INTERNATIONAL STANDARD

ISO/IEC 8651-4

First edition
1991-12-15

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

Part 4:
C

Technologies de l'information — Infographie — Système graphique de base (GKS) — Interface langage —

Partie 4: C

Reference number
ISO/IEC 8651-4:1991(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*.

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

— *Part 1: FORTRAN 77*

— *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 7942 : 1985. As explained in the Scope and Field of Application of ISO 7942, 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 7942 : 1985 , 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 provision of this part of ISO/IEC 8651. At the time of publication, the editions indicated were valid. All standards are subject to revisions, and parties to agreements based on this part of ISO/IEC 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 7942:1985, *Information processing systems – Computer graphics – Graphical Kernel System (GKS) functional description.*

ISO/IEC 8651-1:1988, *Information processing systems – Computer graphics – Graphical Kernel System (GKS) - language bindings - Part 1 : FORTRAN .*

ISO/IEC 8806-4:1991, *Information technology – Computer graphics – Graphical Kernel System for Three Dimensions (GKS-3D) language bindings - Part 4 : C.*

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

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

3 The C Language Binding of GKS

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

3.1 Conformance

This part of ISO/IEC 8651 incorporates the rules of conformance defined in the GKS Standard (ISO 7942) 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 a specific level of GKS as specified in ISO 7942. It shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 8651 for that same level of GKS and all lower levels and for a specific level of C.

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

3.2 Functions versus Macros

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

3.3 Character Strings

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

3.4 Function Identifiers

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

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

3.5 Registration

ISO 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 and IEC councils have designated the National Institute of Standards and Technology (Institute of Computer Sciences and Technology), A-266 Technology Building, Gaithersburg, MD 20899, USA to act as registration authority.

