ginst_specif     inst_specif;
} inst_stencil;
struct Ginst_stencil_path {
 Gint        stencil_name;
 Gint        inst_name;
 Gpoint      start_map;
 Gpoint      end_map;
 Gint        inst_num;
 Gpath       path;
} inst_stencil_path;
struct Ginst_stencil_seq_path {
 Gseq_specif_list inst_seq_specif;
 Gpath_specif     path_specif;
 Gtran_matrix     stencil_tran;
} inst_stencil_seq_path;
struct Gopen_tiling {
 Gint        tiling_name;
} open_tiling;
struct Grename_tiling {
 Gint        old_name;
 Gint        new_name;
} rename_tiling;
struct Gdel_tiling {
 Gint        name;
} del_tiling;
struct Gcreate_tiling_comp {
 Gtiling_params   tiling_params;
 Gpoint      origin;
 Grepl_tech repl_tech;
} create_tiling_comp;
struct Gset_prim_attr {
 Gprim_attr_value attr_value;
} set_prim_attr;
struct Gadd_set_names_nameset {
 Gnameset    set_names;
} add_set_names_nameset;
struct Gremove_set_names_nameset {
 Gnameset    set_names;
} remove_set_names_nameset;
struct Gadd_set_scissors_scissor_set {
 Gscissor_list    scissor_set;
} add_set_scissors_scissor_set;
struct Gremove_set_scissors_scissor_set {
 Gint_list  scissor_ids;
} remove_set_scissors_scissor_set;
struct Gset_win_vp {
 Gint        tran_num;
 Gtran       norm_tran;
} set_win_vp;
struct Gset_vp_in_pri {
 Gint        tran_num;
 Gint        ref_tran_num;
 Grel_pri    rel_pri;
} set_vp_in_pri;
struct Gset_scissor_mode {
 Gscissor_mode    scissor_mode;
} set_scissor_mode;
struct Gset_norm_tran_num {
 Gint        tran_num;
} set_norm_tran_num;
struct Gdel_prims_ndc_pic {
 Gsel_crit  sel_crit;
} del_prims_ndc_pic;
struct Gadd_set_names_ndc_pic {
 Gnameset    set_names;
 Gsel_crit  sel_crit;
} add_set_names_ndc_pic;
struct Gremove_set_names_ndc_pic {
 Gnameset    set_names;
 Gsel_crit  sel_crit;
} remove_set_names_ndc_pic;
struct Gset_ndc_pic_prim_attr {
 Gsel_crit  sel_crit;
 Gprim_attr_value attr_value;
} set_ndc_pic_prim_attr;
struct Gadd_set_scissors_ndc_pic {
 Gscissor_list    scissor_set;
 Gsel_crit  sel_crit;
} add_set_scissors_ndc_pic;
struct Gremove_set_scissors_ndc_pic {
 Gint_list  scissor_ids;
 Gsel_crit  sel_crit;
} remove_set_scissors_ndc_pic;
struct Greorder_ndc_pic {
 Gsel_crit  source_sel_crit;
 Gsel_crit  ref_sel_crit;
 Grel_pos    rel_pos;
} reorder_ndc_pic;
struct Gcopy_ndc_pic_ndc_mf {
 void        *mf_specif;
 Gint        pic_id;
 Gsel_crit  sel_crit;
 Gnameset   set_names;
} copy_ndc_pic_ndc_mf;
struct Gcopy_ndc_mf_pic_ndc_pic {
 void       *mf_specif;
 Gint       pic_id;
 Gnameset   set_names;
} copy_ndc_mf_pic_ndc_pic;
struct Gopen_pic_part {
 Gint       pic_part_name;
} beg_pic_part;
struct Gar_pic_part {
 Gint       pic_part_name;
 void       *ar_specif;
 Gint       ar_name_pic_part;
} ar_pic_part;
struct Gret_pic_part_ar {
 void       *ar_specif;
 Gint       ar_name_pic_part;
 Gint       pic_part_name;
} ret_pic_part_ar;
struct Greopen_pic_part {
 Gint       pic_part_name;
} reopen_pic_part;
struct Gappend_pic_part {
 Gint       source_pic_part_name;
 Gint       sink_pic_part_name;
 Gtran_matrix     global_tran_matrix;
 Gtran_mode global_tran_mode;
 Gtran_matrix     local_tran_matrix;
 Gtran_mode local_tran_mode;
 Gnameset   set_names;
} append_pic_part;
struct Grename_pic_part {
 Gint       old_pic_part_name;
 Gint       new_pic_part_name;
} rename_pic_part;
struct Gdel_pic_part {
 Gint       pic_part_name;
} del_pic_part;
struct Gcopy_pic_part_pic_part_store {
 Gint       pic_part_name;
 Gsel_crit  sel_crit;
 Gtran_matrix     global_tran_matrix;
 Gtran_mode global_tran_mode;
 Gtran_matrix     local_tran_matrix;
 Gtran_mode local_tran_mode;
 Gnameset   set_names;
 Gscissor_sel     scissor_sel;
} copy_pic_part_pic_part_store;
struct Gcopy_ndc_pic_pic_part_store {
 Gsel_crit  sel_crit;
 Gint       pic_part_name;
 Gnameset   set_names;
} copy_ndc_pic_pic_part_store;
struct Gset_in_dev_mode {
 Gdev_id    dev_id;
 Gop_mode   op_mode;
} set_in_dev_mode;
struct Greq_in {
 Gdev_id    dev_id;
 Gin_status in_status;
 Gin_value  log_in_value;
} req_in;
struct Gsample_in {
 Gdev_id    dev_id;
 Gin_value  log_in_value;
} sample_in;
struct Gawait_in {
 Gfloat     timeout;
 Gdev_id_in_value_list  dev_id_values;
} await_in;
struct Gflush_dev_events {
 Gdev_id    dev_id;
} flush_dev_events;
struct Gset_font_ind_map {
 Gfont_ind_map    font_ind_map;
} set_font_ind_map;
struct Gget_glyph_name {
 Gint       font_ind;
 char       character;
 Gint       glyph_name;
} get_glyph_name;
struct Gset_char_code {
 Gint       font_ind;
 char       character;
 Gint       glyph_name;
} set_char_code;
struct Geval_tran_matrix {
 Gpoint     point;
 Gvec       shift;
 Gfloat     angle;
 Gvec       scale;
 Gcoord_switch    coord_switch;
 Gtran_matrix     tran_matrix;
} eval_tran_matrix;
struct Gaccum_tran_matrix {
 Gtran_matrix     matrix;
 Gpoint     point;
 Gvec       shift;
 Gfloat     angle;
 Gvec       scale;
 Gcoord_switch    coord_switch;
 Gtran_matrix     tran_matrix;
} accum_tran_matrix;
```

```
struct Geval_circle {
 Gpoint      ctr;
 Gfloat      radius;
 Gconic_matrix     circle;
} eval_circle;
struct Geval_ell {
 Gpoint      ctr;
 Gpoint      conj_radius_end_point[2];
 Gconic_matrix     ell;
} eval_ell;
struct Geval_circ_arc_3_point {
 Gpoint      circ_point[3];
 Gconic_sec circ_arc;
} eval_circ_arc_3_point;
struct Geval_circ_arc_ctr {
 Gpoint      ctr;
 Gvec        vec[2];
 Gsense_flag       sense_flag;
 Gfloat      radius;
 Gconic_sec circ_arc;
} eval_circ_arc_ctr;
struct Geval_ell_arc {
 Gpoint      ctr;
 Gpoint      conj_radius_end_point[2];
 Gvec        vec[2];
 Gconic_sec conic_arc;
} eval_ell_arc;
struct Geval_hyp_arc {
 Gpoint      ctr;
 Gpoint      tran_radius_end_point;
 Gpoint      conj_radius_end_point;
 Gvec        vec[2];
 Gconic_sec conic_arc;
} eval_hyp_arc;
struct Geval_par_arc {
 Gpoint      intersec_point;
 Gpoint      start_point;
 Gpoint      end_point;
 Gconic_sec conic_arc;
} eval_par_arc;
struct Geval_wc_ord {
 Gfloat      wc_ord;
 Gord_sel    ord_sel;
 Gfloat      ndc_value;
} eval_wc_ord;
struct Gopen_ws {
 Gint        ws_id;
 void        *ws_specif_info;
 Gint        ws_gen_type;
} open_ws;
struct Gclose_ws {
 Gint        ws_id;
```

```
} close_ws;
struct Gsave_ws_sl {
 Gint       ws_id;
 Gws_sl_name_list entry_names;
 Gws_sl_entry_list      entry_list;
} save_ws_sl;
struct Grestore_ws_sl {
 Gint       ws_id;
 Gws_sl_entry_list      entries;
} restore_ws_sl;
struct Gget_ws_status {
 Gint       ws_id;
 Gws_status ws_status;
} get_ws_status;
struct Gremove_backdrop {
 Gint       ws_id;
} remove_backdrop;
struct Gset_rep {
 Gint       ws_id;
 Grep       rep_value;
} set_rep;
struct Gset_view_pri {
 Gint       ws_id;
 Gint       view_ind;
 Gint       ref_view_ind;
 Grel_pri   rel_pri;
} set_view_pri;
struct Gset_view_sel_crit {
 Gint       ws_id;
 Gint       view_ind;
 Gsel_crit  sel_crit;
} set_view_sel_crit;
struct Gset_view {
 Gint       ws_id;
 Gint       ind;
 Gtran_matrix     ori_matrix;
 Gtran_matrix     map_matrix;
 Gscissor   scissor;
} set_view;
struct Gset_ws_vis_effects {
 Gint       ws_id;
 Gvis_effects_st  vis_effects_st;
} set_ws_vis_effects;
struct Gset_ws_win_vp {
 Gint       ws_id;
 Gtran      ws_tran;
} set_ws_win_vp;
struct Gdef_in_dev {
 Gdev_id    dev_id;
 Gint_list  meas_set;
 Gint_list  trigger_set;
 Ginit_value      init_value;
```

```
} def_in_dev;
struct Ginit_in_dev {
 Gdev_id    dev_id;
 Ginit_value      init_value;
} init_in_dev;
struct Gset_ws_sel_crit {
 Gint       ws_id;
 Gsel       sel;
} set_ws_sel_crit;
struct Gcopy_real_pic_real_mf {
 Gint       ws_id;
 void       *mf_specif;
 Gint       pic_id;
} copy_real_pic_real_mf;
struct Gcopy_blank_real_pic_real_mf {
 Gint       ws_id;
 void       *mf_specif;
 Gint       pic_id;
} copy_blank_real_pic_real_mf;
struct Gcopy_real_mf_pic_backdrop {
 Gint       ws_id;
 void       *mf_specif;
 Gint       pic_id;
} copy_real_mf_pic_backdrop;
struct Gmessage {
 Gint       ws_id;
 char       *message;
} message;
struct Greset_specif_ws_dt_entry_st {
 Gint       ws_id;
 Gws_dt_name      ws_dt_name;
} reset_specif_ws_dt_entry_st;
struct Gget_text_extent {
 Gint       ws_id;
 Gpoint     pos;
 char       *str;
 Gtext_extent     extent;
} get_text_extent;
struct Geval_view_ori_matrix {
 Gpoint     ref_point;
 Gvec       view_up;
 Gtran_matrix     view_ori_matrix;
} eval_view_ori_matrix;
struct Geval_view_map_matrix {
 Gtran      win_vp;
 Gtran_matrix     view_map_matrix;
} eval_view_map_matrix;
struct Gconv_colr {
 Gint       ws_id;
 Gcolr_specif     colr_specif;
 Gint       colr_model;
 Gcolr_specif     colr_coords;
 Gconv_flag conv_flag;
} conv_colr;
struct Gcreate_seg {
 Gint        seg_name;
} create_seg;
struct Grename_seg {
 Gint        old_seg_name;
 Gint        new_seg_name;
} rename_seg;
struct Gdel_seg {
 Gint        seg_name;
} del_seg;
struct Gdel_seg_ws {
 Gint        ws_id;
 Gint        seg_name;
} del_seg_ws;
struct Gassoc_seg_ws {
 Gint        ws_id;
 Gint        seg_name;
} assoc_seg_ws;
struct Gcopy_seg_ws {
 Gint        ws_id;
 Gint        seg_name;
} copy_seg_ws;
struct Ginsert_seg {
 Gint        seg_name;
 Gtran_matrix      tran_matrix;
} insert_seg;
struct Gset_seg_attr {
 Gint        seg_name;
 Gseg_attr_value  seg_attr_value;
} set_seg_attr;
struct Gactivate_ws {
 Gint        ws_id;
} activate_ws;
struct Gdeactivate_ws {
 Gint        ws_id;
} deactivate_ws;
struct Gclear_ws {
 Gint        ws_id;
 Gctrl_flag ctrl_flag;
} clear_ws;
struct Gget_map_seg_name {
 Gint        seg_name;
 Gint        map_seg_name;
} get_map_seg_name;
struct Gget_map_ws_id {
 Gint        ws_id;
 Gint        map_ws_id;
} get_map_ws_id;
  } func_data;
} Gfunc_params;
```

A.2 Macros

```
/* Macros*/
/* Error macros*/
#define  GE_NO_ERR                            (0)
#define  GE_GKS_NOT_OPEN                      (1001)
#define  GE_GKS_OPEN                          (1002)
#define  GE_PP_NOT_OPEN                       (1003)
#define  GE_PP_OPEN                           (1004)
#define  GE_SEG_NOT_OPEN                      (1005)
#define  GE_SEG_OPEN                          (1006)
#define  GE_ST_NOT_OPEN                       (1007)
#define  GE_ST_OPEN                           (1008)
#define  GE_TL_NOT_OPEN                       (1009)
#define  GE_TL_OPEN                           (1010)
#define  GE_WS_TYPE_NOT_SUPP                  (1011)
#define  GE_WS_NOT_OPEN                       (1012)
#define  GE_WS_OPEN                           (1013)
#define  GE_WS_NOT_ACTIVE                     (1014)
#define  GE_WS_ACTIVE                         (1015)
#define  GE_AUDIT_NOT_OPEN                    (1016)
#define  GE_AUDIT_OPEN                        (1017)
#define  GE_PLAYBACK_NOT_OPEN                 (1018)
#define  GE_PLAYBACK_OPEN                     (1019)
#define  GE_PP_NOT_EXIST                      (1020)
#define  GE_PP_NAME_USE                       (1021)
#define  GE_OLD_PP_NOT_EXIST                  (1022)
#define  GE_NEW_PP_NAME_USE                   (1023)
#define  GE_SRC_PP_NOT_EXIST                  (1024)
#define  GE_SINK_PP_NOT_EXIST                 (1025)
#define  GE_SEG_NOT_EXIST                     (1026)
#define  GE_SEG_NAME_IN_USE                   (1027)
#define  GE_ST_NOT_EXIST                      (1028)
#define  GE_ST_NAME_USE                       (1029)
#define  GE_OLD_ST_NOT_EXIST                  (1030)
#define  GE_NEW_ST_NAME_USE                   (1031)
#define  GE_INST_NAME_EQ_ST_NAME_USE          (1032)
#define  GE_TL_NOT_EXIST                      (1033)
#define  GE_TL_NAME_USE                       (1034)
#define  GE_OLD_TL_NOT_EXIST                  (1035)
#define  GE_NEW_TL_NAME_USE                   (1036)
#define  GE_IN_Q_OVERF                        (1037)
#define  GE_IN_Q_NOT_OVERF                    (1038)
#define  GE_IN_Q_OVERF_WS_CLOSED              (1039)
#define  GE_ESCAPE_FUNC_NOT_SUPP              (1040)
#define  GE_GLYPH_NOT_AVAIL                   (1041)
#define  GE_FONT_NAME_NOT_AVAIL               (1042)
#define  GE_LINETYPE_NOT_SUPP                 (1043)
#define  GE_MARKER_TYPE_NOT_SUPP              (1044)
#define  GE_FONT_NOT_SUPP_PREC                (1045)
#define  GE_INT_STYLE_NOT_SUPP                (1046)
#define  GE_HATCH_STYLE_NOT_SUPP              (1047)
#define  GE_EDGETYPE_NOT_SUPP                 (1048)
```