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)   /* 'n' is the registered GDP id. */
```

The format for unregistered GDP identifiers is:

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

The format for registered Escape function identifiers is:

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

The format for unregistered Escape function identifiers is:

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

3.7 Return Values

All GKS/C functions return `void`.

3.8 Header Files

C provides a mechanism to allow external files to be included in a compilation. Clause 5 of this part of ISO/IEC 8651 describes the data types that shall be defined in the file `gks.h` which should be included in any application program that intends to use GKS via the C binding.

This part of ISO/IEC 8651 uses the data type `size_t` (as a field in the data type `Gdata`). The type `size_t` is environment-dependent (e.g. `int, long int, unsigned int`) and is defined in the file `<stddef.h>`. Therefore the file `gks.h` shall also include the file `<stddef.h>`.

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

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

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

3.9 Memory Management

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

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

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 E, a possible underlying mechanism is described.

3.9.2 Functions which Return Complex Data Structures

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

The *Store* resource is opaque to the application. The application does not know the structure of the *Store* or how it is implemented. The *Store* is defined as a `void *`. This part of ISO/IEC 8651 defines two new functions which create (in CREATE STORE, bound as `gcreate_store`) and delete (in DELETE STORE, bound as `gdel_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 new errors that can occur when using or creating a *Store*; these errors are described in 3.10.3. For most functions using a *Store*, these and other errors are returned via the "error indicator" parameter. However, the function ESCAPE does not have an error indicator parameter. For this function, the error reporting mechanism is used when an error is encountered. For this function, the implementation shall, in addition to reporting the error, set the pointer to the returned data to NULL when an error occurs. See the binding of these functions for more information.

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

CREATE STORE **GKOP, WSOP, WSAC, SGOP** **L0a**

Parameters:

Out	error indicator	I
Out	store	STORE

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

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

Errors: None.

DELETE STORE **GKOP, WSOP, WSAC, SGOP** **L0a**

Parameters:

Out	error indicator	I
In/Out	store	STORE

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

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

Errors: None.

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

3.10 Error Handling

3.10.1 Application Supplied Error Handlers

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

SET ERROR HANDLER **GKCL, GKOP, WSOP, WSAC, SGOP** **L0a**

Parameters:

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

Effect: Sets the GKS 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 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 knows some specific error messages. This section gives the messages and the error macros.

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

3.11 Colour Representations

A union type definition is used for colour bundles, guaranteeing upward compatibility with GKS-3D/C (ISO/IEC 8806-4), which supports at least two colour models (RGB and CIE L*u*v* 1976).

3.12 Storage of Multi-dimensional Arrays

3.12.1 Storage of 2*3 Matrices

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

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

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

3.12.2 Storage of Colour Arrays

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

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

where `colr_rect` is of type `Gpat_rep`.

4 Tables

4.1 Abbreviation Policy in Construction of Identifiers

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

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

4.2 Table of Abbreviations Used

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

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

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

Word or Phrase	Abbreviation
accepted	NULL

Table of Abbreviations Used

accumulate	accum
action	NULL
actual	act
addition	add
alignment	align
allocate	alloc
and	NULL
application	appl
arithmetic	arith
as soon as possible	asap
aspect source flag	asf
associate(d)	assoc
at some time	asti
at	NULL
attribute	attr
availability	avail
available	avail
before the next interaction globally	bnig
before the next interaction locally	bnil
between	NULL
buffer	buf
cannot	cant
category	cat
centre	ctr
character	char
classification	class
clipping	clip
colour	colr
concatenation	concat
conditional(ly)	cond
connection	conn
control	ctrl
coordinate	coord
corner	cnr
current	cur
dashed	dash
default	def
deferral	defer
delete	del
deletion	del
detectability	det
detectable	det
device coordinate(s)	dc
device(s)	NULL
digital	digit
dimension	dim
display	disp
dotted	dot
duplicate	dup
dynamic	dyn
equal	eq
error	err

evaluate	eval
expansion	expan
facility	fac
factor	NULL
fill area	fill
from	NULL
function	func
generalized drawing primitive	gdp
gks closed	gkcl
gks open	gkop
gksm	NULL
graphical	graph
handling	hand
height	ht
highlighted	highl
highlighting	highl
horizontal	hor
identifier	id
immediate(ly)	imm
implicit regeneration	irg
implicit	impl
in use	NULL
in	NULL
index	ind
indicator	ind
individual	indiv
initial	init
initialize	init
input/output	io
input	in
inquire	inq
integer	int
interior	int
invalid	inval
invisible	invis
length	NULL
less than or equal	le
less than	lt
level x	lx (with x = 0a, 0b, ...)
library	lib
locator	loc
logical	NULL
mapping	map
maximum	max
memory	mem
metafile input	mi
metafile output	mo
minimum	min
modification	mod
monochrome	monochr
necessary	nec
nominal	nom

Table of Abbreviations Used

normal	norm
normalization	norm
normalized device coordinate(s)	ndc
number	num
of	NULL
on	NULL
operating	op
output and input	outin
output	out
overflow	overf
parallel(ogram)	paral
pattern	pat
pending	pend
pointer	ptr
polyline	line
polymarker	marker
position	pos
precision	prec
predefined	pred
presence	pres
primitive	prim
priority	pri
prompt and echo type	pet
prompt	pr
queue	NULL
record	NULL
rectangle	rect
red green blue (colour model)	rgb
reference	ref
regeneration	regen
registered	r
relative	rel
representation	rep
request(ed)	req
rubber	rub
scale factor	NULL
segment open	sgop
segment	seg
select	sel
simultaneous(ly)	simult
spacing	space
specification	specif
specified	specif
state	st
storage	store
supported	NULL
suppressed	suppr
to	NULL
tracking	track
transform(ation)	tran
unavailable	unavail
undefined	undef

undetectable	undet
unregistered	u
update	upd
used	NULL
valuator	val
value	NULL
vector	vec
vertical	vert
viewport	vp
visibility	vis
visible	vis
which	NULL
window	win
with	NULL
workstation active	wsac
workstation independent segment store	wiss
workstation open	wsop
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
gassoc_seg_ws	ASSOCIATE SEGMENT WITH WORKSTATION
gawait_event	AWAIT EVENT
gcell_array	CELL ARRAY
gclear_ws	CLEAR WORKSTATION
gclose_gks	CLOSE GKS
gclose_seg	CLOSE SEGMENT
gclose_ws	CLOSE WORKSTATION
gcopy_seg_ws	COPY SEGMENT FROM WORKSTATION
gcreate_seg	CREATE SEGMENT
gcreate_store	CREATE STORE (GKS/C only)
gdeactivate_ws	DEACTIVATE WORKSTATION
gdel_seg	DELETE SEGMENT
gdel_seg_ws	DELETE SEGMENT FROM WORKSTATION
gdel_store	DELETE STORE (GKS/C only)
gemergency_close_gks	EMERGENCY CLOSE GKS
gerr_hand	ERROR HANDLING
gerr_log	ERROR LOGGING
gescape	ESCAPE
geval_tran_matrix	EVALUATE TRANSFORMATION MATRIX
gfill_area	FILL AREA
gflush_events	FLUSH DEVICE EVENTS
ggdp	GENERALIZED DRAWING PRIMITIVE
gget_choice	GET CHOICE
gget_item_type	GET ITEM TYPE FROM GKSM
gget_loc	GET LOCATOR
gget_pick	GET PICK

gget_string	GET STRING
gget_stroke	GET STROKE
gget_val	GET VALUATOR
ginit_choice	INITIALISE CHOICE
ginit_loc	INITIALISE LOCATOR
ginit_pick	INITIALISE PICK
ginit_string	INITIALISE STRING
ginit_stroke	INITIALISE STROKE
ginit_val	INITIALISE VALUATOR
ginq_asfs	INQUIRE ASPECT SOURCE FLAGS
ginq_char_base_vec	INQUIRE CHARACTER BASE VECTOR
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_char_width	INQUIRE CHARACTER WIDTH
ginq_choice_st	INQUIRE CHOICE DEVICE STATE
ginq_clip	INQUIRE CLIPPING
ginq_colr_facs	INQUIRE COLOUR FACILITIES
ginq_colr_rep	INQUIRE COLOUR REPRESENTATION
ginq_cur_indiv_attrs	INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES
ginq_cur_norm_tran_num	INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER
ginq_cur_pick_id	INQUIRE CURRENT PICK IDENTIFIER VALUE
ginq_cur_prim_attrs	INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES
ginq_def_choice_data	INQUIRE DEFAULT CHOICE DEVICE DATA
ginq_def_defer_sts	INQUIRE DEFAULT DEFERRAL STATE VALUES
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_dyn_mod_seg_attrs	INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES
ginq_dyn_mod_ws_attrs	INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES
ginq_fill_colr_ind	INQUIRE FILL AREA COLOUR INDEX
ginq_fill_facs	INQUIRE FILL AREA FACILITIES
ginq_fill_ind	INQUIRE FILL AREA INDEX
ginq_fill_int_style	INQUIRE FILL AREA INTERIOR STYLE
ginq_fill_rep	INQUIRE FILL AREA REPRESENTATION
ginq_fill_style_ind	INQUIRE FILL AREA STYLE INDEX
ginq_gdp	INQUIRE GENERALIZED DRAWING PRIMITIVE
ginq_in_overf	INQUIRE INPUT QUEUE OVERFLOW
ginq_level_gks	INQUIRE LEVEL OF GKS
ginq_line_colr_ind	INQUIRE POLYLINE COLOUR INDEX
ginq_line_facs	INQUIRE POLYLINE FACILITIES
ginq_line_ind	INQUIRE POLYLINE INDEX
ginq_line_rep	INQUIRE POLYLINE REPRESENTATION
ginq_linetype	INQUIRE LINETYPE
ginq_linewidth	INQUIRE LINEWIDTH SCALE FACTOR
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_fill_inds	INQUIRE LIST OF FILL AREA INDICES
ginq_list_line_inds	INQUIRE LIST OF POLYLINE INDICES
ginq_list_marker_inds	INQUIRE LIST OF POLYMARKER INDICES
ginq_list_norm_tran_nums	INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS
ginq_list_pat_inds	INQUIRE LIST OF PATTERN INDICES
ginq_list_text_inds	INQUIRE LIST OF TEXT INDICES
ginq_loc_st	INQUIRE LOCATOR DEVICE STATE
ginq_marker_colr_ind	INQUIRE POLYMARKER COLOUR INDEX
ginq_marker_facs	INQUIRE POLYMARKER FACILITIES
ginq_marker_ind	INQUIRE POLYMARKER INDEX
ginq_marker_rep	INQUIRE POLYMARKER REPRESENTATION
ginq_marker_size	INQUIRE MARKER SIZE SCALE FACTOR
ginq_marker_type	INQUIRE MARKER TYPE
ginq_max_norm_tran_num	INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER
ginq_max_ws_st_tables	INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES
ginq_more_simult_events	INQUIRE MORE SIMULTANEOUS EVENTS
ginq_name_open_seg	INQUIRE NAME OF OPEN SEGMENT
ginq_norm_tran	INQUIRE NORMALIZATION TRANSFORMATION
ginq_num_avail_in	INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES
ginq_num_seg_pris	INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED
ginq_op_st	INQUIRE OPERATING STATE VALUE
ginq_pat_facs	INQUIRE PATTERN FACILITIES
ginq_pat_ht_vec	INQUIRE PATTERN HEIGHT VECTOR
ginq_pat_ref_point	INQUIRE PATTERN REFERENCE POINT
ginq_pat_rep	INQUIRE PATTERN REPRESENTATION
ginq_pat_width_vec	INQUIRE PATTERN WIDTH VECTOR
ginq_pick_st	INQUIRE PICK DEVICE STATE
ginq_pixel	INQUIRE PIXEL
ginq_pixel_array	INQUIRE PIXEL ARRAY
ginq_pixel_array_dims	INQUIRE PIXEL ARRAY DIMENSIONS
ginq_pred_colr_rep	INQUIRE PREDEFINED COLOUR REPRESENTATION
ginq_pred_fill_rep	INQUIRE PREDEFINED FILL AREA REPRESENTATION
ginq_pred_line_rep	INQUIRE PREDEFINED POLYLINE REPRESENTATION
ginq_pred_marker_rep	INQUIRE PREDEFINED POLYMARKER REPRESENTATION
ginq_pred_pat_rep	INQUIRE PREDEFINED PATTERN REPRESENTATION
ginq_pred_text_rep	INQUIRE PREDEFINED TEXT REPRESENTATION
ginq_seg_attrs	INQUIRE SEGMENT ATTRIBUTES
ginq_set_active_wss	INQUIRE SET OF ACTIVE WORKSTATIONS
ginq_set_assoc_wss	INQUIRE SET OF ASSOCIATED WORKSTATIONS
ginq_set_open_wss	INQUIRE SET OF OPEN WORKSTATIONS
ginq_set_seg_names	INQUIRE SET OF SEGMENT NAMES IN USE
ginq_set_seg_names_ws	INQUIRE SET OF SEGMENT NAMES ON WORKSTATION
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_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_val_st	INQUIRE VALUATOR DEVICE STATE
ginq_ws_cat	INQUIRE WORKSTATION CATEGORY
ginq_ws_class	INQUIRE WORKSTATION CLASSIFICATION
ginq_ws_conn_type	INQUIRE WORKSTATION CONNECTION AND TYPE
ginq_ws_defer_upd_sts	INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES
ginq_ws_max_nums	INQUIRE WORKSTATION MAXIMUM NUMBERS
ginq_ws_st	INQUIRE WORKSTATION STATE
ginq_ws_tran	INQUIRE WORKSTATION TRANSFORMATION
ginsert_seg	INSERT SEGMENT
ginterpret_item	INTERPRET ITEM
gmessage	MESSAGE
gopen_gks	OPEN GKS
gopen_ws	OPEN WORKSTATION
gpolyline	POLYLINE
gpolymarker	POLYMARKER
gread_item	READ ITEM FROM GKSM
gredraw_all_segs_ws	REDRAW ALL SEGMENTS ON WORKSTATION
grename_seg	RENAME SEGMENT
greq_choice	REQUEST CHOICE
greq_loc	REQUEST LOCATOR
greq_pick	REQUEST PICK
greq_string	REQUEST STRING
greq_stroke	REQUEST STROKE
greq_val	REQUEST VALUATOR
gsample_choice	SAMPLE CHOICE
gsample_loc	SAMPLE LOCATOR
gsample_pick	SAMPLE PICK
gsample_string	SAMPLE STRING
gsample_stroke	SAMPLE STROKE
gsample_val	SAMPLE VALUATOR
gsel_norm_tran	SELECT NORMALIZATION TRANSFORMATION
gset_asfs	SET ASPECT SOURCE FLAGS
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_clip_ind	SET CLIPPING INDICATOR
gset_colr_rep	SET COLOUR REPRESENTATION
gset_defer_st	SET DEFERRAL STATE
gset_det	SET DETECTABILITY
gset_err_hand	SET ERROR HANDLER (GKS/C only)
gset_fill_colr_ind	SET FILL AREA COLOUR INDEX
gset_fill_ind	SET FILL AREA INDEX
gset_fill_int_style	SET FILL AREA INTERIOR STYLE
gset_fill_rep	SET FILL AREA REPRESENTATION
gset_fill_style_ind	SET FILL AREA STYLE INDEX
gset_highl	SET HIGHLIGHTING
gset_line_colr_ind	SET POLYLINE COLOUR INDEX
gset_line_ind	SET POLYLINE INDEX
gset_line_rep	SET POLYLINE REPRESENTATION
gset_linetype	SET LINETYPE

gset_linewidth	SET LINEWIDTH SCALE FACTOR
gset_loc_mode	SET LOCATOR MODE
gset_marker_colr_ind	SET POLYMARKER COLOUR INDEX
gset_marker_ind	SET POLYMARKER INDEX
gset_marker_rep	SET POLYMARKER REPRESENTATION
gset_marker_size	SET MARKER SIZE SCALE FACTOR
gset_marker_type	SET MARKER TYPE
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_seg_pri	SET SEGMENT PRIORITY
gset_seg_tran	SET SEGMENT TRANSFORMATION
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_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_val_mode	SET VALUATOR MODE
gset_vis	SET VISIBILITY
gset_vp	SET VIEWPORT
gset_vp_in_pri	SET VIEWPORT INPUT PRIORITY
gset_win	SET WINDOW
gset_ws_vp	SET WORKSTATION VIEWPORT
gset_ws_win	SET WORKSTATION WINDOW
gtext	TEXT
gupd_ws	UPDATE WORKSTATION
gwrite_item	WRITE ITEM TO GKSM

4.3.2 List Ordered Alphabetically by GKS Name

ACCUMULATE TRANSFORMATION MATRIX	gaccum_tran_matrix
ACTIVATE WORKSTATION	gactivate_ws
ASSOCIATE SEGMENT WITH WORKSTATION	gassoc_seg_ws
AWAIT EVENT	gawait_event
CELL ARRAY	gcell_array
CLEAR WORKSTATION	gclear_ws
CLOSE GKS	gclose_gks
CLOSE SEGMENT	gclose_seg
CLOSE WORKSTATION	gclose_ws
COPY SEGMENT FROM WORKSTATION	gcopy_seg_ws
CREATE SEGMENT	gcreate_seg
CREATE STORE (GKS/C only)	gcreate_store
DEACTIVATE WORKSTATION	gdeactivate_ws
DELETE SEGMENT	gdel_seg
DELETE SEGMENT FROM WORKSTATION	gdel_seg_ws
DELETE STORE (GKS/C only)	gdel_store
EMERGENCY CLOSE GKS	gemergency_close_gks

Function Names **Tables**

ERROR HANDLING	gerr_hand
ERROR LOGGING	gerr_log
ESCAPE	gescape
EVALUATE TRANSFORMATION MATRIX	geval_tran_matrix
FILL AREA	gfill_area
FLUSH DEVICE EVENTS	gflush_events
GENERALIZED DRAWING PRIMITIVE	ggdp
GET CHOICE	gget_choice
GET ITEM TYPE FROM GKSM	gget_item_type
GET LOCATOR	gget_loc
GET PICK	gget_pick
GET STRING	gget_string
GET STROKE	gget_stroke
GET VALUATOR	gget_val
INITIALISE CHOICE	ginit_choice
INITIALISE LOCATOR	ginit_loc
INITIALISE PICK	ginit_pick
INITIALISE STRING	ginit_string
INITIALISE STROKE	ginit_stroke
INITIALISE VALUATOR	ginit_val
INQUIRE ASPECT SOURCE FLAGS	ginq_asfs
INQUIRE CHARACTER BASE VECTOR	ginq_char_base_vec
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 CHARACTER WIDTH	ginq_char_width
INQUIRE CHOICE DEVICE STATE	ginq_choice_st
INQUIRE CLIPPING	ginq_clip
INQUIRE COLOUR FACILITIES	ginq_colr_facs
INQUIRE COLOUR REPRESENTATION	ginq_colr_rep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	ginq_cur_indiv_attrs
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	ginq_cur_norm_tran_num
INQUIRE CURRENT PICK IDENTIFIER VALUE	ginq_cur_pick_id
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	ginq_cur_prim_attrs
INQUIRE DEFAULT CHOICE DEVICE DATA	ginq_def_choice_data
INQUIRE DEFAULT DEFERRAL STATE VALUES	ginq_def_defer_sts
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 DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	ginq_dyn_mod_seg_attrs
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	ginq_dyn_mod_ws_attrs
INQUIRE FILL AREA COLOUR INDEX	ginq_fill_colr_ind
INQUIRE FILL AREA FACILITIES	ginq_fill_facs
INQUIRE FILL AREA INDEX	ginq_fill_ind
INQUIRE FILL AREA INTERIOR STYLE	ginq_fill_int_style
INQUIRE FILL AREA REPRESENTATION	ginq_fill_rep
INQUIRE FILL AREA STYLE INDEX	ginq_fill_style_ind
INQUIRE GENERALIZED DRAWING PRIMITIVE	ginq_gdp

Tables	Function Names
INQUIRE INPUT QUEUE OVERFLOW	ginq_in_overf
INQUIRE LEVEL OF GKS	ginq_level_gks
INQUIRE LINETYPE	ginq_linetype
INQUIRE LINEWIDTH SCALE FACTOR	ginq_linewidth
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 FILL AREA INDICES	ginq_list_fill_inds
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	ginq_list_norm_tran_nums
INQUIRE LIST OF PATTERN INDICES	ginq_list_pat_inds
INQUIRE LIST OF POLYLINE INDICES	ginq_list_line_inds
INQUIRE LIST OF POLYMARKER INDICES	ginq_list_marker_inds
INQUIRE LIST OF TEXT INDICES	ginq_list_text_inds
INQUIRE LOCATOR DEVICE STATE	ginq_loc_st
INQUIRE MARKER SIZE SCALE FACTOR	ginq_marker_size
INQUIRE MARKER TYPE	ginq_marker_type
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	ginq_max_ws_st_tables
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	ginq_max_norm_tran_num
INQUIRE MORE SIMULTANEOUS EVENTS	ginq_more_simult_events
INQUIRE NAME OF OPEN SEGMENT	ginq_name_open_seg
INQUIRE NORMALIZATION TRANSFORMATION	ginq_norm_tran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	ginq_num_avail_in
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	ginq_num_seg_pris
INQUIRE OPERATING STATE VALUE	ginq_op_st
INQUIRE PATTERN FACILITIES	ginq_pat_facs
INQUIRE PATTERN HEIGHT VECTOR	ginq_pat_ht_vec
INQUIRE PATTERN REFERENCE POINT	ginq_pat_ref_point
INQUIRE PATTERN REPRESENTATION	ginq_pat_rep
INQUIRE PATTERN WIDTH VECTOR	ginq_pat_width_vec
INQUIRE PICK DEVICE STATE	ginq_pick_st
INQUIRE PIXEL	ginq_pixel
INQUIRE PIXEL ARRAY	ginq_pixel_array
INQUIRE PIXEL ARRAY DIMENSIONS	ginq_pixel_array_dims
INQUIRE POLYLINE COLOUR INDEX	ginq_line_colr_ind
INQUIRE POLYLINE FACILITIES	ginq_line_facs
INQUIRE POLYLINE INDEX	ginq_line_ind
INQUIRE POLYLINE REPRESENTATION	ginq_line_rep
INQUIRE POLYMARKER COLOUR INDEX	ginq_marker_colr_ind
INQUIRE POLYMARKER FACILITIES	ginq_marker_facs
INQUIRE POLYMARKER INDEX	ginq_marker_ind
INQUIRE POLYMARKER REPRESENTATION	ginq_marker_rep
INQUIRE PREDEFINED COLOUR REPRESENTATION	ginq_pred_colr_rep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	ginq_pred_fill_rep
INQUIRE PREDEFINED PATTERN REPRESENTATION	ginq_pred_pat_rep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	ginq_pred_line_rep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	ginq_pred_marker_rep
INQUIRE PREDEFINED TEXT REPRESENTATION	ginq_pred_text_rep
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 OPEN WORKSTATIONS	ginq_set_open_wss
INQUIRE SET OF SEGMENT NAMES IN USE	ginq_set_seg_names

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	ginq_set_seg_names_ws
INQUIRE STRING DEVICE STATE	ginq_string_st
INQUIRE STROKE DEVICE STATE	ginq_stroke_st
INQUIRE TEXT ALIGNMENT	ginq_text_align
INQUIRE TEXT COLOUR INDEX	ginq_text_colr_ind
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 VALUATOR DEVICE STATE	ginq_val_st
INQUIRE WORKSTATION CATEGORY	ginq_ws_cat
INQUIRE WORKSTATION CLASSIFICATION	ginq_ws_class
INQUIRE WORKSTATION CONNECTION AND TYPE	ginq_ws_conn_type
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	ginq_ws_defer_upd_sts
INQUIRE WORKSTATION MAXIMUM NUMBERS	ginq_ws_max_nums
INQUIRE WORKSTATION STATE	ginq_ws_st
INQUIRE WORKSTATION TRANSFORMATION	ginq_ws_tran
INSERT SEGMENT	ginsert_seg
INTERPRET ITEM	ginterpret_item
MESSAGE	gmessage
OPEN GKS	gopen_gks
OPEN WORKSTATION	gopen_ws
POLYLINE	gpolyline
POLYMARKER	gpolymarker
READ ITEM FROM GKSM	gread_item
REDRAW ALL SEGMENTS ON WORKSTATION	gredraw_all_segs_ws
RENAME SEGMENT	grename_seg
REQUEST CHOICE	greq_choice
REQUEST LOCATOR	greq_loc
REQUEST PICK	greq_pick
REQUEST STRING	greq_string
REQUEST STROKE	greq_stroke
REQUEST VALUATOR	greq_val
SAMPLE CHOICE	gsample_choice
SAMPLE LOCATOR	gsample_loc
SAMPLE PICK	gsample_pick
SAMPLE STRING	gsample_string
SAMPLE STROKE	gsample_stroke
SAMPLE VALUATOR	gsample_val
SELECT NORMALIZATION TRANSFORMATION	gsel_norm_tran
SET ASPECT SOURCE FLAGS	gset_asfs
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 CLIPPING INDICATOR	gset_clip_ind
SET COLOUR REPRESENTATION	gset_colr_rep
SET DEFERRAL STATE	gset_defer_st
SET DETECTABILITY	gset_det

Tables	Function Names
SET ERROR HANDLER (GKS/C only)	gset_err_hand
SET FILL AREA COLOUR INDEX	gset_fill_colr_ind
SET FILL AREA INDEX	gset_fill_ind
SET FILL AREA INTERIOR STYLE	gset_fill_int_style
SET FILL AREA REPRESENTATION	gset_fill_rep
SET FILL AREA STYLE INDEX	gset_fill_style_ind
SET HIGHLIGHTING	gset_highl
SET LINETYPE	gset_linetype
SET LINEWIDTH SCALE FACTOR	gset_linewidth
SET LOCATOR MODE	gset_loc_mode
SET MARKER SIZE SCALE FACTOR	gset_marker_size
SET MARKER TYPE	gset_marker_type
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 POLYLINE COLOUR INDEX	gset_line_colr_ind
SET POLYLINE INDEX	gset_line_ind
SET POLYLINE REPRESENTATION	gset_line_rep
SET POLYMARKER COLOUR INDEX	gset_marker_colr_ind
SET POLYMARKER INDEX	gset_marker_ind
SET POLYMARKER REPRESENTATION	gset_marker_rep
SET SEGMENT PRIORITY	gset_seg_pri
SET SEGMENT TRANSFORMATION	gset_seg_tran
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 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 VALUATOR MODE	gset_val_mode
SET VIEWPORT	gset_vp
SET VIEWPORT INPUT PRIORITY	gset_vp_in_pri
SET VISIBILITY	gset_vis
SET WINDOW	gset_win
SET WORKSTATION VIEWPORT	gset_ws_vp
SET WORKSTATION WINDOW	gset_ws_win
TEXT	gtext
UPDATE WORKSTATION	gupd_ws
WRITE ITEM TO GKSM	gwrite_item

4.3.3 List Ordered Alphabetically by Bound Name within Level

4.3.3.1 Level 0A

gactivate_ws	ACTIVATE WORKSTATION
gcell_array	CELL ARRAY
gclear_ws	CLEAR WORKSTATION
gclose_gks	CLOSE GKS

gclose_ws	CLOSE WORKSTATION
gcreate_store	CREATE STORE (GKS/C only)
gdeactivate_ws	DEACTIVATE WORKSTATION
gdel_store	DELETE STORE (GKS/C only)
gemergency_close_gks	EMERGENCY CLOSE GKS
gerr_hand	ERROR HANDLING
gerr_log	ERROR LOGGING
gescape	ESCAPE
gfill_area	FILL AREA
ggdp	GENERALIZED DRAWING PRIMITIVE
gget_item_type	GET ITEM TYPE FROM GKSM
ginq_asfs	INQUIRE ASPECT SOURCE FLAGS
ginq_char_base_vec	INQUIRE CHARACTER BASE VECTOR
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_char_width	INQUIRE CHARACTER WIDTH
ginq_clip	INQUIRE CLIPPING
ginq_colr_facs	INQUIRE COLOUR FACILITIES
ginq_colr_rep	INQUIRE COLOUR REPRESENTATION
ginq_cur_indiv_attrs	INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES
ginq_cur_norm_tran_num	INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER
ginq_cur_prim_attrs	INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES
ginq_disp_space_size	INQUIRE DISPLAY SPACE SIZE
ginq_dyn_mod_ws_attrs	INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES
ginq_fill_colr_ind	INQUIRE FILL AREA COLOUR INDEX
ginq_fill_facs	INQUIRE FILL AREA FACILITIES
ginq_fill_ind	INQUIRE FILL AREA INDEX
ginq_fill_int_style	INQUIRE FILL AREA INTERIOR STYLE
ginq_fill_style_ind	INQUIRE FILL AREA STYLE INDEX
ginq_gdp	INQUIRE GENERALIZED DRAWING PRIMITIVE
ginq_level_gks	INQUIRE LEVEL OF GKS
ginq_linetype	INQUIRE LINETYPE
ginq_linewidth	INQUIRE LINEWIDTH SCALE FACTOR
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_norm_tran_nums	INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS
ginq_marker_size	INQUIRE MARKER SIZE SCALE FACTOR
ginq_marker_type	INQUIRE MARKER TYPE
ginq_max_ws_st_tables	INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES
ginq_max_norm_tran_num	INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER
ginq_norm_tran	INQUIRE NORMALIZATION TRANSFORMATION
ginq_op_st	INQUIRE OPERATING STATE VALUE
ginq_pat_facs	INQUIRE PATTERN FACILITIES
ginq_pat_ht_vec	INQUIRE PATTERN HEIGHT VECTOR
ginq_pat_ref_point	INQUIRE PATTERN REFERENCE POINT
ginq_pat_width_vec	INQUIRE PATTERN WIDTH VECTOR
ginq_pixel	INQUIRE PIXEL
ginq_pixel_array	INQUIRE PIXEL ARRAY
ginq_pixel_array_dims	INQUIRE PIXEL ARRAY DIMENSIONS

ginq_line_colr_ind	INQUIRE POLYLINE COLOUR INDEX
ginq_line_facs	INQUIRE POLYLINE FACILITIES
ginq_line_ind	INQUIRE POLYLINE INDEX
ginq_marker_colr_ind	INQUIRE POLYMARKER COLOUR INDEX
ginq_marker_facs	INQUIRE POLYMARKER FACILITIES
ginq_marker_ind	INQUIRE POLYMARKER INDEX
ginq_pred_colr_rep	INQUIRE PREDEFINED COLOUR REPRESENTATION
ginq_pred_fill_rep	INQUIRE PREDEFINED FILL AREA REPRESENTATION
ginq_pred_pat_rep	INQUIRE PREDEFINED PATTERN REPRESENTATION
ginq_pred_line_rep	INQUIRE PREDEFINED POLYLINE REPRESENTATION
ginq_pred_marker_rep	INQUIRE PREDEFINED POLYMARKER REPRESENTATION
ginq_pred_text_rep	INQUIRE PREDEFINED TEXT REPRESENTATION
ginq_set_active_wss	INQUIRE SET OF ACTIVE WORKSTATIONS
ginq_set_open_wss	INQUIRE SET OF OPEN WORKSTATIONS
ginq_text_align	INQUIRE TEXT ALIGNMENT
ginq_text_colr_ind	INQUIRE TEXT COLOUR INDEX
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_ws_cat	INQUIRE WORKSTATION CATEGORY
ginq_ws_class	INQUIRE WORKSTATION CLASSIFICATION
ginq_ws_conn_type	INQUIRE WORKSTATION CONNECTION AND TYPE
ginq_ws_defer_upd_sts	INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES
ginq_ws_max_nums	INQUIRE WORKSTATION MAXIMUM NUMBERS
ginq_ws_st	INQUIRE WORKSTATION STATE
ginq_ws_tran	INQUIRE WORKSTATION TRANSFORMATION
ginterpret_item	INTERPRET ITEM
gopen_gks	OPEN GKS
gopen_ws	OPEN WORKSTATION
gpolyline	POLYLINE
gpolymarker	POLYMARKER
gread_item	READ ITEM FROM GKSM
gsel_norm_tran	SELECT NORMALIZATION TRANSFORMATION
gset_asfs	SET ASPECT SOURCE FLAGS
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_clip_ind	SET CLIPPING INDICATOR
gset_colr_rep	SET COLOUR REPRESENTATION
gset_err_hand	SET ERROR HANDLER (GKS/C only)
gset_fill_colr_ind	SET FILL AREA COLOUR INDEX
gset_fill_ind	SET FILL AREA INDEX
gset_fill_int_style	SET FILL AREA INTERIOR STYLE
gset_fill_style_ind	SET FILL AREA STYLE INDEX
gset_linetype	SET LINETYPE
gset_linewidth	SET LINEWIDTH SCALE FACTOR
gset_marker_size	SET MARKER SIZE SCALE FACTOR
gset_marker_type	SET MARKER TYPE
gset_pat_ref_point	SET PATTERN REFERENCE POINT

gset_pat_size	SET PATTERN SIZE
gset_line_colr_ind	SET POLYLINE COLOUR INDEX
gset_line_ind	SET POLYLINE INDEX
gset_marker_colr_ind	SET POLYMARKER COLOUR INDEX
gset_marker_ind	SET POLYMARKER INDEX
gset_text_align	SET TEXT ALIGNMENT
gset_text_colr_ind	SET TEXT COLOUR INDEX
gset_text_font_prec	SET TEXT FONT AND PRECISION
gset_text_ind	SET TEXT INDEX
gset_text_path	SET TEXT PATH
gset_vp	SET VIEWPORT
gset_win	SET WINDOW
gset_ws_vp	SET WORKSTATION VIEWPORT
gset_ws_win	SET WORKSTATION WINDOW
gtext	TEXT
gupd_ws	UPDATE WORKSTATION
gwrite_item	WRITE ITEM TO GKSM

4.3.3.2 Level 0B

ginit_choice	INITIALISE CHOICE
ginit_loc	INITIALISE LOCATOR
ginit_string	INITIALISE STRING
ginit_stroke	INITIALISE STROKE
ginit_val	INITIALISE VALUATOR
ginq_choice_st	INQUIRE CHOICE DEVICE STATE
ginq_def_choice_data	INQUIRE DEFAULT CHOICE DEVICE DATA
ginq_def_loc_data	INQUIRE DEFAULT LOCATOR 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_loc_st	INQUIRE LOCATOR DEVICE STATE
ginq_num_avail_in	INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES
ginq_string_st	INQUIRE STRING DEVICE STATE
ginq_stroke_st	INQUIRE STROKE DEVICE STATE
ginq_val_st	INQUIRE VALUATOR DEVICE STATE
greq_choice	REQUEST CHOICE
greq_loc	REQUEST LOCATOR
greq_string	REQUEST STRING
greq_stroke	REQUEST STROKE
greq_val	REQUEST VALUATOR
gset_choice_mode	SET CHOICE MODE
gset_loc_mode	SET LOCATOR MODE
gset_string_mode	SET STRING MODE
gset_stroke_mode	SET STROKE MODE
gset_val_mode	SET VALUATOR MODE
gset_vp_in_pri	SET VIEWPORT INPUT PRIORITY

4.3.3.3 Level 0C

`gawait_event`	AWAIT EVENT
`gflush_events`	FLUSH DEVICE EVENTS
`gget_choice`	GET CHOICE
`gget_loc`	GET LOCATOR
`gget_string`	GET STRING
`gget_stroke`	GET STROKE
`gget_val`	GET VALUATOR
`ginq_in_overf`	INQUIRE INPUT QUEUE OVERFLOW
`ginq_more_simult_events`	INQUIRE MORE SIMULTANEOUS EVENTS
`gsample_choice`	SAMPLE CHOICE
`gsample_loc`	SAMPLE LOCATOR
`gsample_string`	SAMPLE STRING
`gsample_stroke`	SAMPLE STROKE
`gsample_val`	SAMPLE VALUATOR

4.3.3.4 Level 1A

`gaccum_tran_matrix`	ACCUMULATE TRANSFORMATION MATRIX
`gclose_seg`	CLOSE SEGMENT
`gcreate_seg`	CREATE SEGMENT
`gdel_seg`	DELETE SEGMENT
`gdel_seg_ws`	DELETE SEGMENT FROM WORKSTATION
`geval_tran_matrix`	EVALUATE TRANSFORMATION MATRIX
`ginq_def_defer_sts`	INQUIRE DEFAULT DEFERRAL STATE VALUES
`ginq_dyn_mod_seg_attrs`	INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES
`ginq_fill_rep`	INQUIRE FILL AREA REPRESENTATION
`ginq_list_fill_inds`	INQUIRE LIST OF FILL AREA INDICES
`ginq_list_pat_inds`	INQUIRE LIST OF PATTERN INDICES
`ginq_list_line_inds`	INQUIRE LIST OF POLYLINE INDICES
`ginq_list_marker_inds`	INQUIRE LIST OF POLYMARKER INDICES
`ginq_list_text_inds`	INQUIRE LIST OF TEXT INDICES
`ginq_name_open_seg`	INQUIRE NAME OF OPEN SEGMENT
`ginq_num_seg_pris`	INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED
`ginq_pat_rep`	INQUIRE PATTERN REPRESENTATION
`ginq_line_rep`	INQUIRE POLYLINE REPRESENTATION
`ginq_marker_rep`	INQUIRE POLYMARKER REPRESENTATION
`ginq_seg_attrs`	INQUIRE SEGMENT ATTRIBUTES
`ginq_set_assoc_wss`	INQUIRE SET OF ASSOCIATED WORKSTATIONS
`ginq_set_seg_names`	INQUIRE SET OF SEGMENT NAMES IN USE
`ginq_set_seg_names_ws`	INQUIRE SET OF SEGMENT NAMES ON WORKSTATION
`ginq_text_rep`	INQUIRE TEXT REPRESENTATION
`gmessage`	MESSAGE
`gredraw_all_segs_ws`	REDRAW ALL SEGMENTS ON WORKSTATION
`grename_seg`	RENAME SEGMENT
`gset_defer_st`	SET DEFERRAL STATE
`gset_fill_rep`	SET FILL AREA REPRESENTATION
`gset_highl`	SET HIGHLIGHTING
`gset_pat_rep`	SET PATTERN REPRESENTATION
`gset_line_rep`	SET POLYLINE REPRESENTATION
`gset_marker_rep`	SET POLYMARKER REPRESENTATION
`gset_seg_pri`	SET SEGMENT PRIORITY

gset_seg_tran	SET SEGMENT TRANSFORMATION
gset_text_rep	SET TEXT REPRESENTATION
gset_vis	SET VISIBILITY

4.3.3.5 Level 1B

ginit_pick	INITIALISE PICK
ginq_cur_pick_id	INQUIRE CURRENT PICK IDENTIFIER VALUE
ginq_def_pick_data	INQUIRE DEFAULT PICK DEVICE DATA
ginq_pick_st	INQUIRE PICK DEVICE STATE
greq_pick	REQUEST PICK
gset_det	SET DETECTABILITY
gset_pick_id	SET PICK IDENTIFIER
gset_pick_mode	SET PICK MODE

4.3.3.6 Level 1C

gget_pick	GET PICK
gsample_pick	SAMPLE PICK

4.3.3.7 Level 2A

gassoc_seg_ws	ASSOCIATE SEGMENT WITH WORKSTATION
gcopy_seg_ws	COPY SEGMENT FROM WORKSTATION
ginsert_seg	INSERT SEGMENT

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
I	integer	`Gint`
R	real	`Gfloat`
S	string	`char *`
P	point	`Gpoint`
L	list of points	`Gpoint_list`
2*R	vector	`Gvec, Gfloat_size`
N	name	`Gint, char *, void *`
E	enumeration type	`typedef enum`
CLR	colour representation	`Gcolr_rep`

5.2 Environmental Type Definitions

The data types defined in this section 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 environmental data type is the type `size_t`. Depending on the environment, `size_t` can be defined as `int`, `unsigned`, `unsigned long`, etc. `size_t` is defined in the `#include` file `<stddef.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, most implementation and registration dependent parts are of type `Gdata`. They can be replaced by any other structure.
- The data records of registered prompt and echo types are represented by structures with the names

pet_rn (n the prompt and echo type), whereas the data records of unregistered prompt and echo types are represented by structures with the names pet_un (-n the prompt and echo type).

- The data records of registered and unregistered GDPs, escape input and escape output are denoted in a similar way.
- For the input data records, the mandatory part, as specified in GKS (ISO 7942), 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;
- For illustrative reasons, Gitem_data is sketchily (and incompletely) defined such that the items described in ISO 7942, Annex E could be represented by GKS/C data types. This definition is, however, not mandatory. Anyway, Gitem_data shall be defined in such a way that items which correspond with unsupported GDP's can be read from GKSM and written to GKSM.

Gchoice_data choice data record