```
#define  GE_COLR_MODEL_NOT_SUPP                      (1049)
#define  GE_GDP_NOT_SUPP                             (1050)
#define  GE_IN_DEV_NOT_EXIST                         (1051)
#define  GE_COMB_MEAS_TRIG_NOT_AVAIL                 (1052)
#define  GE_SEG_NOT_EXIST_WS                         (1053)
#define  GE_PIC_NOT_EXIST_IN_MF                      (1054)
#define  GE_NO_ITEM_PLAYBACK                         (1055)
#define  GE_PLAYBACK_ITEM_INVAL                      (1056)
#define  GE_PP_NOT_EXIST_AR                          (1057)
#define  GE_EDGE_NOT_INTERSECT_PATH                  (1058)
#define  GE_START_VEC_NOT_INTERSECT_HYP              (1059)
#define  GE_END_VEC_NOT_INTERSECT_HYP                (1060)
#define  GE_OLD_PP_NAME_EQ_NEW_PP_NAME               (1061)
#define  GE_OLD_ST_NAME_EQ_NEW_ST_NAME               (1062)
#define  GE_OLD_TL_NAME_EQ_NEW_TL_NAME               (1063)
/*
 *       Binding specific errors
 */
#define  GE_START_IND_INVAL                          (2201)
#define  GE_APPL_LIST_LENGTH_LT_ZERO                 (2202)
#define  GE_ENUM_TYPE_INVAL                          (2203)
#define  GE_ALLOC_STORE                              (2204)
#define  GE_ALLOC_MEM_STORE                          (2205)
#define  GE_ERR_FILE_INVAL                           (2250)
#define  GE_INSUFF_STORE                             (2251)
#define  GE_ENTRIES_INVAL                            (2252)
#define  GE_ESC_FUNC_ID_INVAL                        (2253)
#define  GE_ESC_IN_DATA_INVAL                        (2254)
#define  GE_ERR_NUM_INVAL                            (2255)
#define  GE_FUNC_NAME_INVAL                          (2256)
#define  GE_DIR_INVAL                                (2257)
#define  GE_PRIM_TYPE_INVAL                          (2258)
#define  GE_NUM_POINTS_INVAL                         (2259)
#define  GE_CODE_INVAL                               (2260)
#define  GE_DIMS_COLR_ARRAY_INVAL                    (2261)
#define  GE_GDP_ID_INVAL                             (2262)
#define  GE_GDP_DATA_INVAL                           (2263)
#define  GE_NURB_DEF_INVAL                           (2264)
#define  GE_CONIC_SEC_INVAL                          (2265)
#define  GE_INSIDE_RULE_INVAL                        (2266)
#define  GE_SEQ_PATHS_INVAL                          (2267)
#define  GE_CONT_ATTR_NAME_INVAL                     (2268)
#define  GE_CONT_ATTR_VALUE_INVAL                    (2269)
#define  GE_ST_NAME_INVAL                            (2270)
#define  GE_OLD_NEW_ST_NAME_INVAL                    (2271)
#define  GE_ST_ATTR_NAME_INVAL                       (2272)
#define  GE_BNDRY_SPECIF_INVAL                       (2273)
#define  GE_ST_ATTR_VALUE_INVAL                      (2274)
#define  GE_INST_NAME_INVAL                          (2275)
#define  GE_INST_TYPE_INVAL                          (2276)
#define  GE_INST_SPECIF_INVAL                        (2277)
#define  GE_PATH_INVAL                               (2278)
```

```
#define  GE_TL_NAME_INVAL                         (2279)
#define  GE_REPL_TECH_INVAL                       (2280)
#define  GE_ATTR_NAME_INVAL                       (2281)
#define  GE_ATTR_VALUE_INVAL                      (2282)
#define  GE_SET_NAMES_INVAL                       (2283)
#define  GE_SCISSOR_SPECIF_INVAL                  (2284)
#define  GE_SCISSOR_IDS_INVAL                     (2285)
#define  GE_SCISSOR_MODE_INVAL                    (2286)
#define  GE_TRAN_NUM_INVAL                        (2287)
#define  GE_WIN_DEF_INVAL                         (2288)
#define  GE_VP_DEF_INVAL                          (2289)
#define  GE_SEL_CRIT_INVAL                        (2290)
#define  GE_MF_SPECIF_INVAL                       (2291)
#define  GE_PIC_ID_INVAL                          (2292)
#define  GE_PP_NAME_INVAL                         (2293)
#define  GE_AR_SPECIF_INVAL                       (2294)
#define  GE_AR_NAME_PP_INVAL                      (2295)
#define  GE_SRC_PP_NAME_INVAL                     (2296)
#define  GE_SINK_PP_NAME_INVAL                    (2297)
#define  GE_GLOBAL_TRAN_MODE_INVAL                (2298)
#define  GE_LOCAL_TRAN_MODE_INVAL                 (2299)
#define  GE_OLD_PP_NAME_INVAL                     (2300)
#define  GE_NEW_PP_NAME_INVAL                     (2301)
#define  GE_SCISSOR_SEL_INVAL                     (2302)
#define  GE_IN_DEV_INVAL                          (2303)
#define  GE_OP_MODE_INVAL                         (2304)
#define  GE_TIMEOUT_INVAL                         (2305)
#define  GE_FONT_IND_ZERO                         (2306)
#define  GE_FONT_NAME_INVAL                       (2307)
#define  GE_CHAR_CODE_INVAL                       (2308)
#define  GE_AUDIT_ID_INVAL                        (2309)
#define  GE_AUDIT_OP_INVAL                        (2310)
#define  GE_AUDIT_USER_DATA_INVAL                 (2311)
#define  GE_PLAYBACK_OP_INVAL                     (2312)
#define  GE_PROC_OP_INVAL                         (2313)
#define  GE_ENTRY_NAME_INVAL                      (2314)
#define  GE_FIXED_POINT_INVAL                     (2315)
#define  GE_SHIFT_VEC_INVAL                       (2316)
#define  GE_RADIUS_INVAL                          (2317)
#define  GE_RADII_ID                              (2318)
#define  GE_POINTS_COLLIN                         (2319)
#define  GE_SENSE_FLAG_INVAL                      (2320)
#define  GE_ORD_SEL_INVAL                         (2321)
#define  GE_WS_ID_INVAL                           (2322)
#define  GE_WS_TYPE_INVAL                         (2323)
#define  GE_WS_SPECIF_INVAL                       (2324)
#define  GE_REP_INVAL                             (2325)
#define  GE_IND_INVAL                             (2326)
#define  GE_TYPE_ZERO                             (2327)
#define  GE_SCALE_FAC_NEG                         (2328)
#define  GE_CHAR_EXPAN_NOT_POS                    (2329)
#define  GE_EDGE_FLAG_INVAL                       (2330)
```

```
#define  GE_INT_STYLE_INVAL                     (2331)
#define  GE_COLR_IND_INVAL                      (2332)
#define  GE_PAT_IND_INVAL                       (2333)
#define  GE_COLR_SPECIF_INVAL                   (2334)
#define  GE_VIEW_IND_INVAL                      (2335)
#define  GE_REF_IND_INVAL                       (2336)
#define  GE_REL_PRI_INVAL                       (2337)
#define  GE_VIEW_SCISSOR_INVAL                  (2338)
#define  GE_VIS_EFFECTS_ST_INVAL                (2339)
#define  GE_WS_WIN_LIMITS_INVAL                 (2340)
#define  GE_WS_VP_LIMITS_INVAL                  (2341)
#define  GE_WS_VP_NOT_IN_DISP_SPACE             (2342)
#define  GE_MEAS_SEQ_INVAL                      (2343)
#define  GE_TRIGGER_SET_INVAL                   (2344)
#define  GE_ECHO_SWITCH_INVAL                   (2345)
#define  GE_ECHO_AREA_RECT_INVAL                (2346)
#define  GE_SEL_TYPE_INVAL                      (2347)
#define  GE_TYPE_RETURNED_VALUES_INVAL          (2348)
#define  GE_WS_GEN_TYPE_INVAL                   (2349)
#define  GE_COLR_MODEL_INVAL                    (2350)
#define  GE_SEG_NAME_INVAL                      (2351)
#define  GE_OLD_NEW_SEG_NAME_INVAL              (2352)
#define  GE_SEG_ATTR_VALUE_INVAL                (2353)

/* Function identifier macros*/
#define  Gfn_open_gks                           (0)
#define  Gfn_close_gks                          (1)
#define  Gfn_save_gks_sl                        (200)
#define  Gfn_restore_gks_sl                     (201)
#define  Gfn_escape                             (11)
#define  Gfn_emergency_close_gks                (153)
#define  Gfn_err_hand                           (154)
#define  Gfn_err_log                            (155)
#define  Gfn_route                              (202)
#define  Gfn_create_out_prim                    (203)
#define  Gfn_polyline                           (12)
#define  Gfn_polyline_set                       (204)
#define  Gfn_nurb_set                           (205)
#define  Gfn_conic_sec_set                      (206)
#define  Gfn_polymarker                         (13)
#define  Gfn_fill_area                          (15)
#define  Gfn_fill_area_set                      (207)
#define  Gfn_ell_sec_set                        (208)
#define  Gfn_ell_seg_set                        (209)
#define  Gfn_ell_disc_set                       (210)
#define  Gfn_closed_nurb_set                    (211)
#define  Gfn_text                               (14)
#define  Gfn_cell_array                         (16)
#define  Gfn_design                             (212)
#define  Gfn_gdp                                (17)
#define  Gfn_open_stencil                       (213)
#define  Gfn_close_stencil                      (214)
```

```
#define  Gfn_rename_stencil                          (215)
#define  Gfn_del_stencil                             (216)
#define  Gfn_create_stencil_bndry                    (217)
#define  Gfn_create_stencil_conts                    (218)
#define  Gfn_set_cont_attr                           (219)
#define  Gfn_set_stencil_attr                        (220)
#define  Gfn_get_stencil_attr                        (221)
#define  Gfn_inst_stencil                            (222)
#define  Gfn_inst_stencil_path                       (223)
#define  Gfn_inst_stencil_seq_path                   (224)
#define  Gfn_open_tiling                             (225)
#define  Gfn_close_tiling                            (226)
#define  Gfn_rename_tiling                           (227)
#define  Gfn_del_tiling                              (228)
#define  Gfn_create_tiling_comp                      (229)
#define  Gfn_set_prim_attr                           (230)
#define  Gfn_set_line_ind                            (18)
#define  Gfn_set_linetype                            (19)
#define  Gfn_set_linewidth                           (20)
#define  Gfn_set_line_colr_specif                    (231)
#define  Gfn_set_line_colr_ind                       (21)
#define  Gfn_set_marker_ind                          (22)
#define  Gfn_set_marker_type                         (23)
#define  Gfn_set_marker_size                         (24)
#define  Gfn_set_marker_colr_specif                  (232)
#define  Gfn_set_marker_colr_ind                     (25)
#define  Gfn_set_text_ind                            (26)
#define  Gfn_set_text_font_prec                      (27)
#define  Gfn_set_char_expan                          (28)
#define  Gfn_set_char_space                          (29)
#define  Gfn_set_text_colr_specif                    (233)
#define  Gfn_set_text_colr_ind                       (30)
#define  Gfn_set_char_ht                             (31)
#define  Gfn_set_char_up_vec                         (32)
#define  Gfn_set_text_path                           (33)
#define  Gfn_set_text_align                          (34)
#define  Gfn_set_text_skew_angle                     (234)
#define  Gfn_set_area_ind                            (235)
#define  Gfn_set_int_style                           (236)
#define  Gfn_set_int_style_ind                       (237)
#define  Gfn_set_int_colr_specif                     (238)
#define  Gfn_set_int_colr_ind                        (239)
#define  Gfn_set_pat_size                            (39)
#define  Gfn_set_pat_ref_point                       (40)
#define  Gfn_set_global_tran_matrix                  (240)
#define  Gfn_set_local_tran_matrix                   (241)
#define  Gfn_set_edge_flag                           (242)
#define  Gfn_set_edgetype                            (243)
#define  Gfn_set_edgewidth                           (244)
#define  Gfn_set_edge_colr_specif                    (245)
#define  Gfn_set_edge_colr_ind                       (246)
#define  Gfn_set_asfs                                (41)
```

```
#define  Gfn_set_pick_id                          (42)
#define  Gfn_set_nameset                          (247)
#define  Gfn_set_scissor_set                      (248)
#define  Gfn_add_set_names_nameset                (249)
#define  Gfn_remove_set_names_nameset             (250)
#define  Gfn_add_set_scissors_scissor_set         (251)
#define  Gfn_remove_set_scissors_scissor_set      (252)
#define  Gfn_set_win_vp                           (49)
#define  Gfn_set_win                              (49)
#define  Gfn_set_vp                               (50)
#define  Gfn_set_vp_in_pri                        (50)
#define  Gfn_set_scissor_mode                     (253)
#define  Gfn_set_norm_tran_num                    (254)
#define  Gfn_sel_norm_tran                        (52)
#define  Gfn_del_prims_ndc_pic                    (255)
#define  Gfn_remove_set_names_ndc_pic             (256)
#define  Gfn_add_set_names_ndc_pic                (257)
#define  Gfn_set_ndc_pic_prim_attr                (258)
#define  Gfn_add_set_scissors_ndc_pic             (259)
#define  Gfn_remove_set_scissors_ndc_pic          (260)
#define  Gfn_reorder_ndc_pic                      (261)
#define  Gfn_copy_ndc_pic_ndc_mf                  (262)
#define  Gfn_copy_ndc_mf_pic_ndc_pic              (263)
#define  Gfn_open_pic_part                        (264)
#define  Gfn_close_pic_part                       (265)
#define  Gfn_ar_pic_part                          (266)
#define  Gfn_ret_pic_part_ar                      (267)
#define  Gfn_reopen_pic_part                      (268)
#define  Gfn_append_pic_part                      (269)
#define  Gfn_rename_pic_part                      (270)
#define  Gfn_del_pic_part                         (271)
#define  Gfn_copy_pic_part_pic_part_store         (272)
#define  Gfn_copy_ndc_pic_pic_part_store          (273)
#define  Gfn_set_in_dev_mode                      (274)
#define  Gfn_set_loc_mode                         (75)
#define  Gfn_set_stroke_mode                      (76)
#define  Gfn_set_val_mode                         (77)
#define  Gfn_set_choice_mode                      (78)
#define  Gfn_set_pick_mode                        (79)
#define  Gfn_set_string_mode                      (80)
#define  Gfn_req_in                               (275)
#define  Gfn_req_loc                              (81)
#define  Gfn_req_stroke                           (82)
#define  Gfn_req_val                              (83)
#define  Gfn_req_choice                           (84)
#define  Gfn_req_pick                             (85)
#define  Gfn_req_string                           (86)
#define  Gfn_sample_in                            (276)
#define  Gfn_sample_loc                           (87)
#define  Gfn_sample_stroke                        (88)
#define  Gfn_sample_val                           (89)
#define  Gfn_sample_choice                        (90)
```