```
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_rep` colour representation

```
    typedef union {
        Grgb        rgb;       /* Red Green Blue colour specification */
    } Gcolr_rep;
```

REMARK. In ISO/IEC 8806-4, `Gcolr_rep` is implementation dependent.

`Gescape_in_data` escape input data record

```
    typedef union {
        struct Gescape_in_rl {
                        Gdata   reg_dep;    /* reg. dep. */
        } escape_rl;   /* escape 1 data */
        struct Gescape_in_ul {
                        Gdata   impl_dep;   /* impl. dep. */
        } escape_ul;   /* escape -1 data */
        /* etc. */
    } Gescape_in_data;
```

`Gescape_out_data` escape output data record

```
    typedef union {
        struct Gescape_out_rl {
                        Gdata   reg_dep;    /* reg. dep. */
        } escape_rl;   /* escape 1 data */
        struct Gescape_out_ul {
                        Gdata   impl_dep;   /* impl. dep. */
        } escape_ul;   /* escape -1 data */
        /* etc. */
    } Gescape_out_data;
```

`Ggdp_data` gdp data record

```
    typedef union {
        struct Ggdp_rl {
                    Gdata   reg_dep;    /* reg. dep. */
        } gdp_rl;   /* gdp 1 data */
        struct Ggdp_ul {
                    Gdata   impl_dep;   /* impl. dep. */
        } gdp_ul;   /* gdp -1 data */
        /* etc. */
    } Ggdp_data;
```

`Gitem_data` item data record

```
typedef struct {
    Gint                    type;         /* item type                          */
    Gint                    length;       /* item data record length            */
    union {
       Gctrl_flag           clear_ws;     /* control flag                     */
       Gupd_regen_flag      upd_ws;       /* regen. flag                      */
       struct Gdefer_st {
                            Gdefer_mode   defer_mode;
                                          /* deferral mode                    */
                            Girg_mode     irg_mode;
                                          /* irg mode                         */
       } defer_st;                        /* deferrral state                  */
       /* etc. */
       Gdata                gdp_unsupp;   /* GDPs not supported by impl.      */
       Gdata                impl_dep;     /* impl. dependent                  */
    } data;
} Gitem_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;
                    /* polyline             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;
                    /* polyline/fill area   control flag */
                    union Gloc_attrs {
                                            Gline_attrs     line_attrs;
                                            /* polyline     attrs. */
                                            Gfill_attrs     fill_attrs;
                                            /* fill area    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 record**

```
typedef struct {
    union Gpick_pets {
        struct Gpick_pet_r1 {
                    Gdata    impl_dep;    /* impl. dep. */
        } pet_r1;   /* pet 1 data */
        struct Gpick_pet_r2 {
                    Gdata    impl_dep;    /* impl. dep. */
        } pet_r2;   /* pet 2 data */
        struct Gpick_pet_r3 {
                    Gdata    impl_dep;    /* impl. dep. */
        } pet_r3;   /* pet 3 data */
        struct Gpick_pet_u1 {
                    Gdata    impl_dep;    /* impl. dep. */
        } 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 [editing] position */
    union string_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 record**

```
typedef struct {
    Gint      in_buf_size;      /* input buffer size (nr. of points) */
    Gint      init_pos;         /* initial [editing] position */
    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;
                                /* polymarker    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;
                                /* polyline      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

REMARKS:

1) For some two-field enumeration types, the order of the fields has been reversed with respect to ISO 7942; this reversal has been done in order to keep numerical compatibility with ISO 8651-1 (GKS/FORTRAN); the enumeration types are: `Gclip_ind`, `Gcolr_avail`, `Gdisp_surf_empty`, `Gecho_switch`, `Gupd_regen_flag`, `Gvis`;
2) For the enumeration type STATUS, the fields NO CHOICE and NO PICK have been compacted to the field NO INPUT, in order to keep numerical compatibility with GKS/FORTRAN.

Gasf aspect source flag

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

Gasfs aspect source flags

```
typedef struct {
    Gasf   linetype;         /* linetype ASF                       */
    Gasf   linewidth;        /* linewidth scale factor ASF         */
    Gasf   line_colr_ind;    /* polyline colour index ASF          */
    Gasf   marker_type;      /* marker type ASF                    */
    Gasf   marker_size;      /* marker size scale factor ASF       */
    Gasf   marker_colr_ind;  /* polymarker colour index 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 index ASF              */
    Gasf   fill_int_style;   /* fill area interior style ASF       */
    Gasf   fill_style_ind;   /* fill area style index ASF          */
    Gasf   fill_colr_ind;    /* fill area colour index ASF         */
} Gasfs;
```

Gattr_ctrl_flag attribute control flag

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

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;
```

Gclip_ind clipping indicator

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

Gcolr_avail colour available

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

Gcolr_facs colour facilities

```
    typedef struct {
        Gint          num_colrs;       /* num. of colours                  */
        Gcolr_avail   colr_avail;      /* colour availability              */
        Gint          num_pred_inds;   /* num. of predef. colour indices   */
    } Gcolr_facs;
```

Gcoord_switch coordinate switch

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

Gctrl_flag control flag

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

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;
```

Gdefer_mode deferral mode

```
    typedef enum {
        GDEFER_ASAP,
        GDEFER_BNIG,
        GDEFER_BNIL,
        GDEFER_ASTI
    } Gdefer_mode;
```

Gdet detectability

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

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;
```

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;
```

`Gecho_switch` **echo switch**

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

`Gfill_attrs` **fill area attributes**

```
typedef struct {
   Gasf           int_style_asf;   /* fill area interior style asf   */
   Gasf           style_ind_asf;   /* fill area style index asf      */
   Gasf           colr_ind_asf;    /* fill area colour index asf     */
   Gint           ind;             /* fill area index                */
   Gfill_bundle   bundle;          /* fill area bundle               */
} Gfill_attrs;
```

`Gfill_bundle` **fill area bundle**

```
typedef struct {
   Gfill_int_style  int_style;   /* fill area interior style   */
   Gint             style_ind;   /* fill area style index      */
   Gint             colr_ind;    /* fill area colour index     */
} Gfill_bundle;
```

`Gfill_facs` **fill area facilities**

```
typedef struct {
   Gint             num_int_styles;   /* num. of interior styles              */
   Gfill_int_style  int_styles[4];    /* list of available interior styles    */
   Gint_list        hatch_styles;     /* list of available hatch styles       */
   Gint             num_pred_inds;    /* num. of predef. fill area indices    */
} Gfill_facs;
```

`Gfill_int_style` **fill area interior style**

```
    typedef enum {
        GSTYLE_HOLLOW,
        GSTYLE_SOLID,
        GSTYLE_PAT,
        GSTYLE_HATCH
    } Gfill_int_style;
```

`Gfloat_size` **float size**

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

`Ghighl` **highlighting**

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

`Ghor_text_align` **horizontal text alignment**

```
    typedef enum {
        GHOR_NORM,
        GHOR_LEFT,
        GHOR_CTR,
        GHOR_RIGHT
    } Ghor_text_align;
```

`Gin_class` **input class**

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

`Gin_status` **[input] status**

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

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;
```

Ginq_type inquire type

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

Gint_list integer list

```
typedef struct {
    Gint   num_ints;   /* num. 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;
```

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;
```

● **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;
```

Gline_attrs **polyline attributes**

```
    typedef struct {
        Gasf            type_asf;       /* linetype asf                  */
        Gasf            width_asf;      /* linewidth asf                 */
        Gasf            colr_ind_asf;   /* polyline colour index asf     */
        Gint            ind;            /* polyline index                */
        Gline_bundle    bundle;         /* polyline bundle               */
    } Gline_attrs;
```

● **Gline_bundle** **polyline bundle**

```
    typedef struct {
        Gint     type;        /* linetype                  */
        Gfloat   width;       /* linewidth scale factor    */
        Gint     colr_ind;    /* polyline colour index     */
    } Gline_bundle;
```

Gline_facs **polyline facilities**

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

`Gline_fill_ctrl_flag` **polyline/fill area control flag**

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

`Gmarker_attrs` **polymarker attributes**

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

`Gmarker_bundle` **polymarker bundle**

```
typedef struct {
    Gint     type;       /* marker type               */
    Gfloat   size;       /* marker size scale factor  */
    Gint     colr_ind;   /* polymarker colour index   */
} Gmarker_bundle;
```

`Gmarker_facs` **polymarker facilities**

```
typedef struct {
    Gint_list   types;           /* list of marker types                */
    Gint        num_sizes;       /* num. 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;   /* num. of predef. polymarker indices  */
} Gmarker_facs;
```

`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;
```

`Gnum_in` **number [of] input [devices]**

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

`Gop_mode` **operating mode**

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

`Gop_st` **operating state**

```
typedef enum {
    GST_GKCL,
    GST_GKOP,
    GST_WSOP,
    GST_WSAC,
    GST_SGOP
} Gop_st;
```

`Gpat_rep` **pattern representation**

```
typedef struct {
    Gint_size   dims;         /* colour array's dimensions   */
    Gint        *colr_array;  /* colour array                */
} Gpat_rep;
```

Gpick pick [value]

```
typedef struct {
    Gint   seg_name;   /* segment name      */
    Gint   pick_id;    /* pick identifier   */
} Gpick;
```

Gpoint point

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

Gpoint_list point list

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

Gpr_flag prompt flag

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

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;
```

Grect rectangle

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

Grel_pri relative priority

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

● **Grgb** Red Green Blue [colour specification]

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

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;
```

● **Gstore** store

```
typedef void *Gstore;
```

Gtext_align text alignment

```
typedef struct {
    Ghor_text_align    hor;    /* horizontal component   */
    Gvert_text_align   vert;   /* vertical component     */
} 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            */
    Gint              colr_ind;         /* text colour index            */
} Gtext_bundle;
```

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

Gtext_font_prec text font [and] precision

```
typedef struct {
    Gint          font;   /* text 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;
```

Gtran transformation

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

Gtran_matrix transformation matrix

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

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;
```

Gvec vector

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

Gvert_text_align vertical text alignment

```
    typedef enum {
        GVERT_NORM,
        GVERT_TOP,
        GVERT_CAP,
        GVERT_HALF,
        GVERT_BASE,
        GVERT_BOTTOM
    } Gvert_text_align;
```

Gvis visibility

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

Gws_cat workstation category

```
    typedef enum {
        GCAT_OUT,
        GCAT_IN,
        GCAT_OUTIN,
        GCAT_WISS,
        GCAT_MO,
        GCAT_MI
    } Gws_cat;
```

`Gws_class` **workstation classification**

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

`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;
```

`Gws_st` **workstation state**

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

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 macros are listed, with their numerical values. These values are identical with the values of the function identifiers in ISO/IEC 8651-1 (GKS FORTRAN).

```
#define   Gfn_open_gks                    (0)
#define   Gfn_close_gks                   (1)
#define   Gfn_open_ws                     (2)
#define   Gfn_close_ws                    (3)
#define   Gfn_activate_ws                 (4)
#define   Gfn_deactivate_ws               (5)
#define   Gfn_clear_ws                    (6)
#define   Gfn_redraw_all_segs_ws          (7)
#define   Gfn_upd_ws                      (8)
#define   Gfn_set_defer_st                (9)
#define   Gfn_message                     (10)
#define   Gfn_escape                      (11)
#define   Gfn_polyline                    (12)
#define   Gfn_polymarker                  (13)
#define   Gfn_text                        (14)
#define   Gfn_fill_area                   (15)
#define   Gfn_cell_array                  (16)
#define   Gfn_gdp                         (17)
#define   Gfn_set_line_ind                (18)
#define   Gfn_set_linetype                (19)
#define   Gfn_set_linewidth               (20)
#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_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_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_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_pat_size                (39)
#define   Gfn_set_pat_ref_point           (40)
```

```
#define   Gfn_set_asfs                        (41)
#define   Gfn_set_pick_id                     (42)
#define   Gfn_set_line_rep                    (43)
#define   Gfn_set_marker_rep                  (44)
#define   Gfn_set_text_rep                    (45)
#define   Gfn_set_fill_rep                    (46)
#define   Gfn_set_pat_rep                     (47)
#define   Gfn_set_colr_rep                    (48)
#define   Gfn_set_win                         (49)
#define   Gfn_set_vp                          (50)
#define   Gfn_set_vp_in_pri                   (51)
#define   Gfn_sel_norm_tran                   (52)
#define   Gfn_set_clip_ind                    (53)
#define   Gfn_set_ws_win                      (54)
#define   Gfn_set_ws_vp                       (55)
#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_tran                    (64)
#define   Gfn_set_vis                         (65)
#define   Gfn_set_highl                       (66)
#define   Gfn_set_seg_pri                     (67)
#define   Gfn_set_det                         (68)
#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_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_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_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_event                  (93)
#define   Gfn_flush_events                 (94)
#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_write_item                   (101)
#define   Gfn_get_item_type                (102)
#define   Gfn_read_item                    (103)
#define   Gfn_interpret_item               (104)
#define   Gfn_eval_tran_matrix             (105)
#define   Gfn_accum_tran_matrix            (106)
#define   Gfn_emergency_close_gks          (153)
#define   Gfn_err_hand                     (154)
#define   Gfn_err_log                      (155)
#define   Gfn_set_err_hand                 (156)
```

6.2 Error Codes

The error codes are represented by macros. The range of the numeric values of the macros is between 0 and 1000.

Below, the error macros are listed.

```
#define  GE_NO_ERR                (0)  /* No Errors */

/* operating state errors */

#define  GE_NOT_GKCL              (1)  /* GKS not in proper state: GKS shall be
                                       in the state GKCL */
#define  GE_NOT_GKOP              (2)  /* GKS not in proper state: GKS shall be
                                       in the state GKOP */
#define  GE_NOT_WSAC              (3)  /* GKS not in proper state: GKS shall be
                                       in the state WSAC */
#define  GE_NOT_SGOP              (4)  /* GKS not in proper state: GKS shall be
                                       in the state SGOP */
#define  GE_NOT_WSAC_SGOP         (5)  /* GKS not in proper state: GKS shall be
                                       either in the state WSAC or in the state
                                       SGOP */
#define  GE_NOT_WSOP_WSAC         (6)  /* GKS not in proper state: GKS shall be
                                       either in the state WSOP or in the state
                                       WSAC */
#define  GE_GKCL_GKOP             (7)  /* GKS not in proper state: GKS shall be
                                       in one of the states WSOP, WSAC or SGOP
                                       */
#define  GE_GKCL                  (8)  /* GKS not in proper state: GKS shall be
                                       in one of the states GKOP, WSOP, WSAC or
                                       SGOP */

/* workstation errors */
```

```
#define  GE_WS_ID_INVAL           (20)  /* Specified workstation identifier is
                                        invalid */
#define  GE_CONN_ID_INVAL         (21)  /* Specified connection identifier is
                                        invalid */
#define  GE_WS_TYPE_INVAL         (22)  /* Specified workstation type is invalid
                                        */
#define  GE_NO_WS_TYPE            (23)  /* Specified workstation type does not
                                        exist */
#define  GE_WS_OPEN               (24)  /* Specified workstation is open */
#define  GE_WS_NOT_OPEN           (25)  /* Specified workstation is not open */
#define  GE_WS_CANT_OPEN          (26)  /* Specified workstation cannot be opened
                                        */
#define  GE_WISS_NOT_OPEN         (27)  /* Workstation Independent Segment
                                        Storage is not open */
#define  GE_WISS_OPEN             (28)  /* Workstation Independent Segment
                                        Storage is already open */
#define  GE_WS_ACTIVE             (29)  /* Specified workstation is active */
#define  GE_WS_INACTIVE           (30)  /* Specified workstation is not active */
#define  GE_WS_MO                 (31)  /* Specified workstation is of category
                                        MO */
#define  GE_WS_NOT_MO             (32)  /* Specified workstation is not of
                                        category MO */
#define  GE_WS_MI                 (33)  /* Specified workstation is of category
                                        MI */
#define  GE_WS_NOT_MI             (34)  /* Specified workstation is not of
                                        category MI */
#define  GE_WS_IN                 (35)  /* Specified workstation is of category
                                        INPUT */
#define  GE_WS_WISS               (36)  /* Specified workstation is Workstation
                                        Independent Segment Storage */
#define  GE_WS_NOT_OUTIN          (37)  /* Specified workstation is not of
                                        category OUTIN */
#define  GE_WS_NOT_IN_OUTIN       (38)  /* Specified workstation is neither of
                                        category INPUT nor of category OUTIN */
#define  GE_WS_NOT_OUT_OUTIN      (39)  /* Specified workstation is neither of
                                        category OUTPUT nor of category OUTIN */
#define  GE_WS_NO_PIXEL           (40)  /* Specified workstation has no pixel
                                        store readback capability */
#define  GE_WS_TYPE_NO_GDP        (41)  /* Specified workstation type is not able
                                        to generate the specified generalized
                                        drawing primitive */
#define  GE_WS_MAX_OPEN           (42)  /* Maximum number of simultaneously open
                                        workstations would be exceeded */
#define  GE_WS_MAX_ACTIVE         (43)  /* Maximum number of simultaneously
                                        active workstations would be exceeded */

/* transformation errors */

#define  GE_TRAN_NUM_INVAL        (50)  /* Transformation number is invalid */
#define  GE_RECT_INVAL            (51)  /* Rectangle definition is invalid */
#define  GE_VP_INVAL              (52)  /* Viewport is not within the Normalized
                                        Device Coordinate unit square */
```

```
#define  GE_WIN_INVAL            (53)  /* Workstation window is not within the
                                       Normalized Device Coordinate unit square
                                       */
#define  GE_WS_VP_INVAL          (54)  /* Workstation viewport is not within the
                                       display space */

/* output attribute errors */

#define  GE_LINE_IND_INVAL       (60)  /* Polyline index is invalid */
#define  GE_LINE_REP_UNDEF       (61)  /*  A  representation  for  the  specified
                                       polyline  index  has  not  been  defined  on
                                       this workstation */
#define  GE_LINE_REP_NOT_PRED    (62)  /*  A  representation  for  the  specified
                                       polyline index has not been predefined on
                                       this workstation */
#define  GE_LINETYPE_ZERO        (63)  /* Linetype is equal to zero */
#define  GE_LINETYPE_NOT_WS      (64)  /* Specified linetype is not supported on
                                       this workstation */
#define  GE_LINEWIDTH_LT_ZERO    (65)  /*  Linewidth  scale  factor  is  less  than
                                       zero */
#define  GE_MARKER_IND_INVAL     (66)  /* Polymarker index is invalid */
#define  GE_MARKER_REP_UNDEF     (67)  /*  A  representation  for  the  specified
                                       polymarker index has not been defined on
                                       this workstation */
#define  GE_MARKER_REP_NOT_PRED  (68)  /*  A  representation  for  the  specified
                                       polymarker index has not been predefined
                                       on this workstation */
#define  GE_MARKER_TYPE_ZERO     (69)  /* Marker type is equal to zero */
#define  GE_MARKER_TYPE_NOT_WS   (70)  /* Specified marker type is not supported
                                       on this workstation */
#define  GE_MARKER_SIZE_LT_ZERO  (71)  /* Marker size scale factor is less than
                                       zero */
#define  GE_TEXT_IND_INVAL       (72)  /* Text index is invalid */
#define  GE_TEXT_REP_UNDEF       (73)  /*  A  representation  for  the  specified
                                       text  index  has  not  been  defined  on  this
                                       workstation */
#define  GE_TEXT_REP_NOT_PRED    (74)  /*  A  representation  for  the  specified
                                       text  index  has  not  been  predefined  on
                                       this workstation */
#define  GE_FONT_ZERO            (75)  /* Text font is equal to zero */
#define  GE_FONT_NOT_WS          (76)  /*  Requested  text  font  is  not  supported
                                       for  the  specified  precision  on  this
                                       workstation */
#define  GE_EXPAN_LE_ZERO        (77)  /*  Character  expansion  factor  is  less
                                       than or equal to zero */
#define  GE_HT_LE_ZERO           (78)  /* Character height is less than or equal
                                       to zero */
#define  GE_UP_VEC_ZERO          (79)  /* Length of character up vector is zero
                                       */
#define  GE_FILL_IND_INVAL       (80)  /* Fill area index is invalid */
#define  GE_FILL_REP_UNDEF       (81)  /*  A  representation  for  the  specified
                                       fill area index has not been defined on
```

```
                                         this workstation */
#define  GE_FILL_REP_NOT_PRED       (82) /*  A  representation  for  the  specified
                                         fill  area  index  has  not  been  predefined
                                         on this workstation */
#define  GE_INT_STYLE_NOT_WS        (83) /* Specified fill area interior style is
                                         not supported on this workstation */
#define  GE_STYLE_IND_ZERO          (84) /*  Style  (pattern  or  hatch)  index  is
                                         equal to zero */
#define  GE_PAT_IND_INVAL           (85) /* Specified pattern index is invalid */
#define  GE_HATCH_STYLE_NOT_WS      (86) /* Specified hatch style is not supported
                                         on this workstation */
#define  GE_PAT_SIZE_LE_ZERO        (87) /* Pattern size value is not positive */
#define  GE_PAT_REP_UNDEF           (88) /*  A  representation  for  the  specified
                                         pattern  index  has  not  been  defined  on
                                         this workstation */
#define  GE_PAT_REP_NOT_PRED        (89) /*  A  representation  for  the  specified
                                         pattern index has not been predefined on
                                         this workstation */
#define  GE_PAT_NOT_WS              (90) /*  Interior  style  PATTERN  is  not  sup-
                                         ported on this workstation */
#define  GE_DIM_INVAL               (91) /* Dimensions of colour array are invalid
                                         */
#define  GE_COLR_IND_LT_ZERO        (92) /* Colour index is less than zero */
#define  GE_COLR_IND_INVAL          (93) /* Colour index is invalid */
#define  GE_COLR_REP_UNDEF          (94) /*  A  representation  for  the  specified
                                         colour index has not been defined on this
                                         workstation */
#define  GE_COLR_REP_NOT_PRED       (95) /*  A  representation  for  the  specified
                                         colour  index  has  not  been  predefined  on
                                         this workstation */
#define  GE_COLR_INVAL              (96) /*  Colour  component  is  outside  valid
                                         range for colour model */
#define  GE_PICK_ID_INVAL           (97) /* Pick identifier is invalid */

/* output errors */

#define  GE_NUM_POINT_INVAL        (100) /* Number of points is invalid */
#define  GE_INVAL_CODE             (101) /* Invalid code in string */
#define  GE_GDP_ID_INVAL           (102) /* Generalized drawing primitive identif-
                                         ier is invalid */
#define  GE_GDP_DATA_INVAL         (103) /* Content of generalized drawing primi-
                                         tive data record is invalid */
#define  GE_CANT_GEN_GDP           (104) /* At least one active workstation is not
                                         able to generate the specified general-
                                         ized drawing primitive */
#define  GE_CANT_GEN_GDP_CLIP      (105) /* At least one active workstation is not
                                         able to generate the specified general-
                                         ized drawing primitive under the current
                                         transformations and clipping conditions
                                         */

/* segment errors */
```

```
#define  GE_SEG_NAME_INVAL          (120) /* Specified segment name is invalid */
#define  GE_SEG_NAME_USED           (121) /* Specified segment name is already in
                                          use */
#define  GE_SEG_ABSENT              (122) /* Specified segment does not exist */
#define  GE_SEG_NOT_WS              (123) /* Specified segment does not exist on
                                          specified workstation */
#define  GE_SEG_NOT_WISS            (124) /* Specified segment does not exist on
                                          Workstation Independent Segment Storage
                                          */
#define  GE_SEG_OPEN                (125) /* Specified segment is open */
#define  GE_SEG_PRI_INVAL           (126) /* Segment priority is outside the range
                                          [0,1] */