```
#define  Gfn_sample_pick                          (91)
#define  Gfn_sample_string                        (92)
#define  Gfn_await_in                             (277)
#define  Gfn_await_event                          (93)
#define  Gfn_flush_dev_events                     (278)
#define  Gfn_get_loc                              (95)
#define  Gfn_get_stroke                           (96)
#define  Gfn_get_val                              (97)
#define  Gfn_get_choice                           (98)
#define  Gfn_get_pick                             (99)
#define  Gfn_get_string                           (100)
#define  Gfn_set_font_ind_map                     (279)
#define  Gfn_get_glyph_name                       (280)
#define  Gfn_set_char_code                        (281)
#define  Gfn_audit                                (282)
#define  Gfn_write_user_rec_audit                 (283)
#define  Gfn_playback                             (284)
#define  Gfn_read_item_func_name_audit            (285)
#define  Gfn_read_item_audit                      (286)
#define  Gfn_proc_audit_item                      (287)
#define  Gfn_eval_tran_matrix                     (105)
#define  Gfn_accum_tran_matrix                    (106)
#define  Gfn_eval_circle                          (288)
#define  Gfn_eval_ell                             (289)
#define  Gfn_eval_circ_arc_3_point                (290)
#define  Gfn_eval_circ_arc_ctr                    (291)
#define  Gfn_eval_ell_arc                         (292)
#define  Gfn_eval_hyp_arc                         (293)
#define  Gfn_eval_par_arc                         (294)
#define  Gfn_eval_wc_ord                          (295)
#define  Gfn_set_err_hand                         (156)
#define  Gfn_create_store                         (296)
#define  Gfn_del_store                            (297)
#define  Gfn_open_ws                              (2)
#define  Gfn_close_ws                             (3)
#define  Gfn_save_ws_sl                           (298)
#define  Gfn_restore_ws_sl                        (299)
#define  Gfn_get_ws_status                        (300)
#define  Gfn_remove_backdrop                      (301)
#define  Gfn_set_rep                              (302)
#define  Gfn_set_line_rep                         (43)
#define  Gfn_set_marker_rep                       (44)
#define  Gfn_set_text_rep                         (45)
#define  Gfn_set_area_rep                         (303)
#define  Gfn_set_pat_rep                          (47)
#define  Gfn_set_colr_rep                         (48)
#define  Gfn_set_view_pri                         (304)
#define  Gfn_set_view_sel_crit                    (305)
#define  Gfn_set_view                             (306)
#define  Gfn_set_ws_vis_effects                   (307)
#define  Gfn_set_ws_win_vp                        (308)
#define  Gfn_set_ws_win                           (54)
```

```
#define  Gfn_set_ws_vp                          (55)
#define  Gfn_def_in_dev                         (309)
#define  Gfn_init_in_dev                        (310)
#define  Gfn_init_loc                           (69)
#define  Gfn_init_stroke                        (70)
#define  Gfn_init_val                           (71)
#define  Gfn_init_choice                        (72)
#define  Gfn_init_pick                          (73)
#define  Gfn_init_string                        (74)
#define  Gfn_set_ws_sel_crit                    (311)
#define  Gfn_copy_real_pic_real_mf              (312)
#define  Gfn_copy_blank_real_pic_real_mf        (313)
#define  Gfn_copy_real_mf_pic_backdrop          (314)
#define  Gfn_message                            (10)
#define  Gfn_reset_specif_ws_dt_entry_st        (315)
#define  Gfn_get_text_extent                    (316)
#define  Gfn_eval_view_ori_matrix               (317)
#define  Gfn_eval_view_map_matrix               (318)
#define  Gfn_conv_colr                          (319)
#define  Gfn_create_seg                         (56)
#define  Gfn_close_seg                          (57)
#define  Gfn_rename_seg                         (58)
#define  Gfn_del_seg                            (59)
#define  Gfn_del_seg_ws                         (60)
#define  Gfn_assoc_seg_ws                       (61)
#define  Gfn_copy_seg_ws                        (62)
#define  Gfn_insert_seg                         (63)
#define  Gfn_set_seg_attr                       (320)
#define  Gfn_set_seg_tran                       (64)
#define  Gfn_set_vis                            (65)
#define  Gfn_set_highl                          (66)
#define  Gfn_set_det                            (68)
#define  Gfn_set_seg_pri                        (67)
#define  Gfn_activate_ws                        (4)
#define  Gfn_deactivate_ws                      (5)
#define  Gfn_clear_ws                           (6)
#define  Gfn_get_map_seg_name                   (321)
#define  Gfn_get_map_ws_id                      (322)
/* Miscellaneous macros*/
#define  GLINE_SOLID                            (1)
#define  GLINE_DASH                             (2)
#define  GLINE_DOT                              (3)
#define  GLINE_DASH_DOT                         (4)
#define  GLINE_DASH_DOT_DOT                     (5)
#define  GMARKER_DOT                            (1)
#define  GMARKER_PLUS                           (2)
#define  GMARKER_ASTERISK                       (3)
#define  GMARKER_CIRCLE                         (4)
#define  GMARKER_CROSS                          (5)
#define  GHATCH_HOR                             (1)
#define  GHATCH_VERT                            (2)
#define  GHATCH_DIAG_45                         (3)
```

```
#define  GHATCH_DIAG_135                    (4)
#define  GHATCH_HOR_VERT                    (5)
#define  GHATCH_DIAG_45_135                 (6)
#define  GMODEL_RGB                         (1)
#define  GMODEL_CIELUV                      (2)
#define  GMODEL_HSV                         (3)
#define  GMODEL_HLS                         (4)
#define  GLOC_DEF                           (1)
#define  GLOC_CROSS_HAIR                    (2)
#define  GLOC_TRACK_CROSS                   (3)
#define  GLOC_RUB_BAND                      (4)
#define  GLOC_RECT                          (5)
#define  GLOC_DIGIT                         (6)
#define  GSTROKE_DEF                        (1)
#define  GSTROKE_DIGIT                      (2)
#define  GSTROKE_MARKER                     (3)
#define  GSTROKE_LINE                       (4)
#define  GVAL_DEF                           (1)
#define  GVAL_GRAPH                         (2)
#define  GVAL_DIGIT                         (3)
#define  GCHOICE_DEF                        (1)
#define  GCHOICE_PR_ECHO                    (2)
#define  GCHOICE_STRING_PR                  (3)
#define  GCHOICE_STRING_IN                  (4)
#define  GCHOICE_SEG                        (5)
#define  GPICK_DEF                          (1)
#define  GPICK_GROUP_HIGHL                  (2)
#define  GPICK_SEG_HIGHL                    (3)
#define  GPICK_SET_NAMES_HIGHL              (3)
#define  GSTRING_DEF                        (1)
#define  GDEF_ERR_FILE                      ((char*)0)
```

A.3 Function calls

```
/* GKS Functions
/* Control functions*/
extern void gopen_gks(
 const char *err_file,  /* name of error file*/
 size_t     mem_units   /* number of units of memory available
                   for buffer space*/);
extern void gclose_gks(
 void                   );
extern void gsave_gks_sl(
 const Ggks_sl_name_list     *entry_names,/* GKS state list entry
                   names     */
 Gstore     store,      /* handle to Store object*/
 Ggks_sl_entry_list     **entries/* OUT GKS state
                   list entries */);
extern void grestore_gks_sl(
 const Ggks_sl_entry_list     *entries/*  GKS state list entries */);
extern void gescape(
 Gint        func_id,    /* escape function identifier*/
 const Gescape_in_data  *in_data,/* escape input data record*/
 Gstore      store,       /* handle to Store object*/
 Gescape_out_data **out_data  /* OUT  escape output data
                           record    */);
extern void gemergency_close_gks(
 void                     );
extern void gerr_hand(
 Gint        err_num,     /* error number*/
 Gint        func_num,    /* number of function that detected the error*/
 const char *err_file    /* name of error file*/);
extern void gerr_log(
 Gint        err_num,     /* error number*/
 Gint        func_num,    /* number of function that detected the error*/
 const char *err_file    /* name of error file*/);
extern void groute(
 Groute_dir dir     /* direction*/);
/* Output functions*/
extern void gcreate_out_prim(
 const Gprim_params      *prim_params/* primitive parameters*/);
extern void gpolyline(
 const Gpoint_list       *point_list/* list of points*/);
extern void gpolyline_set(
 const Gpoint_list_list *point_list_list/* list of point lists*/);
extern void gnurb_set(
 const Gnurb_list *nurb_set    /* NURB parameter lists*/);
extern void gconic_sec_set(
 const Gconic_sec_list  *conic_sec_set/* Conic section set*/);
extern void gpolymarker(
 const Gpoint_list       *point_list/* list of points*/);
extern void gfill_area(
 const Gpoint_list       *point_list/* list of points*/);
extern void gfill_area_set(
 const Gpoint_list_list *point_list_list/* list of point lists*/);
extern void gclosed_nurb_set(
 const Gnurb_list *closed_nurb_set/* NURB parameter lists*/);
extern void gell_sec_set(
 const Gconic_sec_list  *ell_sec_set/* elliptic sector set*/);
extern void gell_seg_set(
 const Gconic_sec_list  *ell_seg_set/* elliptic segment set*/);
extern void gell_disc_set(
 const Gell_disc_list   *ell_disc_set/* elliptic disc set*/);
extern void gtext(
 const Gpoint      *text_pos,   /* text position*/
 const char *char_string       /* character string*/);
extern void gcell_array(
 const Grect       *rect,       /* cell rectangle*/
 const Gpat_rep    *colr_array /* colour array*/);
extern void gdesign(
 const Gdesign     *design      /* design*/);
extern void ggdp(
 const Gpoint_list       *point_list,/* list of points*/
 Gint         gdp_id,      /* gdp identifier*/
 const Ggdp_data   *gdp_data    /* gdp data record*/);
```