/* input errors */

#define  GE_IN_DEV_NOT_WS           (140) /* Specified input device is not present
                                          on workstation */
#define  GE_IN_DEV_NOT_REQ          (141) /* Input device is not in REQUEST mode */
#define  GE_IN_DEV_NOT_SAMPLE       (142) /* Input device is not in SAMPLE mode */
#define  GE_EV_SAMPLE_UNAVAIL       (143) /* EVENT and SAMPLE input mode are not
                                          available at this level of GKS */
#define  GE_PET_NOT_WS              (144) /* Specified prompt and echo type is not
                                          supported on this workstation */
#define  GE_ECHO_INVAL              (145) /* Echo area is outside display space */
#define  GE_IN_DATA_INVAL           (146) /* Contents of input data record are
                                          invalid */
#define  GE_QUE_OVERF               (147) /* Input queue has overflowed */
#define  GE_NO_QUE_OVERF            (148) /* Input queue has not overflowed since
                                          GKS was opened or the last invocation of
                                          INQUIRE INPUT QUEUE OVERFLOW */
#define  GE_ASSOC_WS_CLOSED         (149) /* Input queue has overflowed, but asso-
                                          ciated workstation has been closed */
#define  GE_NO_CUR_EV               (150) /* No input value of the correct class is
                                          in the current event report */
#define  GE_TIMEOUT_INVAL           (151) /* Timeout is invalid */
#define  GE_INIT_INVAL              (152) /* Initial value is invalid */
#define  GE_INIT_STROKE_INVAL       (153) /* Number of points in the initial stroke
                                          is greater than the buffer size */
#define  GE_INIT_STRING_INVAL       (154) /* Length of the initial string is
                                          greater than the buffer size */

/* GKSM errors */

#define  GE_ITEM_RESERVED           (160) /* Item type is not allowed for user
                                          items */
#define  GE_ITEM_LENGTH_INVAL       (161) /* Item length is invalid */
#define  GE_NO_ITEM_MI              (162) /* No item is left in GKS Metafile input
                                          */
#define  GE_ITEM_INVAL              (163) /* Metafile item is invalid */
#define  GE_ITEM_GKS_INVAL          (164) /* Item type is not a valid GKS item */
#define  GE_ITEM_DATA_INVAL         (165) /* Content of item data record is invalid
                                          for the specified item type */
```

```
#define  GE_MAX_ITEM_DATA_INVAL       (166) /* Maximum item data record length is
                                            invalid */
#define  GE_USER_ITEM                 (167) /* User item cannot be interpreted */
#define  GE_FUNC_UNAVAIL              (168) /* Specified function is not supported in
                                            this level of GKS */

/* escape errors */

#define  GE_ESCAPE_FUNC_UNAVAIL       (180) /* Specified escape function is not sup-
                                            ported */
#define  GE_ESCAPE_ID_INVAL           (181) /* Specified escape function identifica-
                                            tion is invalid */
#define  GE_ESCAPE_DATA_INVAL         (182) /* Contents of escape data record are
                                            invalid */

/* error file errors */

#define  GE_ERR_FILE_INVAL            (200) /* Specified error file is invalid */

/* I/O errors */

#define  GE_MEM_OVERF                 (300) /* Storage overflow has occurred in GKS
                                            */
#define  GE_SEG_MEM_OVERF             (301) /* Storage overflow has occurred in seg-
                                            ment storage */
#define  GE_IO_ERR_READ               (302) /* Input/Output error has occurred while
                                            reading */
#define  GE_IO_ERR_WRITE              (303) /* Input/Output error has occurred while
                                            writing */
#define  GE_IO_ERR_WRITE_WS           (304) /* Input/Output error has occurred while
                                            sending data to a workstation */
#define  GE_IO_ERR_READ_WS            (305) /* Input/Output error has occurred while
                                            receiving data from a workstation */
#define  GE_IO_ERR_LIB                (306) /* Input/Output error has occurred during
                                            program library management */
#define  GE_IO_ERR_WS_TABLE           (307) /* Input/Output error has occurred while
                                            reading workstation description table */
#define  GE_ARITH_ERR                 (308) /* Arithmetic error has occurred */

/* binding specific errors */

#define  GE_START_IND_INVAL           (2200)/* Start index out of range*/
#define  GE_APPL_LIST_LENGTH_LT_ZERO  (2201)/* Length of application list is nega-
                                            tive */
#define  GE_ENUM_TYPE_INVAL           (2202)/* Enumeration type out of range*/
#define  GE_ALLOC_STORE               (2203)/* Error while allocating Store */
#define  GE_ALLOC_MEM_STORE           (2204)/* Error while allocating memory for
                                            Store */
```

6.3 Miscellaneous Macros

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

6.3.4 Default Parameters of OPEN GKS

```
#define   GDEF_MEM_SIZE        ((size_t) (-1))/* Default memory size */
#define   GDEF_ERR_FILE        ((char *) (""))/* 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 **LEVEL**

```
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.

"LEVEL" is the GKS level 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 return `void`.

7.2 Control Functions

OPEN GKS **0A**

```
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. If `err_file` has the value `GDEF_ERR_FILE`, the `stderr` file is used. If `mem_units` has the value `GDEF_MEM_SIZE`, the number of available bytes is implementation dependent.

CLOSE GKS **0A**

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

OPEN WORKSTATION 0A

```
void gopen_ws(
     Gint         ws_id,      /* workstation identifier   */
     const void   *conn_id,   /* connection identifier    */
     Gint         ws_type     /* workstation type         */
);
```

CLOSE WORKSTATION 0A

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

ACTIVATE WORKSTATION 0A

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

DEACTIVATE WORKSTATION 0A

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

CLEAR WORKSTATION 0A

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

REDRAW ALL SEGMENTS ON WORKSTATION 1A

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

UPDATE WORKSTATION 0A

```
void gupd_ws(
     Gint              ws_id,            /* workstation identifier      */
     Gupd_regen_flag   upd_regen_flag    /* update regeneration flag    */
);
```

SET DEFERRAL STATE 1A

```
void gset_defer_st(
     Gint          ws_id,         /* workstation identifier      */
     Gdefer_mode   defer_mode,    /* deferral mode               */
     Girg_mode     irg_mode       /* implicit regeneration mode  */
);
```

MESSAGE 1A

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

ESCAPE 0A

```
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 `store`.

7.3 Output Functions

POLYLINE 0A

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

POLYMARKER 0A

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

TEXT 0A

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

FILL AREA **0A**

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

CELL ARRAY **0A**

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

GENERALIZED DRAWING PRIMITIVE **0A**

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

7.4 Output Attribute Functions

7.4.1 Workstation Independent Primitive Attributes

SET POLYLINE INDEX **0A**

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

SET LINETYPE **0A**

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

SET LINEWIDTH SCALE FACTOR **0A**

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

SET POLYLINE COLOUR INDEX **0A**

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

SET POLYMARKER INDEX 0A

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

SET MARKER TYPE 0A

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

SET MARKER SIZE SCALE FACTOR 0A

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

SET POLYMARKER COLOUR INDEX 0A

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

SET TEXT INDEX 0A

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

SET TEXT FONT AND PRECISION 0A

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

SET CHARACTER EXPANSION FACTOR 0A

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

SET CHARACTER SPACING 0A

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

SET TEXT COLOUR INDEX 0A

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

SET CHARACTER HEIGHT 0A

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

SET CHARACTER UP VECTOR 0A

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

SET TEXT PATH 0A

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

SET TEXT ALIGNMENT 0A

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

SET FILL AREA INDEX 0A

```
void gset_fill_ind(
     Gint   fill_ind   /* fill area index   */
);
```

SET FILL AREA INTERIOR STYLE 0A

```
void gset_fill_int_style(
     Gfill_int_style   fill_int_style   /* fill area interior style   */
);
```

SET FILL AREA STYLE INDEX 0A

```
void gset_fill_style_ind(
     Gint   fill_style_ind   /* fill area style index   */
);
```

SET FILL AREA COLOUR INDEX 0A

```
void gset_fill_colr_ind(
     Gint   fill_colr_ind    /* fill area colour index   */
);
```

SET PATTERN SIZE 0A

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

SET PATTERN REFERENCE POINT 0A

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

SET ASPECT SOURCE FLAGS 0A

```
void gset_asfs(
     const Gasfs   *list_asf   /* list of aspect source flags    */
);
```

SET PICK IDENTIFIER 1B

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

7.4.2 Workstation Attributes

SET POLYLINE REPRESENTATION 1A

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

SET POLYMARKER REPRESENTATION 1A

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

SET TEXT REPRESENTATION 1A

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

SET FILL AREA REPRESENTATION 1A

```
void gset_fill_rep(
     Gint                  ws_id,          /* workstation identifier      */
     Gint                  fill_ind,       /* fill area index             */
     const Gfill_bundle    *fill_bundle    /* fill area representation    */
);
```

SET PATTERN REPRESENTATION 1A

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

SET COLOUR REPRESENTATION 0A

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

7.5 Transformation Functions

7.5.1 Normalization Transformation

SET WINDOW 0A

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

SET VIEWPORT 0A

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

SET VIEWPORT INPUT PRIORITY 0B

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

SELECT NORMALIZATION TRANSFORMATION 0A

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

SET CLIPPING INDICATOR 0A

```
void gset_clip_ind(
     Gclip_ind   clip_ind   /* clipping indicator   */
);
```

7.5.2 Workstation transformation

SET WORKSTATION WINDOW 0A

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

SET WORKSTATION VIEWPORT 0A

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

7.6 Segment Functions

7.6.1 Segment Manipulation Functions

CREATE SEGMENT 1A

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

CLOSE SEGMENT 1A

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

RENAME SEGMENT 1A

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

DELETE SEGMENT 1A

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

DELETE SEGMENT FROM WORKSTATION 1A

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

ASSOCIATE SEGMENT WITH WORKSTATION 2A

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

COPY SEGMENT TO WORKSTATION 2A

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

INSERT SEGMENT 2A

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

7.6.2 Segment Attribute Functions

SET SEGMENT TRANSFORMATION **1A**

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

SET VISIBILITY **1A**

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

SET HIGHLIGHTING **1A**

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

SET SEGMENT PRIORITY **1A**

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

SET DETECTABILITY **1B**

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

7.7 Input Functions

7.7.1 Initialization of Input Devices Functions

INITIALISE LOCATOR 0B

```
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        */
);
```

INITIALISE STROKE 0B

```
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        */
);
```

INITIALISE VALUATOR 0B

```
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     */
);
```

INITIALISE CHOICE 0B

```
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       */
);
```

INITIALISE PICK 1B

```
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         */
);
```

INITIALISE STRING 0B

```
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       */
);
```

7.7.2 Setting the Mode of Input Devices Functions

SET LOCATOR MODE 0B

```
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 STROKE MODE 0B

```
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 VALUATOR MODE 0B

```
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 CHOICE MODE 0B

```
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 PICK MODE 1B

```
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 STRING MODE 0B

```
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               */
);
```

7.7.3 Request Input Functions

REQUEST LOCATOR 0B

```
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 STROKE 0B

```
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 VALUATOR 0B

```
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 CHOICE 0B

```
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 PICK 1B

```
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  */
);
```

REQUEST STRING **0B**

```
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`.

● **7.7.4 Sample Input Functions**

SAMPLE LOCATOR **0C**

```
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 STROKE **0C**

```
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 VALUATOR **0C**

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

SAMPLE CHOICE 0C

```
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 PICK 1C

```
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           */
);
```

SAMPLE STRING 0C

```
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`.

7.7.5 Event Input Functions

AWAIT EVENT 0C

```
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    */
);
```

FLUSH DEVICE EVENTS 0C

```
void gflush_events(
     Gint       ws_id,     /* workstation identifier        */
     Gin_class  class,     /* device class                  */
     Gint       in_num     /* logical input device number   */
);
```

GET LOCATOR 0C

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

GET STROKE 0C

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

GET VALUATOR 0C

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

GET CHOICE 0C

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

GET PICK 1C

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

GET STRING 0C

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

7.8 Metafile Functions

WRITE ITEM TO GKSM **0A**

```
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          */
);
```

GET ITEM TYPE FROM GKSM **0A**

```
void gget_item_type(
     Gint    ws_id,               /* workstation identifier        */
     Gint    *item_type,          /* OUT item type                 */
     Gint    *item_data_length    /* OUT item data record length   */
);
```

READ ITEM FROM GKSM **0A**

```
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           */
);
```

INTERPRET ITEM **0A**

```
void ginterpret_item(
     Gint                type,                /* item type                 */
     Gint                item_data_length,    /* item data record length   */
     const Gitem_data    *item_data           /* item data record          */
);
```

7.9 Inquiry Functions

7.9.1 Inquiry Functions for Operating State Value

INQUIRE OPERATING STATE VALUE **0A**

```
void ginq_op_st(
     Gop_st    *op_st    /* OUT operating state value    */
);
```

7.9.2 Inquiry Functions for GKS Description Table

INQUIRE LEVEL OF GKS 0A

```
void ginq_level_gks(
     Gint      *err_ind,     /* OUT error indicator   */
     Glevel    *level        /* OUT level of GKS      */
);
```

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES 0A

```
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_type,              /* OUT list of avalailable ws types     */
     Gint         *num_elems_impl_list   /* OUT length of impl. list             */
);
```

INQUIRE WORKSTATION MAXIMUM NUMBERS 0A

```
void ginq_ws_max_nums(
     Gint             *err_ind,      /* OUT error indicator                 */
     Gws_max_nums     *ws_max_num    /* OUT workstation maximum numbers     */
);
```

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER 0A