```
/* Design output functions*/
extern void gopen_stencil(
 Gint        stencil_name,     /* stencil name*/
 Ginside_rule     inside_rule /* inside rule*/);
extern void gclose_stencil(
 void);
extern void grename_stencil(
 Gint        old_name,    /* old stencil name*/
 Gint        new_name     /* new stencil name*/);
extern void gdel_stencil(
 Gint        name   /* stencil name*/);
extern void gcreate_stencil_bndry(
 Gint        stencil_name,     /* stencil name*/
 Ginside_rule     inside_rule,/* inside rule*/
 const Gbndry_list      *bndry/* boundary*/);
extern void gcreate_stencil_conts(
 Gint        stencil_name,     /* stencil name*/
 Ginside_rule     inside_rule,/* inside rule*/
 const Gpath_list *path_seq    /* path sequence*/);
extern void gset_cont_attr(
 const Gcont_attr_value *cont_attr/* contour attribute*/);
extern void gset_stencil_attr(
 Gint        stencil_name,     /* stencil name*/
 const Gstencil_attr_value    *stencil_attr/* stencil attribute*/);
extern void gget_stencil_attr(
 Gint        stencil_name,     /* stencil name*/
 Gstencil_attr_name     attr_name,/* stencil attribute name*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gstencil_attr_value    *attr /* OUT stencil attribute*/);
extern void ginst_stencil(
 Gint        stencil_name,     /* stencil name*/
 Gint        inst_name,   /* instance name*/
 Gtran_matrix     stencil_tran,/* stencil transformation*/
 const Ginst_specif     *inst_specif/* instance specifier*/);
extern void ginst_stencil_path(
 Gint        stencil_name,     /* stencil name*/
 Gint        inst_name,   /* instance name*/
 const Gpoint     *start_map, /* start map*/
 const Gpoint     *end_map,   /* end map*/
 Gint        inst_num,    /* instance number*/
 const Gpath      *path_def   /* path definition*/);
extern void ginst_stencil_seq_path(
 const Gseq_specif_list *inst_seq_specif,/* instance sequence
                    specifier*/
 const Gpath_specif     *path_specif,/* path specifier*/
 Gtran_matrix     stencil_tran/* stencil transformation*/);
extern void gopen_tiling(
 Gint        tiling_name /* tiling name*/);
extern void gclose_tiling(
 void);
extern void grename_tiling(
 Gint        old_name,    /* old tiling name*/
 Gint        new_name    /* new tiling name*/);
extern void gdel_tiling(
 Gint        name  /* tiling name*/);
extern void gcreate_tiling_comp(
 const Gtiling_params   *tiling_params,/* tiling parameters*/
 const Gpoint      *origin,     /* origin */
 const Grepl_tech *repl_tech  /* replication technique*/);
extern void gset_prim_attr(
 const Gprim_attr_value *attr_value/* attr_value */);
extern void gset_pick_id(
 Gint        pick_id      /* pick identifier*/);
extern void gset_nameset(
 const Gnameset    *nameset     /* nameset */);
extern void gset_scissor_set(
 const Gscissor_list    *scissor_set/* scissor_set */);
extern void gset_global_tran_matrix(
 Gtran_matrix      global_tran_matrix/* global transformation
                      matrix    */);
extern void gset_local_tran_matrix(
 Gtran_matrix      local_tran_matrix/* local transformation
                      matrix    */);
extern void gset_pat_size(
 const Gfloat_size      *pat_size/* pattern size*/);
extern void gset_pat_ref_point(
 const Gpoint      *pat_ref_point/* pattern reference point*/);
extern void gset_char_ht(
 Gfloat      char_ht      /* character height*/);
extern void gset_char_up_vec(
 const Gvec *char_up_vec      /* character up vector*/);
extern void gset_text_skew_angle(
 Gfloat      text_skew_angle   /* text skew angle*/);
extern void gset_text_path(
 Gtext_path text_path    /* text path*/);
extern void gset_text_align(
 const Gtext_align      *text_align/* text alignment*/);
extern void gset_line_ind(
 Gint        line_ind     /* line index*/);
extern void gset_linetype(
 Gint        linetype     /* linetype*/);
extern void gset_linewidth(
 Gfloat      linewidth    /* linewidth scale factor*/);
extern void gset_line_colr_specif(
 const Gcolr_specif     *line_colr_specif/* line colour specifier*/);
extern void gset_line_colr_ind(
 Gint        line_colr_ind     /* line colour index*/);
extern void gset_marker_ind(
 Gint        marker_ind   /* marker index*/);
extern void gset_marker_type(
 Gint        marker_type /* marker type*/);
extern void gset_marker_size(
 Gfloat      marker_size /* marker size scale factor*/);
extern void gset_marker_colr_specif(
 const Gcolr_specif     *marker_colr_specif/* marker colour
                     specifier*/);
extern void gset_marker_colr_ind(
 Gint        marker_colr_ind    /* marker colour index*/);
extern void gset_area_ind(
 Gint        area_ind     /* area index*/);
extern void gset_int_style(
 Gint_style int_style    /* interior style*/);
extern void gset_int_style_ind(
 Gint        int_style_ind      /* interior style index*/);
extern void gset_int_colr_specif(
 const Gcolr_specif     *int_colr_specif/* interior colour specifier*/);
extern void gset_int_colr_ind(
 Gint        int_colr_ind       /* interior colour index*/);
extern void gset_edge_flag(
 Gedge_flag edge_flag    /* edge flag*/);
extern void gset_edgetype(
 Gint        edgetype     /* edgetype*/);
extern void gset_edgewidth(
 Gfloat      edgewidth    /* edgewidth scale factor*/);
extern void gset_edge_colr_specif(
 const Gcolr_specif     *edge_colr_specif/* edge colour specifier*/);
extern void gset_edge_colr_ind(
 Gint        edge_colr_ind      /* edge colour index*/);
extern void gset_text_ind(
 Gint        text_ind     /* text index*/);
extern void gset_text_font_prec(
 const Gtext_font_prec  *text_font_prec/* text font and precision */);
extern void gset_char_expan(
 Gfloat      char_expan   /* character expansion factor*/);
extern void gset_char_space(
 Gfloat      char_space   /* character spacing*/);
extern void gset_text_colr_specif(
 const Gcolr_specif     *text_colr_specif/* text colour specifier*/);
extern void gset_text_colr_ind(
 Gint        text_colr_ind      /* text colour index*/);
extern void gset_asfs(
 const Gasfs       *asfs /* list of attribute source flags */);
extern void gadd_set_names_nameset(
 const Gnameset     *set_names   /* set of names */);
extern void gremove_set_names_nameset(
 const Gnameset     *set_names   /* set of names */);
extern void gadd_set_scissors_scissor_set(
 const Gscissor_list     *set_scissors/*set of scissors */);
extern void gremove_set_scissors_scissor_set(
 const Gint_list   *scissor_ids/* scissor identifiers */);
/* Normalization transformation functions*/
extern void gset_win_vp(
 Gint        tran_num,    /* transformation number*/
 const Gtran       *win_wp      /* window and viewport limits*/);
extern void gset_win(
 Gint        tran_num,    /* transformation number*/
 const Glimit     *win  /* window limits*/);
extern void gset_vp(
 Gint        tran_num,   /* transformation number*/
 const Glimit     *vp   /* viewport limits*/);
extern void gset_vp_in_pri(
 Gint        tran_num,   /* transformation number*/
 Gint        ref_tran_num,     /* reference transformation number*/
 Grel_pri    rel_pri     /* relative priority*/);
extern void gset_scissor_mode(
 Gscissor_mode    scissor_mode/* scissor mode */);
extern void gset_norm_tran_num(
 Gint        tran_num    /* transformation number*/);
extern void gsel_norm_tran(
 Gint        tran_num    /* transformation number*/);
/* NDC picture functions*/
extern void gdel_prims_ndc_pic(
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void gadd_set_names_ndc_pic(
 const Gnameset   *set_names, /* set of names*/
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void gremove_set_names_ndc_pic(
 const Gnameset   *set_names, /* set of names*/
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void gset_ndc_pic_prim_attr(
 const Gsel_crit  *sel_crit,  /* selection criterion*/
 const Gprim_attr_value *attr_value/* attr_value */);
extern void gadd_set_scissors_ndc_pic(
 const Gscissor_list    *set_scissors,/* scissor set*/
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void gremove_set_scissors_ndc_pic(
 const Gint_list  *scissor_ids,/* scissor identifiers*/
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void greorder_ndc_pic(
 const Gsel_crit  *source_sel_crit,/* source selection criterion*/
 const Gsel_crit  *ref_sel_crit,/* reference selection criterion*/
 Grel_pos    rel_pos     /* relative position*/);
/* Metafile functions*/
extern void gcopy_ndc_pic_ndc_mf(
 const void *mf_specif, /* metafile specifier*/
 Gint        pic_id,     /* picture identifier*/
 const Gsel_crit  *sel_crit,  /* selection criterion*/
 const Gnameset   *set_names  /* set of names*/);
extern void gcopy_ndc_mf_pic_ndc_pic(
 const void *mf_specif, /* metafile specifier*/
 Gint        pic_id,     /* picture identifier*/
 const Gnameset   *set_names  /* set of names*/);
/* Picture part store functions*/
extern void gopen_pic_part(
 Gint        pic_part_name     /* picture part name*/);
extern void gclose_pic_part(
 void                    );
extern void gar_pic_part(
 Gint       pic_part_name,    /* picture part name*/
 const void *ar_specif, /* archive specifier*/
 Gint       ar_name_pic_part  /* archive name of picture part*/);
extern void gret_pic_part_ar(
 const void *ar_specif, /* archive specifier*/
 Gint       ar_name_pic_part, /* archive name of picture part*/
 Gint       pic_part_name     /* picture part name*/);
extern void greopen_pic_part(
 Gint       pic_part_name     /* picture part name*/);
extern void gappend_pic_part(
 Gint       source_pic_part_name,/* source
                      picture part name*/
 Gint       sink_pic_part_name,/* sink
                      picture part name*/
 const Gtran_matrix     global_tran_matrix,/* global
                      transformation matrix*/
 Gtran_mode global_tran_mode, /* global
                      transformation mode*/
 const Gtran_matrix     local_tran_matrix,/* local
                      transformation matrix*/
 Gtran_mode local_tran_mode,  /* local
                      transformation mode*/
 const Gnameset   *set_names  /* set of names*/);
extern void grename_pic_part(
 Gint       old_pic_part_name,/* old picture part name*/
 Gint       new_pic_part_name /* new picture part name*/);
extern void gdel_pic_part(
 Gint       pic_part_name     /* picture part name*/);
extern void gcopy_pic_part_pic_part_store(
 Gint       pic_part_name,    /* picture part name*/
 const Gsel_crit  *sel_crit,  /* selection criterion*/
 const Gtran_matrix     global_tran_matrix,/* global
                      transformation matrix*/
 Gtran_mode global_tran_mode, /* global
                      transformation mode*/
 const Gtran_matrix     local_tran_matrix,/* local
                      transformation matrix*/
 Gtran_mode local_tran_mode,  /* local
                      transformation mode*/
 const Gnameset   *set_names, /* set of names*/
 Gscissor_sel     scissor_sel /* scissor select*/);
extern void gcopy_ndc_pic_pic_part_store(
 const Gsel_crit  *sel_crit,  /* sel. criterion*/
 Gint       pic_part_name,    /* picture part name*/
 const Gnameset   *set_names  /* set of names*/);
/* Input functions*/
extern void gset_in_dev_mode(
 const Gdev_id    *dev_id,    /* input device number*/
 Gop_mode   op_mode     /* operating mode*/);
extern void gset_loc_mode(
 Gint       ws_id,      /* workstation identifier*/
 Gint       loc_num,    /* locator device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void gset_stroke_mode(
 Gint       ws_id,       /* workstation identifier*/
 Gint       stroke_num, /* stroke device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void gset_val_mode(
 Gint       ws_id,       /* workstation identifier*/
 Gint       val_num,     /* valuator device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void gset_choice_mode(
 Gint       ws_id,       /* workstation identifier*/
 Gint       choice_num, /* choice device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void gset_pick_mode(
 Gint       ws_id,       /* workstation identifier*/
 Gint       pick_num,    /* pick device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void gset_string_mode(
 Gint       ws_id,       /* workstation identifier*/
 Gint       string_num, /* string device number*/
 Gop_mode   op_mode,     /* operating mode*/
 Gecho_switch     echo_switch /* echo switch*/);
extern void greq_in(
 const Gdev_id    *dev_id,    /* input device number*/
 Gin_status *in_status, /* OUT input status*/
 Gin_value  *log_in_value      /* OUT logical input value*/);
extern void greq_loc(
 Gint       ws_id,       /* workstation identifier*/
 Gint       loc_num,     /* locator device number*/
 Gin_status *in_status, /* OUT input status*/
 Gint       *norm_tran_num,    /* OUT normalization
                          transformation number*/
 Gpoint     *loc_pos     /* OUT locator position*/);
extern void greq_stroke(
 Gint       ws_id,       /* workstation identifier*/
 Gint       stroke_num, /* stroke device number*/
 Gin_status *in_status, /* OUT input status*/
 Gint       *norm_tran_num,    /* OUT normalization
                          transformation number*/
 Gpoint_list      *stroke      /* OUT stroke*/);
extern void greq_val(
 Gint       ws_id,       /* workstation identifier*/
 Gint       val_num,     /* valuator device number*/
 Gin_status *in_status, /* OUT input status*/
 Gfloat     *value       /* OUT value*/);
extern void greq_choice(
 Gint       ws_id,       /* workstation identifier*/
 Gint       choice_num, /* choice device number*/
 Gin_status *in_status, /* OUT input status*/
 Gint       *choice     /* OUT requested choice*/);
extern void greq_pick(
 Gint       ws_id,      /* workstation identifier*/
 Gint       pick_num,   /* pick device number*/
 Gin_status *in_status, /* OUT input status*/
 Gpick      *pick /* OUT requested pick value*/);
extern void greq_string(
 Gint       ws_id,      /* workstation identifier*/
 Gint       string_num, /* string device number*/
 Gin_status *in_status, /* OUT input status*/
 char       *string     /* OUT requested string*/);
extern void gsample_in(
 const Gdev_id    *dev_id,    /* input device number*/
 Gin_value  *log_in_value     /* OUT logical input value*/);
extern void gsample_loc(
 Gint       ws_id,      /* workstation identifier*/
 Gint       loc_num,    /* locator device number*/
 Gint       *norm_tran_num,   /* OUT normalization
                      transformation number*/
 Gpoint     *loc_pos    /* OUT locator position*/);
extern void gsample_stroke(
 Gint       ws_id,      /* workstation identifier*/
 Gint       stroke_num, /* stroke device number*/
 Gint       *norm_tran_num,   /* OUT normalization
                      transformation number*/
 Gpoint_list      *stroke     /* OUT stroke*/);
extern void gsample_val(
 Gint       ws_id,      /* workstation identifier*/
 Gint       val_num,    /* valuator device number*/
 Gfloat     *value      /* OUT value*/);
extern void gsample_choice(
 Gint       ws_id,      /* workstation identifier*/
 Gint       choice_num, /* choice device number*/
 Gin_status *in_status, /* OUT input status*/
 Gint       *choice     /* OUT choice*/);
extern void gsample_pick(
 Gint       ws_id,      /* workstation identifier*/
 Gint       pick_num,   /* pick device number*/
 Gin_status *in_status, /* OUT input status*/
 Gpick      *pick /* OUT pick value*/);
extern void gsample_string(
 Gint       ws_id,      /* workstation identifier*/
 Gint       string_num, /* string device number*/
 char       *string     /* OUT string*/);
extern void gawait_in(
 Gfloat     timeout,    /* timeout (seconds)*/
 Gstore     store,      /* handle to Store object*/
 Gdev_id_in_value_list  **dev_id_in_values/* OUT logical
                     input devices and
                     logical input value*/);
```

```
extern void gawait_event(
 Gfloat     timeout,     /* timeout (seconds)*/
 Gint       *ws_id,      /* OUT workstation identifier*/
 Gin_class  *class,      /* OUT device class*/
 Gint       *in_num      /* OUT logical input device number*/);
extern void gget_loc(
 Gint       *norm_tran_num,    /* OUT normalization
                     transformation number*/
 Gpoint     *loc_pos     /* OUT locator position*/);
extern void gget_stroke(
 Gint       *norm_tran_num,    /* OUT normalization
                     transformation number*/
 Gpoint_list      *stroke      /* OUT stroke*/);
extern void gget_val(
 Gfloat     *value       /* OUT value*/);
extern void gget_choice(
 Gin_status *in_status,  /* OUT input status*/
 Gint       *choice      /* OUT choice*/);
extern void gget_pick(
 Gin_status *in_status,  /* OUT input status*/
 Gpick      *pick /* OUT pick value*/);
extern void gget_string(
 char       *string      /* OUT string*/);
extern void gflush_dev_events(
 Gdev_id    *dev_id      /* device identifier*/);
extern void gset_font_ind_map(
 const Gfont_ind_map    *font_ind_map/* font index mapping*/);
extern void gget_glyph_name(
 Gint       font_ind,    /* font index */
 char       char_code,   /* character code*/
 Gint       *glyph_name  /* OUT glyph name*/);
extern void gset_char_code(
 Gint       font_ind,    /* font index */
 char       char_code,   /* character code*/
 Gint       glyph_name   /* glyph name*/);
/* Audit and playback functions */
extern void gaudit(
 Gint       audit_id,    /* audit identifier*/
 const Gaudit_op  *audit_op    /* audit operation*/);
extern void gwrite_user_rec_audit(
 Gint       audit_id,    /* audit identifier*/
 const Gaudit_user_data *user_data/* user data record*/);
extern void gplayback(
 Gint       audit_id,    /* audit identifier*/
 const Gplayback_op     *playback_op/* playback operation*/);
extern void gread_item_func_name_audit(
 Gint       audit_id,    /* audit identifier*/
 Gint       *func_name   /* OUT function name*/);
extern void gread_item_audit(
 Gint       audit_id,    /* audit identifier*/
 Gstore     store,       /* handle to Store object*/
 Gfunc_params     **func_params/* OUT function parameters*/);
```

```
extern void gproc_audit_item(
 Gint       audit_id,    /* audit identifier*/
 Gproc_op   proc_op      /* process operation*/);
extern void ginq_op_st_entry(
 Gop_st_name      op_st_name, /* entry name*/
 Gint       *err_ind,    /* OUT error indicator*/
 Gop_st_entry     *op_st_entry/* OUT operating state */);
extern void ginq_gks_dt_entry(
 Ggks_dt_name     gks_dt_name,/* entry name*/
 Gstore     store,       /* handle to Store object*/
 Gint       *err_ind,    /* OUT error indicator*/
 Ggks_dt_entry    **gks_dt     /* OUT GKS
                   description  table entry*/);
extern void ginq_list_avail_ws_types(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,   /* starting index*/
 Gint       *err_ind,    /* OUT error indicator*/
 Gint_list  *ws_gen_types,     /* OUT list of avalailable generic
                   ws types */
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_list_avail_fonts(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,   /* starting index*/
 Gint       *err_ind,    /* OUT error indicator*/
 Gint_list  *fonts,      /* OUT list of avalailable fonts */
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_def_font_ind_map(
 Gint       *err_ind,    /* OUT error indicator*/
 Gfont_ind_map    *font_ind_map/* OUT font index mapping*/);
extern void ginq_gks_sl_entry(
 Ggks_sl_name     gks_sl_name,/* entry name*/
 Gstore     store,       /* handle to Store object*/
 Gint       *err_ind,    /* OUT error indicator*/
 Ggks_sl_entry    **gks_sl_entry/* OUT GKS state
                   list     */);
extern void ginq_route_dir(
 Gint       *err_ind,    /* OUT error indicator*/
 Groute_dir *route_dir   /* OUT current route diretion*/);
extern void ginq_scissor_mode(
 Gint       *err_ind,    /* OUT error indicator*/
 Gscissor_mode    *scissor_mode/* OUT current scissor mode*/);
extern void ginq_set_open_wss(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,   /* starting index*/
 Gint       *err_ind,    /* OUT error indicator*/
 Gint_list  *open_ws,    /* OUT list of open ws ids*/
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_set_active_wss(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,   /* starting index*/
 Gint       *err_ind,    /* OUT error indicator*/
 Gint_list  *active_ws,  /* OUT list of active ws ids*/
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_set_open_audits(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,  /* starting index*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gopen_audit_list *open_audit_set,/* OUT set of open audits*/
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_set_open_playbacks(
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,  /* starting index*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint_list  *open_playbacks,  /* OUT set of open playbacks*/
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_in_queue(
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gin_queue  **in_queue  /* OUT input queue */);
extern void ginq_font_ind_map(
 Gint       *err_ind,   /* OUT error indicator*/
 Gfont_ind_map    *font_ind_map/* OUT font index mapping*/);
extern void ginq_cur_pick_id(
 Gint       *err_ind,   /* OUT error indicator*/
 Gint       *pick_id    /* OUT current pick identifier*/);
extern void ginq_nameset(
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gnameset   **nameset   /* nameset */);
extern void ginq_scissor_set(
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gscissor_list    **scissor_set/* scissor */);
extern void ginq_global_tran_matrix(
 Gtran_matrix     global_tran_matrix/* global transformation
                     matrix   */);
extern void ginq_local_tran_matrix(
 Gtran_matrix     local_tran_matrix/* local transformation
                     matrix   */);
extern void ginq_pat_size(
 Gint       *err_ind,   /* OUT error indicator*/
 Gfloat_size      *pat_size    /* OUT current pattern size*/);
extern void ginq_pat_ref_point(
 Gint       *err_ind,   /* OUT error indicator*/
 Gpoint     *pat_ref_point    /* OUT current pattern reference point*/);
extern void ginq_char_ht(
 Gint       *err_ind,   /* OUT error indicator*/
 Gfloat     *char_ht    /* OUT current character height*/);
extern void ginq_char_up_vec(
 Gint       *err_ind,   /* OUT error indicator*/
 Gvec       *char_up_vec      /* OUT current character up vector*/);
extern void ginq_text_skew_angle(
 Gint       *err_ind,   /* OUT error indicator*/
 Gfloat     *text_skew_angle  /* OUT current text skew angle*/);
```

```
extern void ginq_text_path(
 Gint        *err_ind,    /* OUT error indicator*/
 Gtext_path *text_path   /* OUT current text path*/);
extern void ginq_text_align(
 Gint        *err_ind,    /* OUT error indicator*/
 Gtext_align      *text_align /* OUT current text alignment*/);
extern void ginq_line_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *line_ind    /* OUT current line index*/);
extern void ginq_linetype(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *linetype    /* OUT current linetype*/);
extern void ginq_linewidth(
 Gint        *err_ind,    /* OUT error indicator*/
 Gfloat      *linewidth   /* OUT current linewidth scale factor*/);
extern void ginq_line_colr_specif(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_specif     *line_colr_specif/* OUT current line colour specifier*/);
extern void ginq_line_colr_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *line_colr_ind    /* OUT line colour index*/);
extern void ginq_marker_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *marker_ind /* OUT current marker index*/);
extern void ginq_marker_type(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *marker_type      /* OUT current marker type*/);
extern void ginq_marker_size(
 Gint        *err_ind,    /* OUT error indicator*/
 Gfloat      *marker_size      /* OUT current marker size scale factor*/);
extern void ginq_marker_colr_specif(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_specif     *marker_colr_specif/* OUT current marker
                       colour specifier*/);
extern void ginq_marker_colr_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *marker_colr_ind  /* OUT marker colour index*/);
extern void ginq_area_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *area_ind    /* OUT current area index*/);
extern void ginq_int_style(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_style *int_style   /* OUT current interior style*/);
extern void ginq_int_style_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *int_style_ind    /* OUT current interior style index*/);
extern void ginq_int_colr_specif(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_specif     *int_colr_specif/* OUT current interior
                       colour specifier*/);
extern void ginq_int_colr_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *int_colr_ind     /* OUT interior colour index*/);
extern void ginq_edge_flag(
 Gint        *err_ind,    /* OUT error indicator*/
 Gedge_flag *edge_flag   /* OUT current edge flag*/);
extern void ginq_edgetype(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *edgetype    /* OUT current edgetype*/);
extern void ginq_edgewidth(
 Gint        *err_ind,    /* OUT error indicator*/
 Gfloat      *edgewidth   /* OUT current edgewidth scale factor*/);
extern void ginq_edge_colr_specif(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_specif     *edge_colr_specif/* OUT current edge colour specifier*/);
extern void ginq_edge_colr_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *edge_colr_ind     /* OUT edge colour index*/);
extern void ginq_text_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *text_ind    /* OUT current text index*/);
extern void ginq_text_font_prec(
 Gint        *err_ind,    /* OUT error indicator*/
 Gtext_font_prec   *font_prec  /* OUT current text font and precision */);
extern void ginq_char_expan(
 Gint        *err_ind,    /* OUT error indicator*/
 Gfloat      *char_expan /* OUT current character expansion factor*/);
extern void ginq_char_space(
 Gint        *err_ind,    /* OUT error indicator*/
 Gfloat      *char_space /* OUT current character spacing*/);
extern void ginq_text_colr_specif(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_specif     *text_colr_specif/* OUT current text colour specifier*/);
extern void ginq_text_colr_ind(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *text_colr_ind     /* OUT text colour index*/);
extern void ginq_asfs(
 Gint        *err_ind,    /* OUT error indicator*/
 Gasfs       *asfs /* OUT current attribute source flags */);
extern void ginq_cur_norm_tran_num(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *norm_tran_num     /* OUT current normalization
                      transformation number*/);
extern void ginq_list_norm_trans(
 Gint        *err_ind,    /* OUT error indicator*/
 Gtran       norm_tran[64]     /* OUT list of normalization
                      transformations */);
extern void ginq_norm_tran(
 Gint        num,  /* normalization transformation number*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gtran       *norm_tran   /* OUT normalization transformation*/);
extern void ginq_list_vp_in_pris(
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        vp_in_pri[64]     /* OUT list of viewport input
```

```
                  priorities*/);
extern void ginq_name_open_pic_part(
 Gint        *err_ind,    /* OUT  error indicator*/
 Gint        *name_open_pic_part/* OUT name of open picture
                  part     */);
extern void ginq_set_pic_part_names(
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *pic_part_names,  /* OUT list of picture part names*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_name_open_seg(
 Gint        *err_ind,    /* OUT  error indicator*/
 Gint        *name_open_seg    /* OUT name of open segment*/);
extern void ginq_set_seg_names(
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *seg_names, /* OUT list of segment names*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_seg_attrs(
 Gint        seg_name,    /* segment name*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gseg_attrs  *seg_attrs   /* OUT segment attributes */);
extern void ginq_set_assoc_wss(
 Gint        seg_name,    /* segment name*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *assoc_wss, /* OUT list of
                  asociated workstations*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_name_open_stencil(
 Gint        *err_ind,    /* OUT  error indicator*/
 Gint        *name_open_stencil/* OUT name of open stencil*/);
extern void ginq_set_stencil_names(
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *stencil_names,   /* OUT list of stencil names*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_stencil_attrs(
 Gint        stencil_name,     /* stencil name*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gstencil_attrs    *stencil_attrs/* OUT stencil attributes */);
extern void ginq_cur_cont_attrs(
 Gint        *err_ind,    /* OUT error indicator*/
 Gcont_attrs       *cont_attrs /* OUT contour attributes */);
extern void ginq_name_open_tiling(
 Gint        *err_ind,    /* OUT  error indicator*/
 Gint        *name_open_tiling /* OUT name of open tiling*/);
extern void ginq_set_tiling_names(
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *tiling_names,    /* OUT list of tiling names*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_in_overf(
 Gint        *err_ind,   /* OUT error indicator*/
 Gint        *ws_id,     /* OUT workstation identifier*/
 Gin_class   *meas_class,      /* OUT measure class*/
 Gint        *in_num     /* OUT logical input device number*/);
/* Utility functions*/
/* GKS Utilities*/
extern void geval_tran_matrix(
 const Gpoint     *point,     /* fixed point*/
 const Gvec *shift,     /* shift vector*/
 Gfloat     angle,      /* rotation angle*/
 const Gvec *scale,     /* scale factors*/
 Gcoord_switch    coord_switch,/* coordinate switch*/
 Gtran_matrix     tran_matrix /* OUT transformation matrix*/);
extern void gaccum_tran_matrix(
 const Gtran_matrix     matrix,/* transformation matrix*/
 const Gpoint     *point,     /* fixed point*/
 const Gvec *shift,     /* shift vector*/
 Gfloat     angle,      /* rotation angle*/
 const Gvec *scale,     /* scale factors*/
 Gcoord_switch    coord_switch,/* coordinate switch*/
 Gtran_matrix     tran_matrix /* OUT transformation matrix*/);
extern void geval_circle(
 const Gpoint     *circ_ctr,  /* circle centre*/
 Gfloat     radius,     /* radius*/
 Gconic_matrix    circle      /* OUT circle*/);
extern void geval_ell(
 const Gpoint     *ell_ctr,   /* ellipse centre*/
 const Gpoint     conj_radius_end_point[2],/* conjugate
                     radius end points*/
 Gconic_matrix    ell    /* OUT ellipse*/);
extern void geval_circ_arc_3_point(
 const Gpoint     circ_point[3],/* 3 circle points
                     (start, intermediate, end)*/
 Gconic_sec *circ_arc    /* OUT circular arc*/);
extern void geval_circ_arc_ctr(
 const Gpoint     *circ_ctr,  /* circle centre*/
 const Gvec vec[2],     /* start and end vector*/
 Gsense_flag      sense_flag, /* sense flag*/
 Gfloat     radius,     /* radius*/
 Gconic_sec *circ_arc    /* OUT circular arc*/);
extern void geval_ell_arc(
 const Gpoint     *ell_ctr,   /* ellipse centre*/
 const Gpoint     conj_radius_end_point[2],/* conjugate
                     radius end points*/
 const Gvec vec[2],     /* start and end vector*/
 Gconic_sec *ell_arc     /* OUT elliptic arc*/);
```

```
extern void geval_hyp_arc(
 const Gpoint     *tran_radius_end_point,/* transverse
                     radius end point*/
 const Gpoint     *hyp_ctr,   /* hyperbola centre*/
 const Gpoint     *conj_radius_end_point,/* conjugate
                     radius end point*/
 const Gvec vec[2],     /* start and end vector*/
 Gconic_sec *hyp_arc    /* OUT hyperbolic arc*/);
extern void geval_par_arc(
 const Gpoint     *start_point,/* start point*/
 const Gpoint     *intersec_point,/* tangential
                     intersection point*/
 const Gpoint     *end_point, /* end point*/
 Gconic_sec *par_arc    /* OUT parabolic arc*/);
extern void geval_wc_ord(
 Gfloat     wc_ord,     /* WC ordinate*/
 Gord_sel   ord_sel,    /* ordinate selector */
 Gfloat     *ndc_value  /* OUT NDC value */);
extern void gset_err_hand(
 const void (*new_hand)(Gint , Gint , const char *),
            /* the application's error handling address */
 void       (**old_hand)(Gint , Gint , const char *)
            /* OUT address of the error handling replaced */);
extern void gcreate_store(
 Gint       *err_ind,   /* OUT error indicator*/
 Gstore     *store      /* OUT handle to Store object*/);
extern void gdel_store(
 Gint       *err_ind,   /* OUT error indicator*/
 Gstore     *store      /* IN/OUT storage to be deleted*/);
/* Workstation functions*/
/* Control functions*/
extern void gopen_ws(
 Gint       ws_id,      /* workstation identifier*/
 const void *ws_specif_info,  /* workstation specifier
                     and specific information*/
 Gint       ws_gen_type /* generic workstation type*/);
extern void gclose_ws(
 Gint       ws_id /* workstation identifier*/);
extern void gsave_ws_sl(
 Gint       ws_id,      /* workstation identifier*/
 const Gws_sl_name_list *entry_names,/* list of entry
                     names    */
 Gstore     store,      /* handle to Store object*/
 Gws_sl_entry_list      **entry_list/* OUT list of
                     entry values*/);
extern void grestore_ws_sl(
 Gint       ws_id,      /* workstation identifier*/
 const Gws_sl_entry_list      *entries/*  list of entries*/);
extern void gget_ws_status(
 Gint       ws_id,      /* workstation identifier*/
 Gws_status *ws_status  /*  OUT workstation status*/);
extern void gremove_backdrop(
 Gint       ws_id /* workstation identifier*/);
extern void gset_rep(
 Gint       ws_id,      /* workstation identifier*/
 const Grep *rep_value  /* representation value */);
extern void gset_line_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       line_ind,   /* line index*/
 const Gline_bundle     *line_bundle/* line representation */);
extern void gset_marker_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       marker_ind, /* marker index*/
 const Gmarker_bundle   *marker_bundle/* marker representation */);
extern void gset_text_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       text_ind,   /* text index*/
 const Gtext_bundle     *text_bundle/* text representation */);
extern void gset_area_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       area_ind,   /* area index*/
 const Garea_bundle     *area_bundle/* area representation */);
extern void gset_pat_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       pat_ind,    /* pattern index*/
 const Gpat_rep   *pat_rep    /* pattern representation */);
extern void gset_colr_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       colr_ind,   /* colour index*/
 const Gcolr_rep  *colr_rep   /* colour representation */);
extern void gset_view_pri(
 Gint       ws_id,      /* workstation identifier*/
 Gint       view_ind,   /* view index*/
 Gint       ref_view_ind,     /* reference view index*/
 Grel_pri   rel_pri     /* relative priority*/);
extern void gset_view_sel_crit(
 Gint       ws_id,      /* workstation identifier*/
 Gint       view_ind,   /* view index*/
 const Gsel_crit  *sel_crit   /* selection criterion*/);
extern void gset_view(
 Gint       ws_id,      /* workstation identifier*/
 Gint       view_ind,   /* view index*/
 const Gtran_matrix     ori_matrix,/* view orientation matrix*/
 const Gtran_matrix     map_matrix,/* view mapping matrix*/
 const Gscissor   *scissor    /* view scissor*/);
extern void gset_ws_vis_effects(
 Gint       ws_id,      /* workstation identifier*/
 Gvis_effects_st  vis_effects_st/* visual effects state*/);
extern void gset_ws_win_vp(
 Gint       ws_id,      /* workstation identifier*/
 const Gtran      *ws_win_vp  /* workstation window
                     and viewport limits*/);
extern void gset_ws_win(
 Gint       ws_id,      /* workstation identifier*/
 const Glimit     *ws_win     /* workstation window */);
extern void gset_ws_vp(
 Gint        ws_id,      /* workstation identifier*/
 const Glimit     *ws_vp      /* workstation viewport limits*/);
extern void gdef_in_dev(
 const Gdev_id    *dev_id,    /* device identifier*/
 const Gint_list  *meas_seq,  /* measure sequence*/
 const Gint_list  *trigger_set,/* trigger set*/
 const Ginit_value      *init_value/* initial value*/);
extern void ginit_in_dev(
 const Gdev_id    *dev_id,    /* device identifier*/
 const Ginit_value      *init_value/* initial value*/);
extern void ginit_loc(
 Gint        ws_id,      /* workstation identifier*/
 Gint        loc_num,    /* locator device number*/
 Gint        init_norm_tran_num,/* initial normalization
                    transformation number*/
 const Gpoint     *init_loc_pos,/* initial locator position*/
 Gint        pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gloc_data  *loc_data   /* locator data record*/);
extern void ginit_stroke(
 Gint        ws_id,      /* workstation identifier*/
 Gint        stroke_num, /* stroke device number*/
 Gint        init_norm_tran_num,/* initial normalization
                    transformation number*/
 const Gpoint_list      *init_stroke,/* initial stroke*/
 Gint        pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gstroke_data     *stroke_data/* stroke data record*/);
extern void ginit_val(
 Gint        ws_id,      /* workstation identifier*/
 Gint        val_num,    /* valuator device number*/
 Gfloat      init_value, /* initial value*/
 Gint        pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gval_data  *val_data   /* valuator data record*/);
extern void ginit_choice(
 Gint        ws_id,      /* workstation identifier*/
 Gint        choice_num, /* choice device number*/
 Gin_status init_status,      /* initial status*/
 Gint        init_choice,      /* initial choice*/
 Gint        pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gchoice_data     *choice_data/* choice data record*/);
extern void ginit_pick(
 Gint        ws_id,      /* workstation identifier*/
 Gint        pick_num,   /* pick device number*/
 Gin_status init_status,      /* initial status*/
 const Gpick      *init_pick, /* initial pick value */
 Gint        pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gpick_data *pick_data  /* pick data record*/);
extern void ginit_string(
 Gint       ws_id,      /* workstation identifier*/
 Gint       string_num, /* string device number*/
 const char *init_string,     /* initial string*/
 Gint       pet,  /* prompt and echo type*/
 const Glimit     *echo_area, /* echo area*/
 const Gstring_data     *string_data/* string data record*/);
extern void gset_ws_sel_crit(
 Gint       ws_id,      /* workstation identifier*/
 const Gsel *sel  /* selection type and criterion*/);
extern void gcopy_real_pic_real_mf(
 Gint       ws_id,      /* workstation identifier*/
 const void *mf_specif, /* metafile specifier*/
 Gint       pic_id      /* picture identifier*/);
extern void gcopy_blank_real_pic_real_mf(
 Gint       ws_id,      /* workstation identifier*/
 const void *mf_specif, /* metafile specifier*/
 Gint       pic_id      /* picture identifier*/);
extern void gcopy_real_mf_pic_backdrop(
 Gint       ws_id,      /* workstation identifier*/
 const void *mf_specif, /* metafile specifier*/
 Gint       pic_id      /* picture identifier*/);
extern void gmessage(
 Gint       ws_id,      /* workstation identifier*/
 const char *message    /* message string*/);
/* Inquiry functions*/
extern void ginq_ws_sl_entry(
 Gint       ws_id,      /* workstation identifier*/
 Gws_sl_name      name, /* entry name*/
 Ginq_type  type, /* type of returned values*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gws_sl_entry     **ws_sl_entry/* OUT workstation
                  state list entry*/);
extern void ginq_ws_conn_type(
 Gint       ws_id,      /* workstation identifier*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 void       **conn_id,  /* OUT connection identifier*/
 Gint       *ws_type    /* OUT workstation type*/);
extern void ginq_ws_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gws_st     *ws_st      /* OUT workstation state*/);
extern void ginq_list_line_inds(
 Gint       ws_id,      /* workstation identifier*/
 Gint       num_elems_appl_list,/* length of application list*/
 Gint       start_ind,  /* starting index*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint_list  *def_line_inds,   /* OUT list of defined
                  line indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_line_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        line_ind,   /* line index*/
 Ginq_type   type, /* type of returned values*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gline_bundle     *line_rep   /* OUT line representation*/);
extern void ginq_list_marker_inds(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *def_marker_inds, /* OUT list of defined
                      marker indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_marker_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        marker_ind, /* marker index*/
 Ginq_type   type, /* type of returned values*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gmarker_bundle   *marker_rep /* OUT marker representation*/);
extern void ginq_list_text_inds(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *def_text_inds,   /* OUT list of defined
                      text indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_text_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        text_ind,   /* text index*/
 Ginq_type   type, /* type of returned values*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gtext_bundle     *text_rep   /* OUT text representation*/);
extern void ginq_list_area_inds(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *def_area_inds,   /* OUT list of defined
                      area indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_area_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        area_ind,   /* area index*/
 Ginq_type   type, /* type of returned values*/
 Gint        *err_ind,   /* OUT error indicator*/
 Garea_bundle     *area_rep   /* OUT area representation*/);
extern void ginq_list_pat_inds(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *def_pat_inds,    /* OUT list of defined
                      pattern indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_pat_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        pat_ind,    /* pattern index*/
 Ginq_type   type, /* type of returned values*/
 Gstore      store,      /* handle to Store object*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gpat_rep    **pat_rep   /* OUT pattern representation*/);
extern void ginq_list_colr_inds(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *def_colr_inds,   /* OUT list of defined
                      colour indices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_colr_rep(
 Gint        ws_id,      /* workstation identifier*/
 Gint        colr_ind,   /* colour index*/
 Ginq_type   type, /* type of returned values*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gcolr_rep   *colr_rep   /* OUT colour representation*/);
extern void ginq_ws_win_vp(
 Gint        ws_id,      /* workstation identifier*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gtran       *ws_win_vp  /* OUT workstation window
                       and viewport*/);
extern void ginq_vis_effects_st(
 Gint        ws_id,      /* workstation identifier*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gvis_effects_st *vis_effects_st/* OUT visual effects state*/);
extern void ginq_sel_table(
 Gint        ws_id,      /* workstation identifier*/
 Gstore      store,      /* handle to Store object*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gsel_table **sel_table /* OUT table of selection criteria*/);
extern void ginq_set_seg_names_ws(
 Gint        ws_id,      /* workstation identifier*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gint_list   *seg_names, /* OUT list of segment names*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_in_dev_op_modes_init_values(
 Gint        ws_id,      /* workstation identifier*/
 Gstore      store,      /* handle to Store object*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gop_modes_init_values  **op_modes_init_values/* OUT table of
```