```
void ginq_max_norm_tran_num(
     Gint     *err_ind,            /* OUT error indicator             */
     Gint     *max_norm_tran_num   /* OUT maximum normalization
                                          transformation number       */
);
```

7.9.3 Inquiry Functions for GKS State List

INQUIRE SET OF OPEN WORKSTATIONS 0A

```
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 SET OF ACTIVE WORKSTATIONS 1A

```
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 CURRENT PRIMITIVE ATTRIBUTE VALUES 0A

```
void ginq_cur_prim_attrs(
     Gint          *err_ind,      /* OUT error indicator        */
     Gprim_attrs   *prim_attrs    /* OUT current primitive
                                     attribute structure        */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES) INQUIRE POLYLINE INDEX 0A

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES) INQUIRE POLYMARKER INDEX 0A

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES) INQUIRE TEXT INDEX 0A

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES) INQUIRE CHARACTER HEIGHT 0A

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE CHARACTER UP VECTOR **0A**

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE CHARACTER WIDTH **0A**

```
void ginq_char_width(
     Gint      *err_ind,      /* OUT error indicator           */
     Gfloat    *char_width    /* OUT current character width   */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE CHARACTER BASE VECTOR **0A**

```
void ginq_char_base_vec(
     Gint    *err_ind,        /* OUT error indicator                  */
     Gvec    *char_base_vec   /* OUT current character base vector    */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE TEXT PATH **0A**

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE TEXT ALIGNMENT **0A**

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

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE FILL AREA INDEX **0A**

```
void ginq_fill_ind(
     Gint    *err_ind,    /* OUT error indicator          */
     Gint    *fill_ind    /* OUT current fill area index  */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE PATTERN WIDTH VECTOR **0A**

```
void ginq_pat_width_vec(
     Gint   *err_ind,        /* OUT error indicator               */
     Gvec   *pat_width_vec   /* OUT current pattern width vector  */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE PATTERN HEIGHT VECTOR **0A**

```
void ginq_pat_ht_vec(
     Gint   *err_ind,     /* OUT error indicator                 */
     Gvec   *pat_ht_vec   /* OUT current pattern height vector   */
);
```

(INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES)
INQUIRE PATTERN REFERENCE POINT **0A**

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

INQUIRE CURRENT PICK IDENTIFIER VALUE **1B**

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

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES **0A**

```
void ginq_cur_indiv_attrs(
     Gint          *err_ind,      /* OUT error indicator      */
     Gindiv_attrs  *indiv_attr    /* OUT current individual
                                     attribute  structure     */
);
```

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE LINETYPE **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE LINEWIDTH SCALE FACTOR **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE POLYLINE COLOUR INDEX **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE MARKER TYPE **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE MARKER SIZE SCALE FACTOR **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE POLYMARKER COLOUR INDEX **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE TEXT FONT AND PRECISION **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE CHARACTER EXPANSION FACTOR **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE CHARACTER SPACING **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE TEXT COLOUR INDEX **0A**

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

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE FILL AREA INTERIOR STYLE **0A**

```
void ginq_fill_int_style(
    Gint             *err_ind,         /* OUT error indicator     */
    Gfill_int_style  *fill_int_style   /* OUT current fill area
                                          interior style          */
);
```

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE FILL AREA STYLE INDEX **0A**

```
void ginq_fill_style_ind(
    Gint   *err_ind,          /* OUT error indicator               */
    Gint   *fill_style_ind    /* OUT current fill area style index */
);
```

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES)
INQUIRE FILL AREA COLOUR INDEX **0A**

```
void ginq_fill_colr_ind(
    Gint   *err_ind,         /* OUT error indicator                */
    Gint   *fill_colr_ind    /* OUT current fill area colour index */
);
```

(INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES) INQUIRE LIST OF ASPECT SOURCE FLAGS 0A

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

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER 0A

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

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS 0A

```
void ginq_list_norm_tran_nums(
     Gint         num_elems_appl_list,    /* length of application list   */
     Gint         start_ind,              /* starting index               */
     Gint         *err_ind,               /* OUT error indicator          */
     Gint_list    *norm_tran_num,         /* OUT list of normalization
                                             transformation numbers      */
     Gint         *num_elems_impl_list    /* OUT length of impl. list     */
);
```

INQUIRE NORMALIZATION TRANSFORMATION 0A

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

INQUIRE CLIPPING 0A

```
void ginq_clip(
     Gint    *err_ind,          /* OUT error indicator                */
     Gclip   *clip_ind_rect     /* OUT current clipping indicator
                                   and rectangle                      */
);
```

INQUIRE NAME OF OPEN SEGMENT 1A

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

INQUIRE SET OF SEGMENT NAMES IN USE 1A

```
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 MORE SIMULTANEOUS EVENTS 0C

```
void ginq_more_simult_events(
     Gint                  *err_ind,         /* OUT error indicator                */
     Gmore_simult_events   *simult_events    /* OUT [more] simultaneous events     */
);
```

7.9.4 Inquiry Functions for Workstation State List

INQUIRE WORKSTATION CONNECTION AND TYPE 0A

```
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 `store`.

INQUIRE WORKSTATION STATE 0A

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

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES 0A

```
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 implicit regeneration mode      */
     Gdisp_surf_empty      *disp_surf_empty,   /* OUT display surface empty           */
     Gnew_frame_nec_upd    *new_frame          /* OUT new frame action
                                                  necessary at update                 */
);
```

INQUIRE LIST OF POLYLINE INDICES 1A

```
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 polyline indices */
     Gint        *num_elems_impl_list /* OUT length of impl. list            */
);
```

INQUIRE POLYLINE REPRESENTATION 1A

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

INQUIRE LIST OF POLYMARKER INDICES 1A

```
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 polymarker indices */
     Gint        *num_elems_impl_list /* OUT length of impl. list              */
);
```

INQUIRE POLYMARKER REPRESENTATION 1A

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

INQUIRE LIST OF TEXT INDICES 1A

```
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 TEXT REPRESENTATION 1A

```
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 TEXT EXTENT 0A

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

INQUIRE LIST OF FILL AREA INDICES 1A

```
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                */
);
```

INQUIRE FILL AREA REPRESENTATION 1A

```
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   */
);
```

INQUIRE LIST OF PATTERN INDICES 1A

```
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 PATTERN REPRESENTATION 1A

```
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 `store`.

INQUIRE LIST OF COLOUR INDICES 0A

```
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 COLOUR REPRESENTATION 0A

```
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 TRANSFORMATION 0A

```
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    */
);
```

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION 1A

```
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 LOCATOR DEVICE STATE 0B

```
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 `store`.

INQUIRE STROKE DEVICE STATE 0B

```
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 VALUATOR DEVICE STATE 0B

```
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 `store`.

INQUIRE CHOICE DEVICE STATE 0B

```
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 `store`.

INQUIRE PICK DEVICE STATE **1B**

```
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 `store`.

INQUIRE STRING DEVICE STATE **0B**

```
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`.

7.9.5 Inquiry Functions for Workstation Description Table

INQUIRE WORKSTATION CATEGORY **0A**

```
void ginq_ws_cat(
     Gint      ws_type,     /* workstation type          */
     Gint      *err_ind,    /* OUT error indicator       */
     Gws_cat   *cat         /* OUT workstation category  */
);
```

INQUIRE WORKSTATION CLASSIFICATION 0A

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

INQUIRE DISPLAY SPACE SIZE 0A

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

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES 1A

```
void ginq_dyn_mod_ws_attrs(
     Gint                ws_type,     /* workstation type                 */
     Gint                *err_ind,    /* OUT error indicator              */
     Gdyn_mod_ws_attrs   *dyn_mod     /* OUT dynamic modification of
                                         workstation attributes           */
);
```

INQUIRE DEFAULT DEFERRAL STATE VALUES 1A

```
void ginq_def_defer_sts(
     Gint           ws_type,       /* workstation type              */
     Gint           *err_ind,      /* OUT error indicator           */
     Gdefer_mode    *defer_mode,   /* OUT default deferral mode     */
     Girg_mode      *irg_mode      /* OUT default impl. reg. mode   */
);
```

INQUIRE POLYLINE FACILITIES 0A

```
void ginq_line_facs(
     Gint         ws_type,               /* workstation 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 polyline facilities       */
     Gint         *num_elems_impl_list   /* OUT length of linetype
                                            list in impl.                 */
);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION 0A

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

INQUIRE POLYMARKER FACILITIES 0A

```
void ginq_marker_facs(
     Gint          ws_type,              /* workstation 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 polymarker facilities  */
     Gint          *num_elems_impl_list  /* OUT length of marker type
                                            list in impl.              */
);
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION 0A

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

INQUIRE TEXT FACILITIES 0A

```
void ginq_text_facs(
     Gint        ws_type,              /* workstation 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 PREDEFINED TEXT REPRESENTATION 0A

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

INQUIRE FILL AREA FACILITIES 0A

```
void ginq_fill_facs(
     Gint         ws_type,               /* workstation 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      */
);
```

INQUIRE PREDEFINED FILL AREA REPRESENTATION 0A

```
void ginq_pred_fill_rep(
     Gint           ws_type,     /* workstation type             */
     Gint           ind,         /* predefined index             */
     Gint           *err_ind,    /* OUT error indicator          */
     Gfill_bundle   *fill_rep    /* OUT predefined fill area rep. */
);
```

INQUIRE PATTERN FACILITIES 0A

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

INQUIRE PREDEFINED PATTERN REPRESENTATION 0A

```
void ginq_pred_pat_rep(
     Gint       ws_type,    /* workstation 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 `store`.

INQUIRE COLOUR FACILITIES 0A

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

INQUIRE PREDEFINED COLOUR REPRESENTATION 0A

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

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 0A

```
void ginq_list_avail_gdps(
     Gint        ws_type,                /* workstation 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 GENERALIZED DRAWING PRIMITIVE 0A

```
void ginq_gdp(
     Gint     ws_type,     /* workstation 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 */
);
```

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES 0A

```
void ginq_max_ws_st_tables(
     Gint                ws_type,    /* workstation type                */
     Gint                *err_ind,   /* OUT error indicator             */
     Gmax_ws_st_tables   *lengths    /* OUT lengths of workstation tables */
);
```

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED 1A

```
void ginq_num_seg_pris(
     Gint   ws_type,        /* workstation type              */
     Gint   *err_ind,       /* OUT error indicator           */
     Gint   *num_seg_pris   /* OUT num. of segment priorities */
);
```

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES 1A

```
void ginq_dyn_mod_seg_attrs(
    Gint                ws_type,    /* workstation type                */
    Gint                *err_ind,   /* OUT error indicator             */
    Gdyn_mod_seg_attrs  *dyn_mod    /* OUT dynamic modification of
                                       segment attributess             */
);
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES 0B

```
void ginq_num_avail_in(
    Gint      ws_type,    /* workstation type          */
    Gint      *err_ind,   /* OUT error indicator       */
    Gnum_in   *num_in     /* OUT num. of input devices */
);
```

INQUIRE DEFAULT LOCATOR DEVICE DATA 0B

```
void ginq_def_loc_data(
    Gint        ws_type,      /* workstation 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 DEFAULT STROKE DEVICE DATA 0B

```
void ginq_def_stroke_data(
    Gint          ws_type,        /* workstation 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 DEFAULT VALUATOR DEVICE DATA 0B

```
void ginq_def_val_data(
     Gint         ws_type,      /* workstation 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 DEFAULT CHOICE DEVICE DATA 0B

```
void ginq_def_choice_data(
     Gint            ws_type,            /* workstation 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 DEFAULT PICK DEVICE DATA 1B

```
void ginq_def_pick_data(
     Gint          ws_type,      /* workstation 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 DEFAULT STRING DEVICE DATA 0B

```
void ginq_def_string_data(
    Gint           ws_type,          /* workstation 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`.

7.9.6 Inquire functions for the Segment State List

INQUIRE SET OF ASSOCIATED WORKSTATIONS 1A

```
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   *ws,                   /* OUT list of workstations      */
    Gint        *num_elems_impl_list   /* OUT length of impl. list      */
);
```

INQUIRE SEGMENT ATTRIBUTES 1A

```
void ginq_seg_attrs(
    Gint         seg_name,   /* segment name                      */
    Gint         *err_ind,   /* OUT error indicator               */
    Gseg_attrs   *seg_attr   /* OUT segment attribute structure   */
);
```

7.9.7 Pixel Inquiries

INQUIRE PIXEL ARRAY DIMENSIONS 0A

```
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 */
);
```

INQUIRE PIXEL ARRAY **0A**

```
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 index array         */
);
```

REMARK. Befpore this function is called, the application shall allocate the parameter `pixel_array` as an integer array of size `dims->size_x*dims->size_y`. The entries of `pixel_array` shall be such that `pixel_array[i + j*dims->size_x]` corresponds with the (i,j)-th entry of the colour grid.

INQUIRE PIXEL **0A**

```
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        */
);
```

7.9.8 Inquiry Functions for Error State List

INQUIRE INPUT QUEUE OVERFLOW **0C**

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

7.10 Utility Functions

7.10.1 Utility Functions in GKS

EVALUATE TRANSFORMATION MATRIX **1A**

```
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      */
);
```

ACCUMULATE TRANSFORMATION MATRIX 1A

```
void gaccum_tran_matrix(
     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  */
);
```

7.10.2 Binding Specific Utilities

SET ERROR HANDLING 0A

```
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 3.10.