```
                    input device
                    operating modes and
                    initial values*/);
extern void ginq_view_rep(
 Gint       ws_id,      /* workstation identifier*/
 Gint       view_ind,   /* view index*/
 Ginq_type  type, /* type of returned values*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gview_rep  **view_rep  /* OUT view representation*/);
extern void ginq_view_pris(
 Gint       ws_id,      /* workstation identifier*/
 Ginq_type  type, /* type of returned values*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint       view_pri[16]      /* OUT view priorities*/);
extern void ginq_loc_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       loc_num,    /* locator device number*/
 Ginq_type  type, /* type of returned values*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 Gint       *init_norm_tran_num,/* OUT initial normalization
                    transformation number*/
 Gpoint     *init_loc_pos,    /* OUT initial locator position*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gloc_data  **loc_data  /* OUT locator data record*/);
extern void ginq_stroke_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       stroke_num, /* stroke device number*/
 Ginq_type  type, /* type of returned values*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 Gint       *init_norm_tran_num,/* OUT initial normalization
                    transformation number*/
 Gpoint_list      **init_stroke,/* OUT initial stroke*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gstroke_data     **stroke_data/* OUT stroke data record*/);
extern void ginq_val_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       val_num,    /* valuator device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 Gfloat     *init_value,      /* OUT initial value*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gval_data  **val_data  /* OUT valuator data record*/);
extern void ginq_choice_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       choice_num, /* choice device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 Gin_status *init_status,     /* OUT initial choice status*/
 Gint       *init_choice,     /* OUT initial choice*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gchoice_data     **choice_data/* OUT choice data record*/);
extern void ginq_pick_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       pick_num,   /* pick device number*/
 Ginq_type  type, /* type of returned values*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 Gin_status *init_status,     /* OUT initial pick status*/
 Gpick      *init_pick, /* OUT initial pick value*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gpick_data **pick_data /* OUT pick data record*/);
extern void ginq_string_st(
 Gint       ws_id,      /* workstation identifier*/
 Gint       string_num, /* string device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gop_mode   *mode,      /* OUT operating mode*/
 Gecho_switch     *esw, /* OUT echo switch*/
 char       **init_string,    /* OUT intial string*/
 Gint       *pet, /* OUT prompt/echo type*/
 Glimit     *echo_area, /* OUT echo area*/
 Gstring_data     **string_data/* OUT string data record*/);
extern void ginq_gen_ws_dt_entry(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gws_dt_name      ws_dt_name, /* workstation description table
                         entry name*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gws_dt_entry     **ws_dt_entry/* OUT workstation description
                         table entry*/);
extern void ginq_ws_class(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gws_class  *class      /* OUT workstation class*/);
extern void ginq_disp_space_size(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gdisp_space_size *disp_size  /* OUT display [space] size*/);
extern void ginq_line_facs(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gline_facs *line_facs, /* OUT line facilities*/
 Gint        *num_elems_impl_list/* OUT length of linetype
                      list in impl.*/);
extern void ginq_pred_line_rep(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gline_bundle     *line_rep    /* OUT predefined line rep.*/);
extern void ginq_marker_facs(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gmarker_facs     *marker_facs,/* OUT marker facilities*/
 Gint        *num_elems_impl_list/* OUT length of marker type
                      list in impl.*/);
extern void ginq_pred_marker_rep(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gmarker_bundle   *marker_rep /* OUT predefined marker rep.*/);
extern void ginq_text_facs(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,  /* starting index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gtext_facs *text_facs, /* OUT text facilities*/
 Gint        *num_elems_impl_list/* OUT length of text font
                      and precision list in impl.*/);
extern void ginq_pred_text_rep(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Gtext_bundle     *text_rep    /* OUT predefined text rep.*/);
extern void ginq_area_facs(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gstore      store,       /* handle to Store object*/
 Gint        *err_ind,   /* OUT error indicator*/
 Garea_facs **area_facs /* OUT interior facilities*/);
extern void ginq_pred_area_rep(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gint        *err_ind,   /* OUT error indicator*/
 Garea_bundle     *area_rep    /* OUT predefined area rep.*/);
extern void ginq_pat_facs(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *num_pred_inds     /* OUT num. of predef. pattern indices*/);
extern void ginq_pred_pat_rep(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gstore      store,       /* handle to Store object*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gpat_rep    **pat_rep    /* OUT predefined pattern rep.*/);
extern void ginq_colr_facs(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_facs *colr_facs   /* OUT colour facilities*/);
extern void ginq_pred_colr_rep(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        ind,  /* predefined index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gcolr_rep   *colr_rep    /* OUT predefined colour rep.*/);
extern void ginq_list_avail_gdps(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        start_ind,   /* starting index*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *gdp, /* OUT list of GDPs*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_set_avail_in_devs(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gdev_id_list      *dev_ids,    /* OUT set of input devices*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_num_avail_in(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gnum_in     *num_in      /* OUT number of input devices*/);
extern void ginq_in_dev_init_values(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gstore      store,       /* handle to Store object*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gin_dev_init_value_list       **in_dev_init_values/* OUT input device
                      initial values */);
extern void ginq_set_avail_meass(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *meas_ids,   /* OUT set of measure ids*/
 Gint        *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_set_avail_triggers(
 Gint        ws_gen_type,       /* workstation generic type*/
 Gint        num_elems_appl_list,/* length of application list*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint_list   *trigger_ids,      /* OUT set of trigger ids*/
 Gint       *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_in_dev_comps(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gdev_comp_list   **dev_comps /* OUT input device
                   compositions */);
extern void ginq_def_loc_data(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       loc_num,    /* logical input device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gpoint     *loc_pos,   /* OUT default locator position*/
 Gint_list  **pet_list, /* OUT list of prompt and echo types*/
 Glimit     *echo_area, /* OUT default echo area*/
 Gloc_data  **loc_data  /* OUT default data record*/);
extern void ginq_def_stroke_data(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       stroke_num, /* logical input device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint       *max_buf_size,    /* OUT max. input buffer size
                        (nr. of points)*/
 Gint_list  **pet_list, /* OUT list of prompt and echo types*/
 Glimit     *echo_area, /* OUT default echo area*/
 Gstroke_data     **stroke_data/* OUT default data record*/);
extern void ginq_def_val_data(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       val_num,    /* logical input device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gfloat     *def_val,   /* OUT default initial value*/
 Gint_list  **pet_list, /* OUT list of prompt and echo types*/
 Glimit     *echo_area, /* OUT default echo area*/
 Gval_data  **val_data  /* OUT default data record*/);
extern void ginq_def_choice_data(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       choice_num, /* logical input device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint       *max_num_choices, /* OUT max. num. of choices*/
 Gint_list  **pet_list, /* OUT list of prompt and echo types*/
 Glimit     *echo_area, /* OUT default echo area*/
 Gchoice_data     **choice_data/* OUT default data record*/);
extern void ginq_def_pick_data(
 Gint       ws_gen_type,      /* workstation generic type*/
 Gint       pick_num,   /* logical input device number*/
 Gstore     store,      /* handle to Store object*/
 Gint       *err_ind,   /* OUT error indicator*/
 Gint_list  **pet_list, /* OUT list of prompt and echo types*/
 Glimit     *echo_area, /* OUT default echo area*/
 Gpick_data **pick_data /* OUT default data record*/);
```

```
extern void ginq_def_string_data(
 Gint        ws_gen_type,      /* workstation generic type*/
 Gint        string_num, /* logical input device number*/
 Gstore      store,       /* handle to Store object*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gint        *max_buf_size,    /* OUT max. input buffer size
                          (nr. of bytes)*/
 Gint_list   **pet_list, /* OUT list of prompt and echo types*/
 Glimit      *echo_area, /* OUT default echo area*/
 Gstring_data     **string_data/* OUT default data record*/);
extern void ginq_specif_ws_dt_entry(
 Gint        ws_id,       /* workstation identifier*/
 Gws_dt_name      ws_dt_name, /* workstation description table
                     entry name*/
 Gstore      store,       /* handle to Store object*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gws_dt_entry     **ws_dt_entry/* OUT workstation description
                     table entry*/);
extern void ginq_specif_ws_dt_entry_st(
 Gint        ws_id,       /* workstation identifier*/
 Gws_dt_name      ws_dt_name, /* entry name*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gchange_flag     *entry_st    /* OUT  entry state*/);
extern void greset_specif_ws_dt_entry_st(
 Gint        ws_id,       /* workstation identifier*/
 Gws_dt_name      ws_dt_name   /* entry name*/);
/* Retrieval functions*/
extern void gget_text_extent(
 Gint        ws_id,       /* workstation identifier*/
 const Gpoint     *pos, /* text position*/
 const char *str, /* character string*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gtext_extent     *extent      /* OUT concatenation point and
                     text extent parallelogram*/);
extern void ginq_text_extent(
 Gint        ws_id,       /* workstation identifier*/
 const Gpoint     *pos, /* text position*/
 const char *str, /* character string*/
 Gint        *err_ind,    /* OUT error indicator*/
 Gtext_extent     *extent      /* OUT concatenation point and
                     text extent parallelogram*/);
extern void geval_view_ori_matrix(
 const Gpoint     *ref_point, /* fixed reference point*/
 const Gvec *view_up,    /* view up vector*/
 Gtran_matrix     view_ori_matrix/* OUT view orientation
                     matrix   */);
extern void geval_view_map_matrix(
 const Gtran      *win_vp,     /* window/viewport limits*/
 Gtran_matrix     view_map_matrix/* OUT view orientation
                     matrix   */);
/* Segment functions*/
extern void gcreate_seg(
 Gint       seg_name    /* segment name*/);
extern void gclose_seg(
 void                   );
extern void grename_seg(
 Gint       old_seg_name,     /* old segment name*/
 Gint       new_seg_name      /* new segment name*/);
extern void gdel_seg(
 Gint       seg_name    /* segment name*/);
extern void gdel_seg_ws(
 Gint       ws_id,      /* workstation identifier*/
 Gint       seg_name    /* segment name*/);
extern void gassoc_seg_ws(
 Gint       ws_id,      /* workstation identifier*/
 Gint       seg_name    /* segment name*/);
extern void gcopy_seg_ws(
 Gint       ws_id,      /* workstation identifier*/
 Gint       seg_name    /* segment name*/);
extern void ginsert_seg(
 Gint       seg_name,   /* segment name*/
 const Gtran_matrix     tran_matrix/* transformation matrix*/);
extern void gset_seg_attr(
 Gint       seg_name,   /* segment name*/
 const Gseg_attr_value  *seg_attr_value/* segment attribute value*/);
extern void gset_seg_tran(
 Gint       seg_name,   /* segment name*/
 Gtran_matrix     tran_matrix /* transformation matrix*/);
extern void gset_vis(
 Gint       seg_name,   /* segment name*/
 Gvis       vis    /* visibility */);
extern void gset_highl(
 Gint       seg_name,   /* segment name*/
 Ghighl     highl /* highlighting*/);
extern void gset_det(
 Gint       seg_name,   /* segment name*/
 Gdet       det    /* detectability*/);
extern void gset_seg_pri(
 Gint       seg_name,   /* segment name*/
 Gfloat     seg_pri     /* segment priority */);
/* Workstation activation functions*/
extern void gactivate_ws(
 Gint       ws_id /* workstation identifier*/);
extern void gdeactivate_ws(
 Gint       ws_id /* workstation identifier*/);
extern void gclear_ws(
 Gint       ws_id,      /* workstation identifier*/
 Gctrl_flag ctrl_flag   /* control flag*/);
/* Utility functions*/
extern void gget_map_seg_name(
 Gint       seg_name,   /* segment name*/
 Gint       *map_seg_name     /* OUT mapped segment name*/);
extern void gget_map_ws_id(
 Gint       ws_id,      /* workstation identifier*/
 Gint        *map_ws_id  /* OUT mapped workstation identifier*/);
```

A.4 Compatibility layer

```
/* Data types */
typedef  Garea_attrs         Gfill_attrs;
typedef  Garea_bundle        Gfill_bundle;
typedef  Garea_facs          Gfill_facs;
typedef  Gint_style          Gfill_int_style;
/* Overtaken data types */
/* Implementation independent */
/* Gattrs attributes [used] */
typedef enum {
         GATTR_LINE,
         GATTR_MARKER,
         GATTR_TEXT,
GATTR_FILL
} Gattrs;
/* Gclip clipping */
typedef struct {
         Gclip_ind           clip_ind;/* clipping indicator*/
         Glimit              clip_rect;/* clipping rectangle*/
} Gclip;
/* Gdefer_mode deferral mode */
typedef enum {
         GDEFER_ASAP,
         GDEFER_BNIG,
         GDEFER_BNIL,
         GDEFER_ASTI
} Gdefer_mode;
/* Gdisp_surf_empty display surface empty */
typedef enum {
         GSURF_NOT_EMPTY,
         GSURF_EMPTY
} Gdisp_surf_empty;
/* Gdyn_mod dynamic modification [accepted] */
typedef enum {
         GDYN_IRG,
         GDYN_IMM
} Gdyn_mod;
/* Gdyn_mod_seg_attrs dynamic modification [of] segment attributes */
typedef struct {
                                 /* changeability of:*/
         Gdyn_mod            tran;/* segment transformation */
         Gdyn_mod            invis_vis;/* appearing (invisible → visible) */
         Gdyn_mod            vis_invis;/* disappearing (visible → invisible) */
         Gdyn_mod            highl;/* highlighting  */
         Gdyn_mod            pri;/* priority  */
         Gdyn_mod            add_prims;/* addition of primitives to segment  */
         Gdyn_mod            del;/* deletion of segment  */
} Gdyn_mod_seg_attrs;
/* Gdyn_mod_ws_attrs dynamic modification [of] workstation attributes  */
typedef struct {
```

```
                                    /* changeability of:*/
        Gdyn_mod                line_bundle;/* polyline representation */
        Gdyn_mod                marker_bundle;/* polymarker representation */
        Gdyn_mod                text_bundle;/* text representation */
        Gdyn_mod                fill_bundle;/* fill area representation */
        Gdyn_mod                pat_rep;/* pattern representation */
        Gdyn_mod                colr_rep;/* colour representation */
        Gdyn_mod                ws_tran;/* workstation transformation */
} Gdyn_mod_ws_attrs;
/* Gindiv_attrs individual attributes */
typedef struct {
        Gint                    linetype;/* linetype*/
        Gfloat                  linewidth;/* linewidth scale factor*/
        Gint                    line_colr_ind;/* polyline colour index*/
        Gint                    marker_type;/* marker type*/
        Gfloat                  marker_size;/* marker size scale factor*/
        Gint                    marker_colr_ind;/* polymarker colour index*/
        Gtext_font_prec         text_font_prec;/* text font and precision*/
        Gfloat                  char_expan;/* character expansion factor*/
        Gfloat                  char_space;/* character spacing*/
        Gint                    text_colr_ind;/* text colour index*/
        Gfill_int_style         fill_int_style;/* fill area interior style*/
        Gint                    fill_style_ind;/* fill area style index*/
        Gint                    fill_colr_ind;/* fill area colour index*/
        Gasfs                   asfs;/* aspect source flags */
} Gindiv_attrs;
/* Girg_mode implicit regeneration mode */
typedef enum {
        GIRG_SUPPR,
        GIRG_ALLOWED
} Girg_mode;
/* Glevel GKS level */
typedef enum {
        GLEVEL_0A,
        GLEVEL_0B,
        GLEVEL_0C,
        GLEVEL_1A,
        GLEVEL_1B,
        GLEVEL_1C,
        GLEVEL_2A,
        GLEVEL_2B,
        GLEVEL_2C
} Glevel;
/* Gmax_ws_st_tables max. [length of] workstation state tables  */
typedef struct {
                                    /* max. num. of :*/
        Gint                    line_bundles;/* polyline bundle table entries*/
        Gint                    marker_bundles;/* polymarker bundle table entries*/
        Gint                    text_bundles;/* text bundle table entries*/
        Gint                    fill_bundles;/* fill area bundle table entries*/
        Gint                    pat_reps;/* pattern table entries*/
        Gint                    colr_reps;/* colour table entries*/
```

```
} Gmax_ws_st_tables;
/* Gmore_simult_events more simultaneous events */
typedef enum {
        GSIMULT_NO_MORE,
        GSIMULT_MORE
} Gmore_simult_events;
/* Gnew_frame_nec_upd new frame [action] necessary [at] update */
typedef enum {
        GNEW_NO,
        GNEW_YES
} Gnew_frame_nec_upd;
/* Gop_st operating state */
typedef enum {
        GST_GKCL,
        GST_GKOP,
        GST_WSOP,
        GST_WSAC,
        GST_SGOP
} Gop_st;
/* Gpres_inval presence [of] invalid [values] */
typedef enum {
        GINVAL_ABSENT,
        GINVAL_PRESENT
} Gpres_inval;
/* Gprim_attrs primitive attributes */
typedef struct {
        Gint                  line_ind;/* polyline index*/
        Gint                  marker_ind;/* polymarker index*/
        Gint                  text_ind;/* text index*/
        Gfloat                char_ht;/* character height*/
        Gvec                  char_up_vec;/* character up vector*/
        Gfloat                char_width;/* character width*/
        Gvec                  char_base_vec;/* character base vector*/
        Gtext_path            text_path;/* text path*/
        Gtext_align           text_align;/* text alignment*/
        Gint                  fill_ind;/* fill area index*/
        Gvec                  pat_width_vec;/* pattern width vector*/
        Gvec                  pat_ht_vec;/* pattern height vector*/
        Gpoint                pat_ref_point;/* pattern reference point*/
} Gprim_attrs;
/* Gupd_regen_flag update regeneration flag */
typedef enum {
        GFLAG_POSTPONE,
        GFLAG_PERFORM
} Gupd_regen_flag;
/* Gupd_st update state */
typedef enum {
        GUPD_NOT_PEND,
        GUPD_PEND
} Gupd_st;
/* Gws_cat workstation category */
typedef enum {
        GCAT_OUT,
        GCAT_IN,
        GCAT_OUTIN,
        GCAT_WISS,
        GCAT_MO,
        GCAT_MI
} Gws_cat;
/* Gws_max_nums workstation max. numbers */
typedef struct {
        Gint                simult_open;/* max. num. of simult. open wss*/
        Gint                simult_active;/* max. num. of simult. active wss*/
        Gint                assoc_seg;/* max. num. of wss associated with segment */
} Gws_max_nums;
/* Implementation dependent */
/* Gitem_data  item data record */
typedef struct {
        Gint                type;/* item type*/
        Gint                length;/* item data record length*/
        union {
                            Gaudit_user_datauser_rec;/* user record*/
                            Gfunc_paramsfunc_params;/* function parameters flag*/
                            Gdataimpl_dep;/* impl. dependent*/
        } data;
} Gitem_data;
/* Conversion macros for fields */
#define fill_int_style      int_style
#define fill_style_ind      int_style_ind
#define fill_colr_ind       int_colr_ind
#define pet_r5.attrs.fill_attrspet_r5.attrs.area_attrs
/* Obsolete macros */
/* Function macros */
#define Gfn_redraw_all_segs_ws (7)
#define Gfn_upd_ws          (8)
#define Gfn_set_defer_st    (9)
#define Gfn_set_fill_ind    (35)
#define Gfn_set_fill_int_style (36)
#define Gfn_set_fill_style_ind (37)
#define Gfn_set_fill_colr_ind  (38)
#define Gfn_set_fill_rep    (46)
#define Gfn_set_clip_ind    (53)
#define Gfn_flush_events    (94)
#define Gfn_write_item      (101)
#define Gfn_get_item_type   (102)
#define Gfn_interpret_item  (104)
/* Error macros */
#define GE_NOT_GKCL         (1)
#define GE_NOT_GKOP         (2)
#define GE_NOT_WSAC         (3)
#define GE_NOT_SGOP         (4)
#define GE_NOT_WSAC_SGOP    (5)
#define GE_NOT_WSOP_WSAC    (6)
#define GE_GKCL_GKOP        (7)
```

```
#define  GE_GKCL              (8)
#define  GE_CONN_ID_INVAL     (21)
#define  GE_NO_WS_TYPE        (23)
#define  GE_WS_CANT_OPEN      (26)
#define  GE_WISS_NOT_OPEN     (27)
#define  GE_WISS_OPEN         (28)
#define  GE_WS_INACTIVE       (30)
#define  GE_WS_MO             (31)
#define  GE_WS_NOT_MO         (32)
#define  GE_WS_MI             (33)
#define  GE_WS_NOT_MI         (34)
#define  GE_WS_IN             (35)
#define  GE_WS_WISS           (36)
#define  GE_WS_NOT_OUTIN      (37)
#define  GE_WS_NOT_IN_OUTIN   (38)
#define  GE_WS_NOT_OUT_OUTIN(39)
#define  GE_WS_NO_PIXEL       (40)
#define  GE_WS_TYPE_NO_GDP    (41)
#define  GE_WS_MAX_OPEN       (42)
#define  GE_WS_MAX_ACTIVE     (43)
#define  GE_RECT_INVAL        (51)
#define  GE_VP_INVAL          (52)
#define  GE_WIN_INVAL         (53)
#define  GE_WS_VP_INVAL       (54)
#define  GE_LINE_IND_INVAL    (60)
#define  GE_LINE_REP_UNDEF    (61)
#define  GE_LINE_REP_NOT_PRED    (62)
#define  GE_LINETYPE_ZERO     (63)
#define  GE_LINETYPE_NOT_WS   (64)
#define  GE_LINEWIDTH_LT_ZERO    (65)
#define  GE_MARKER_IND_INVAL(66)
#define  GE_MARKER_REP_UNDEF(67)
#define  GE_MARKER_REP_NOT_PRED (68)
#define  GE_MARKER_TYPE_ZERO(69)
#define  GE_MARKER_TYPE_NOT_WS   (70)
#define  GE_MARKER_SIZE_LT_ZERO (71)
#define  GE_TEXT_IND_INVAL    (72)
#define  GE_TEXT_REP_UNDEF    (73)
#define  GE_TEXT_REP_NOT_PRED    (74)
#define  GE_FONT_ZERO         (75)
#define  GE_FONT_NOT_WS       (76)
#define  GE_EXPAN_LE_ZERO     (77)
#define  GE_HT_LE_ZERO        (78)
#define  GE_UP_VEC_ZERO       (79)
#define  GE_FILL_IND_INVAL    (80)
#define  GE_FILL_REP_UNDEF    (81)
#define  GE_FILL_REP_NOT_PRED    (82)
#define  GE_INT_STYLE_NOT_WS(83)
#define  GE_STYLE_IND_ZERO    (84)
#define  GE_HATCH_STYLE_NOT_WS   (86)
#define  GE_PAT_SIZE_LE_ZERO(87)
#define  GE_PAT_REP_UNDEF     (88)
```

```
#define  GE_PAT_REP_NOT_PRED(89)
#define  GE_PAT_NOT_WS      (90)
#define  GE_DIM_INVAL       (91)
#define  GE_COLR_IND_LT_ZERO(92)
#define  GE_COLR_REP_UNDEF  (94)
#define  GE_COLR_REP_NOT_PRED   (95)
#define  GE_COLR_INVAL      (96)
#define  GE_PICK_ID_INVAL   (97)
#define  GE_COLR_MODEL_UNAVAIL  (98)
#define  GE_NUM_POINT_INVAL (100)
#define  GE_INVAL_CODE      (101)
#define  GE_CANT_GEN_GDP    (104)
#define  GE_CANT_GEN_GDP_CLIP   (105)
#define  GE_SEG_NAME_USED   (121)
#define  GE_SEG_ABSENT      (122)
#define  GE_SEG_NOT_WS      (123)
#define  GE_SEG_NOT_WISS    (124)
#define  GE_SEG_PRI_INVAL   (126)
#define  GE_IN_DEV_NOT_WS   (140)
#define  GE_IN_DEV_NOT_REQ  (141)
#define  GE_IN_DEV_NOT_SAMPLE   (142)
#define  GE_EV_SAMPLE_UNAVAIL   (143)
#define  GE_PET_NOT_WS      (144)
#define  GE_ECHO_INVAL      (145)
#define  GE_IN_DATA_INVAL   (146)
#define  GE_QUE_OVERF       (147)
#define  GE_NO_QUE_OVERF    (148)
#define  GE_ASSOC_WS_CLOSED (149)
#define  GE_NO_CUR_EV       (150)
#define  GE_INIT_INVAL      (152)
#define  GE_INIT_STROKE_INVAL   (153)
#define  GE_INIT_STRING_INVAL   (154)
#define  GE_ITEM_RESERVED   (160)
#define  GE_ITEM_LENGTH_INVAL   (161)
#define  GE_NO_ITEM_MI      (162)
#define  GE_ITEM_INVAL      (163)
#define  GE_ITEM_GKS_INVAL  (164)
#define  GE_ITEM_DATA_INVAL (165)
#define  GE_MAX_ITEM_DATA_INVAL (166)
#define  GE_USER_ITEM       (167)
#define  GE_FUNC_UNAVAIL    (168)
#define  GE_ESCAPE_FUNC_UNAVAIL (180)
#define  GE_ESCAPE_ID_INVAL (181)
#define  GE_ESCAPE_DATA_INVAL   (182)
#define  GE_MEM_OVERF       (300)
#define  GE_SEG_MEM_OVERF   (301)
#define  GE_IO_ERR_READ     (302)
#define  GE_IO_ERR_WRITE    (303)
#define  GE_IO_ERR_WRITE_WS (304)
#define  GE_IO_ERR_READ_WS  (305)
#define  GE_IO_ERR_LIB      (306)
#define  GE_IO_ERR_WS_TABLE (307)
```

```
#define  GE_ARITH_ERR        (308)
/* GKSM macros */
#define  Gksm_end_item         ( 0)
#define  Gksm_clear_ws         ( 1)
#define  Gksm_redraw_all_segs_ws ( 2)
#define  Gksm_upd_ws           ( 3)
#define  Gksm_defer_st         ( 4)
#define  Gksm_message          ( 5)
#define  Gksm_escape           ( 6)
#define  Gksm_polyline         (11)
#define  Gksm_polymarker       (12)
#define  Gksm_text             (13)
#define  Gksm_fill_area        (14)
#define  Gksm_cell_array       (15)
#define  Gksm_gdp              (16)
#define  Gksm_line_ind         (21)
#define  Gksm_linetype         (22)
#define  Gksm_linewidth        (23)
#define  Gksm_line_colr_ind    (24)
#define  Gksm_marker_ind       (25)
#define  Gksm_marker_type      (26)
#define  Gksm_marker_size      (27)
#define  Gksm_marker_colr_ind    (28)
#define  Gksm_text_ind         (29)
#define  Gksm_text_font_prec   (30)
#define  Gksm_char_expan       (31)
#define  Gksm_char_space       (32)
#define  Gksm_text_colr_ind    (33)
#define  Gksm_char_vec         (34)
#define  Gksm_text_path        (35)
#define  Gksm_text_align       (36)
#define  Gksm_fill_ind         (37)
#define  Gksm_fill_int_style   (38)
#define  Gksm_fill_style_ind   (39)
#define  Gksm_fill_colr_ind    (40)
#define  Gksm_pat_ref_point    (41)
#define  Gksm_pat_vecs         (42)
#define  Gksm_asf              (43)
#define  Gksm_pick_id          (44)
#define  Gksm_line_rep         (51)
#define  Gksm_marker_rep       (52)
#define  Gksm_text_rep         (53)
#define  Gksm_fill_rep         (54)
#define  Gksm_pat_rep          (55)
#define  Gksm_colr_rep         (56)
#define  Gksm_clip_rect        (61)
#define  Gksm_ws_win           (71)
#define  Gksm_ws_vp            (72)
#define  Gksm_create_seg       (81)
#define  Gksm_close_seg        (82)
#define  Gksm_rename_seg       (83)
#define  Gksm_del_seg          (84)
```

```
#define Gksm_set_seg_tran   (91)
#define Gksm_set_vis        (92)
#define Gksm_set_highl      (93)
#define Gksm_set_seg_pri    (94)
#define Gksm_set_det        (95)
#define Gksm_user_item_start    (100)
/* Miscellaneous */
#define GDEF_MEM_SIZE      (0)
/* GKS functions */
/* Renamed GKS functions */
/* Output attribute functions */
extern void gset_fill_ind(
        Gint                fill_ind/* fill area index*/);
extern void gset_fill_int_style(
        Gfill_int_style     fill_int_style/* fill area interior style*/);
extern void gset_fill_style_ind(
        Gint                fill_style_ind/* fill area style index*/);
extern void gset_fill_colr_ind(
        Gint                fill_colr_ind/* fill area colour index*/);
extern void gset_fill_rep(
        Gint                ws_id,/* workstation identifier*/
        Gint                fill_ind,/* fill area index*/
        const Gfill_bundle  *fill_bundle/* fill area representation */);
/* Inquire functions */
extern void ginq_fill_ind(
        Gint                *err_ind,/* OUT error indicator*/
        Gint                *fill_ind/* OUT current fill area index*/);
extern void ginq_fill_int_style(
        Gint                *err_ind,/* OUT error indicator*/
        Gfill_int_style     *fill_int_style/* OUT current fill area
                                   interior style*/);
extern void ginq_fill_style_ind(
        Gint                *err_ind,/* OUT error indicator*/
        Gint                *fill_style_ind/* OUT current fill area style index*/);
extern void ginq_fill_colr_ind(
        Gint                *err_ind,/* OUT error indicator*/
        Gint                *fill_colr_ind/* OUT current fill area colour index*/);
extern void ginq_list_fill_inds(
        Gint                ws_id,/* workstation identifier*/
        Gint                num_elems_appl_list,/* length of application list*/
        Gint                start_ind,/* starting index*/
        Gint                *err_ind,/* OUT error indicator*/
        Gint_list           *def_fill_inds,/* OUT list of defined fill area indices*/
        Gint                *num_elems_impl_list/* OUT length of impl. list */);
extern void ginq_fill_rep(
        Gint                ws_id,/* workstation identifier*/
        Gint                fill_ind,/* fill area index*/
        Ginq_type           type,/* type of returned values*/
        Gint                *err_ind,/* OUT error indicator*/
        Gfill_bundle        *fill_rep/* OUT fill area representation*/);
extern void ginq_fill_facs(
        Gint                ws_gen_type,/* workstation generic type*/
        Gint                num_elems_appl_list,/* length of application list*/
        Gint                start_ind,/* starting index*/
        Gint                *err_ind,/* OUT error indicator*/
        Gfill_facs          *fill_facs,/* OUT fill area facilities*/
        Gint                *num_elems_impl_list/* OUT length of hatch list */);
extern void ginq_pred_fill_rep(
        Gint                ws_gen_type,/* workstation generic type*/
        Gint                ind,/* predefined index*/
        Gint                *err_ind,/* OUT error indicator*/
        Gfill_bundle        *fill_rep/* OUT predefined fill area rep.*/);
/* Overtaken functions */
/* Control functions */
extern void gredraw_all_segs_ws(
        Gint                ws_id/* workstation identifier*/);
extern void gupd_ws(
        Gint                ws_id,/* workstation identifier*/
        Gupd_regen_flag     upd_regen_flag/* update regeneration flag*/);
extern void gset_defer_st(
        Gint                ws_id,/* workstation identifier*/
        Gdefer_mode         defer_mode,/* deferral mode*/
        Girg_mode           irg_mode/* implicit regeneration mode*/);
/* Transformation functions */
extern void gset_clip_ind(
        Gclip_ind           clip_ind/* clipping indicator*/);
/* Input functions */
extern void gflush_events(
        Gint                ws_id,/* workstation identifier*/
        Gin_class           class,/* device class*/
        Gint                in_num/* logical input device number*/);
/* GKSM functions */
extern void gwrite_item(
        Gint                ws_id,/* workstation identifier*/
        Gint                item_type,/* item type*/
        Gint                item_data_length,/* item data record length*/
        const Gitem_data    *item_data/* item data record*/);
extern void gget_item_type(
        Gint                ws_id,/* workstation identifier*/
        Gint                *item_type,/* OUT item type*/
        Gint                *item_data_length/* OUT item data record length*/);
extern void gread_item(
        Gint                ws_id,/* workstation identifier*/
        Gint                max_item_data_length,/* max. item data record length*/
        Gitem_data          *item_data/* OUT item data record*/);
extern void ginterpret_item(
        Gint                type,/* item type*/
        Gint                item_data_length,/* item data record length*/
        const Gitem_data    *item_data/* item data record*/);
/* Inquire functions */
extern void ginq_op_st(
        Gop_st              *op_st/* OUT operating state value*/);
extern void ginq_level_gks(
        Gint                *err_ind,/* OUT error indicator*/
        Glevel              *level/* OUT level of GKS */);
extern void ginq_ws_max_nums(
        Gint                *err_ind,/* OUT error indicator*/
        Gws_max_nums        *ws_max_num/* OUT workstation maximum numbers */);
extern void ginq_max_norm_tran_num(
        Gint                *err_ind,/* OUT error indicator*/
        Gint                *max_norm_tran_num/* OUT maximum normalization
                                   transformation number */);
extern void ginq_cur_prim_attrs(
        Gint                *err_ind,/* OUT error indicator*/
        Gprim_attrs         *prim_attrs/* OUT current primitive
                                   attribute structure*/);
extern void ginq_char_width(
        Gint                *err_ind,/* OUT error indicator*/
        Gfloat              *char_width/* OUT current character width*/);
extern void ginq_char_base_vec(
        Gint                *err_ind,/* OUT error indicator*/
        Gvec                *char_base_vec/* OUT current character base vector*/);
extern void ginq_pat_width_vec(
        Gint                *err_ind,/* OUT error indicator*/
        Gvec                *pat_width_vec/* OUT current pattern width vector*/);
extern void ginq_pat_ht_vec(
        Gint                *err_ind,/* OUT error indicator*/
        Gvec                *pat_ht_vec/* OUT current pattern height vector*/);
extern void ginq_cur_indiv_attrs(
        Gint                *err_ind,/* OUT error indicator*/
        Gindiv_attrs        *indiv_attr/* OUT current individual
                                   attribute  structure*/);
extern void ginq_clip(
        Gint                *err_ind,/* OUT error indicator*/
        Gclip               *clip_ind_rect/* OUT current clipping indicator
                                   and rectangle*/);
extern void ginq_more_simult_events(
        Gint                *err_ind,/* OUT error indicator*/
        Gmore_simult_events *simult_events/* OUT [more]
                                   simultaneous events*/);
extern void ginq_ws_defer_upd_sts(
        Gint                ws_id,/* workstation identifier*/
        Gint                *err_ind,/* OUT error indicator*/
        Gdefer_mode         *defer_mode,/* OUT deferral mode*/
        Girg_mode           *irg_mode,/* OUT IRG mode*/
        Gdisp_surf_empty    *disp_surf_empty,/* OUT display surface empty*/
        Gnew_frame_nec_upd  *new_frame/* OUT new frame action
                                   necessary at update*/);
extern void ginq_ws_tran(
        Gint                ws_id,/* workstation identifier*/
        Gint                *err_ind,/* OUT error indicator*/
        Gupd_st             *ws_tran_upd_st,/* OUT workstation transformation
                                   update state*/
        Glimit              *req_ws_win,/* OUT requested workstation window*/
        Glimit              *cur_ws_win,/* OUT current workstation window*/
        Glimit              *req_ws_vp,/* OUT requested workstation viewport*/
        Glimit               *cur_ws_vp/* OUT current workstation viewport*/);
extern void ginq_ws_cat(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gws_cat              *cat/* OUT workstation category*/);
extern void ginq_dyn_mod_ws_attrs(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gdyn_mod_ws_attrs    *dyn_mod/* OUT dynamic modification of
                                    workstation attributes */);
extern void ginq_def_defer_sts(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gdefer_mode          *defer_mode,/* OUT default deferral mode */
        Girg_mode            *irg_mode/* OUT default impl. reg. mode */);
extern void ginq_gdp(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 gdp,/* GDP function number*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gint                 *num_attr,/* OUT num. of attributes used*/
        Gattrs               attr[4]/* OUT list of attributes used*/);
extern void ginq_max_ws_st_tables(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gmax_ws_st_tables    *lengths/* OUT lengths of workstation tables*/);
extern void ginq_num_seg_pris(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gint                 *num_seg_pris/* OUT num. of segment priorities*/);
extern void ginq_dyn_mod_seg_attrs(
        Gint                 ws_gen_type,/* workstation generic type*/
        Gint                 *err_ind,/* OUT error indicator*/
        Gdyn_mod_seg_attrs   *dyn_mod/* OUT dynamic modification of
                                    segment attributess*/);
extern void ginq_pixel_array_dims(
        Gint                 ws_id,/* workstation identifier*/
        const Grect          *rect,/* rectangle */
        Gint                 *err_ind,/* OUT error indicator*/
        Gint_size            *dims/* OUT pixel array dimensions */);
extern void ginq_pixel_array(
        Gint                 ws_id,/* workstation identifier*/
        const Gpoint         *pixel_loc,/* pixel location */
        const Gint_size      *dims,/* pixel array dimensions */
        Gint                 *err_ind,/* OUT error indicator*/
        Gpres_inval          *pres_inval,/* OUT presence of invalid values*/
        Gint                 *pixel_array/* OUT colour specifier array*/);
extern void ginq_pixel(
        Gint                 ws_id,/* workstation identifier*/
        const Gpoint         *pixel_loc,/* pixel location */
        Gint                 *err_ind,/* OUT error indicator*/
        Gint                 *colr_ind/* OUT colour index*/);
```

Annex B
(informative)

Sample programs

B.1 STAR

```
/*      Program:      STAR
 *      File:                          star.c        5.1
 *
 *      Description
 *
 *      This program draws a yellow star on a blue background and writes
 *      The title "STAR" in white under the star.
 */

/*
 *      Include files
 */
#include <stdio.h>
#include <math.h>
#include "gks.h"                         /* GKS definitions */
/*
 *      Local definitions (implementation and site dependent)
 */
#define  GKS_ERR        "gks.err"      /* Name of the GKS error file */
#define  WS_SPECIF      "file_star"    /* Workstation specifier */
#define  WS_TYPE        (19)           /* Workstation type */
/*
 *      Convenient definitions
 */
#define  MY_WS          (1)            /* The workstation we'll be using */
#define  TRAN_NUM       (1)            /* Normalization transformation  number */
#define  BLUE           (0)            /* Some colours we define below */
#define  YELLOW         (1)
#define  WHITE          (2)
#define  PI             (3.14159265)   /* The well-known number */
#define  LARGE_TEXT     (0.15)         /* Size of large texts */

/*
 *      State Variables:
 *
 *                     These variables are used to set the
 *                     corresponding GKS state
 */

static Glimit          win = {         /* Window Limits */
        -1.25, 1.25, -1.25, 1.25};

Gcolr_rep              colr[3] ;       /* Colour Table */
```

```
static Gtext_align      ct_half = {    /* Text Alignment */
        GHOR_CTR, GVERT_HALF};

static Gpoint           title_pos =    /* Position for title */
        {0.0, -1.1};

static char             title[] = "STAR";             /* Title text */


/* Main entry point */
int main()
{
        Gpoint_list     star;          /* Points of the star */
        double angle;  /* Current angle */
        Gpoint *starptr;               /* Star point pointer */
        Gint colr_ind; /* Colour index */
        Gcolr_rep *colrptr;            /* Colour representation pointer */
        Gint i;

/*
 *      colours are set
 *
 *      colr[0] = blue;
 *      colr[1] = yellow;
 *      colr[2] = white;
 */

        for (i=0; i<3; i++){
                        colr[i].rgb.red = 1.0;
                        colr[i].rgb.green = 1.0;
                        colr[i].rgb.blue = 1.0;
        }
        colr[0].rgb.red = colr[0].rgb.green = colr[1].rgb.blue = 0.0;
/*
 *      Define the coordinates of star
 */

        star.num_points = 5;
        star.points = calloc(5, sizeof(Gpoint));
        for (starptr = star.points, angle = 0.0;
                        angle < (4.*PI); angle += 0.8*PI, starptr ++) {
                        starptr->x = (Gfloat)sin(angle);
                        starptr->y = (Gfloat)cos(angle);
        }
/*
 *      Open GKS and open/ and activate the workstation
 */

        gopen_gks(GKS_ERR, 0);
        gopen_ws(MY_WS, WS_SPECIF,WS_TYPE);
        gactivate_ws(MY_WS);

/*
```