CREATE STORAGE 0A

```
void gcreate_store(
     Gint     *err_ind,    /* OUT error indicator              */
     Gstore   *store       /* OUT pointer to allocated storage */
);
```

Effect: See 3.9.2

DELETE STORAGE 0A

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

Effect: See 3.9.2

7.11 Error Handling

EMERGENCY CLOSE GKS 0A

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

ERROR HANDLING 0A

```
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 0A

```
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                        */
);
```

Annex A
(informative)

Compiled GKS/C Specification

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

1) The basic `#include` statements

```
        #include    <stdio.h>
        #include    <stddef.h>
```

2) The data types from clause 5 in compilation order;
3) The macros from clause 6 and annex C; the numerical values of the macros from annex C are optional;
4) The specification of the GKS functions as `extern void`;
5) Additional implementation-dependent items; they should preferably have the sentinels "g" or "G".

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

```
/*
 * -------------      gks.h      ------------------------------------------
 */

/*
 *      A.1  Standard #include Files
 */
#include        <stdio.h>
#include        <stddef.h>
/*
 *      A.2  Data Types in Compilation Order
 */
/*
 *      A.2.1  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];
/* --- Gdata  data --- */
   typedef struct {
       size_t   size;    /* size of data       */
       void     *data;   /* pointer to data    */
   } Gdata;
/*
 *      A.2.2  Enumeration Types
 */
```

```
/* --- Gasf  aspect source flag --- */
   typedef enum {
     GASF_BUNDLED, GASF_INDIV
   } Gasf;
/* --- Gattrs  attributes [used] --- */
   typedef enum {
     GATTR_LINE, GATTR_MARKER, GATTR_TEXT, GATTR_FILL
   } Gattrs;
/* --- Gattr_ctrl_flag  attribute control flag --- */
   typedef enum {
     GFLAG_CUR, GFLAG_SPECIF
   } Gattr_ctrl_flag;
/* --- Gclip_ind  clipping indicator --- */
   typedef enum {
     GIND_NO_CLIP, GIND_CLIP
   } Gclip_ind;
/* --- Gcolr_avail  colour available --- */
   typedef enum {
     GAVAIL_MONOCHR, GAVAIL_COLR
   } Gcolr_avail;
/* --- 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;
/* --- Gdefer_mode  deferral mode --- */
   typedef enum {
     GDEFER_ASAP, GDEFER_BNIG, GDEFER_BNIL, GDEFER_ASTI
   } Gdefer_mode;
/* --- Gdet  detectability --- */
   typedef enum {
     GSEG_UNDET, GSEG_DET
   } Gdet;
/* --- 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;
/* --- Gecho_switch  echo switch --- */
   typedef enum {
     GSWITCH_NO_ECHO, GSWITCH_ECHO
   } Gecho_switch;
```

```
/* --- Gfill_int_style  fill area interior style --- */
   typedef enum {
     GSTYLE_HOLLOW, GSTYLE_SOLID, GSTYLE_PAT, GSTYLE_HATCH
   } Gfill_int_style;
/* --- Ghighl  highlighting --- */
   typedef enum {
     GSEG_NORM, GSEG_HIGHL
   } Ghighl;
/* --- Ghor_text_align  horizontal text alignment --- */
   typedef enum {
     GHOR_NORM, GHOR_LEFT, GHOR_CTR, GHOR_RIGHT
   } Ghor_text_align;
/* --- Gin_class  input class --- */
   typedef enum {
     GIN_NONE,  GIN_LOC,  GIN_STROKE,  GIN_VAL,  GIN_CHOICE,
     GIN_PICK, GIN_STRING
   } Gin_class;
/* --- Ginq_type  inquire type --- */
   typedef enum {
     GINQ_SET, GINQ_REALIZED
   } Ginq_type;
/* --- 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;
/* --- Gline_fill_ctrl_flag  polyline/fill area control flag
         --- */
   typedef enum {
     GFLAG_LINE, GFLAG_FILL
   } Gline_fill_ctrl_flag;
/* --- Gnew_frame_nec_upd  new frame [action] necessary [at]
         update --- */
   typedef enum {
     GNEW_NO, GNEW_YES
   } Gnew_frame_nec_upd;
/* --- Gop_mode  operating mode --- */
   typedef enum {
     GOP_REQ, GOP_SAMPLE, GOP_EVENT
   } Gop_mode;
/* --- Gop_st  operating state --- */
   typedef enum {
     GST_GKCL, GST_GKOP, GST_WSOP, GST_WSAC, GST_SGOP
   } Gop_st;
/* --- Gpr_flag  prompt flag --- */
   typedef enum {
     GPR_OFF, GPR_ON
   } Gpr_flag;
```

```
/* --- Gpres_inval  presence [of] invalid [values] --- */
   typedef enum {
     GINVAL_ABSENT, GINVAL_PRESENT
   } Gpres_inval;
/* --- Grel_pri  relative priority --- */
   typedef enum {
     GPRI_HIGHER, GPRI_LOWER
   } Grel_pri;
/* --- Gmore_simult_events  more simultaneous events --- */
   typedef enum {
     GSIMULT_NO_MORE, GSIMULT_MORE
   } Gmore_simult_events;
/* --- Gin_status  [input] status --- */
   typedef enum {
     GIN_STATUS_NONE, GIN_STATUS_OK, GIN_STATUS_NO_IN
   } Gin_status;
/* --- 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;
/* --- 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;
/* --- Gvert_text_align  vertical text alignment --- */
   typedef enum {
     GVERT_NORM,    GVERT_TOP,    GVERT_CAP,    GVERT_HALF,
     GVERT_BASE, GVERT_BOTTOM
   } Gvert_text_align;
/* --- Gvis  visibility --- */
   typedef enum {
     GSEG_INVIS, GSEG_VIS
   } Gvis;
/* --- Gws_cat  workstation category --- */
   typedef enum {
     GCAT_OUT,  GCAT_IN,  GCAT_OUTIN,  GCAT_WISS,   GCAT_MO,
     GCAT_MI
   } Gws_cat;
/* --- Gws_class  workstation classification --- */
   typedef enum {
     GCLASS_VEC, GCLASS_RASTER, GCLASS_OTHER
   } Gws_class;
/* --- Gws_st  workstation state --- */
   typedef enum {
     GWS_INACTIVE, GWS_ACTIVE
   } Gws_st;
/*
 *      A.2.3  Simple Structures
 */
/* --- Gasfs  aspect source flags --- */
   typedef struct {
       Gasf    linetype;          /* linetype ASF                       */
       Gasf    linewidth;         /* linewidth scale factor ASF         */
       Gasf    line_colr_ind;     /* polyline colour index ASF          */
       Gasf    marker_type;       /* marker type ASF                    */
       Gasf    marker_size;       /* marker size scale factor ASF       */
       Gasf    marker_colr_ind;   /* polymarker colour index 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 index ASF              */
       Gasf    fill_int_style;    /* fill area interior style ASF       */
       Gasf    fill_style_ind;    /* fill area style index ASF          */
       Gasf    fill_colr_ind;     /* fill area colour index ASF         */
   } Gasfs;
/* --- Gcolr_facs  colour facilities --- */
   typedef struct {
       Gint           num_colrs;        /* num. of colours                    */
       Gcolr_avail    colr_avail;       /* colour availability                */
       Gint           num_pred_inds;    /* num. of predef. colour indices     */
   } Gcolr_facs;
/* --- 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] worksta-
          tion 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;
/* --- Gfill_bundle  fill area bundle --- */
```

```
   typedef struct {
       Gfill_int_style   int_style;   /* fill area interior style   */
       Gint              style_ind;   /* fill area style index      */
       Gint              colr_ind;    /* fill area colour index     */
   } Gfill_bundle;
/* --- Gfloat_size  float size --- */
   typedef struct {
       Gfloat   size_x;   /* x size   */
       Gfloat   size_y;   /* y size   */
   } Gfloat_size;
/* --- Gint_list  integer list --- */
   typedef struct {
       Gint    num_ints;   /* num. 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;
/* --- 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;
/* --- Gline_bundle  polyline bundle --- */
   typedef struct {
       Gint     type;       /* linetype                 */
       Gfloat   width;      /* linewidth scale factor   */
       Gint     colr_ind;   /* polyline colour index    */
   } Gline_bundle;
/* --- Gmarker_bundle  polymarker bundle --- */
   typedef struct {
       Gint     type;       /* marker type                */
       Gfloat   size;       /* marker size scale factor   */
       Gint     colr_ind;   /* polymarker colour index    */
   } Gmarker_bundle;
/* --- 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;
/* --- Gnum_in  number [of] input [devices] --- */
   typedef struct {
        Gint    loc;       /* num. of locator devices    */
        Gint    stroke;    /* num. of stroke devices     */
        Gint    val;       /* num. of valuator devices   */
        Gint    choice;    /* num. of choice devices     */
        Gint    pick;      /* num. of pick devices       */
        Gint    string;    /* num. of string devices     */
   } Gnum_in;
/* --- Gpick  pick [value] --- */
   typedef struct {
        Gint    seg_name;   /* segment name      */
        Gint    pick_id;    /* pick identifier   */
   } Gpick;
/* --- 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_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;
/* --- Gtext_align  text alignment --- */
   typedef struct {
        Ghor_text_align   hor;    /* horizontal component   */
        Gvert_text_align  vert;   /* vertical component     */
   } Gtext_align;
/* --- Gtext_font_prec  text font [and] precision --- */
   typedef struct {
        Gint          font;   /* text font        */
        Gtext_prec    prec;   /* text precision   */
   } Gtext_font_prec;
/* --- Gvec  vector --- */
   typedef struct {
        Gfloat   delta_x;   /* x coordinate   */
        Gfloat   delta_y;   /* y coordinate   */
   } Gvec;
/* --- 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;
```

```
/*
 *      A.2.4  Nested Structures
 */
/* --- Gclip  clipping --- */
   typedef struct {
       Gclip_ind    clip_ind;     /* clipping indicator    */
       Glimit       clip_rect;    /* clipping rectangle    */
   } Gclip;
/* --- 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;
/* --- Gfill_attrs  fill area attributes --- */
   typedef struct {
       Gasf            int_style_asf;    /* fill area interior style asf    */
       Gasf            style_ind_asf;    /* fill area style index asf       */
       Gasf            colr_ind_asf;     /* fill area colour index asf      */
       Gint            ind;              /* fill area index                 */
       Gfill_bundle    bundle;           /* fill area bundle                */
   } Gfill_attrs;
/* --- Gfill_facs  fill area facilities --- */
   typedef struct {
       Gint                num_int_styles;   /* num. of interior styles                */
       Gfill_int_style     int_styles[4];    /* list of available interior styles      */
       Gint_list           hatch_styles;     /* list of available hatch styles         */
       Gint                num_pred_inds;    /* num. of predef. fill area indices      */
   } Gfill_facs;
/* --- 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;
/* --- Gline_attrs  polyline attributes --- */
   typedef struct {
       Gasf            type_asf;       /* linetype asf                  */
       Gasf            width_asf;      /* linewidth asf                 */
       Gasf            colr_ind_asf;   /* polyline colour index asf     */
       Gint            ind;            /* polyline index                */
```

```
        Gline_bundle   bundle;          /* polyline bundle           */
    } Gline_attrs;
/* --- Gline_facs  polyline facilities --- */
    typedef struct {
        Gint_list    types;           /* list of linetypes                */
        Gint         num_widths;      /* num. of available linewidths     */
        Gfloat       nom_width;       /* nominal linewidth                */
        Gfloat       min_width;       /* min. linewidth                   */
        Gfloat       max_width;       /* max. linewidth                   */
        Gint         num_pred_inds;   /* num. of predef. polyline indices */
    } Gline_facs;
/* --- Gmarker_attrs  polymarker attributes --- */
    typedef struct {
        Gasf            type_asf;       /* marker type asf              */
        Gasf            size_asf;       /* marker size scale factor asf */
        Gasf            colr_ind_asf;   /* marker colour index asf      */
        Gint            ind;            /* polymarker index             */
        Gmarker_bundle  bundle;         /* polymarker bundle            */
    } Gmarker_attrs;
/* --- Gmarker_facs  polymarker facilities --- */
    typedef struct {
        Gint_list    types;           /* list of marker types               */
        Gint         num_sizes;       /* num. 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;   /* num. of predef. polymarker indices */
    } Gmarker_facs;
/* --- Gpat_rep  pattern representation --- */
    typedef struct {
        Gint_size    dims;          /* colour array's dimensions   */
        Gint         *colr_array;   /* colour array                */
    } Gpat_rep;
/* --- 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;
/* --- Gpoint_list  point list --- */
    typedef struct {
        Gint      num_points;    /* num. of points in the list    */
        Gpoint   *points;      /* list of points                  */
    } Gpoint_list;
/* --- Grect  rectangle --- */
    typedef struct {
        Gpoint   p;    /* point p    */
        Gpoint   q;    /* point q    */
    } Grect;
/* --- Gtext_extent  text extent --- */
    typedef struct {
        Gpoint   concat_point;    /* concatenation point          */
        Gpoint   paral[4];        /* text extent parallelogram    */
    } Gtext_extent;
/* --- 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           */
        Gint               colr_ind;          /* text colour index           */
    } Gtext_bundle;
/* --- Gtext_facs  text facilities --- */
    typedef struct {
        Gint               num_font_precs;     /* num. of fonts and precisions       */
        Gtext_font_prec    *font_precs;        /* list of fonts and precisions       */
        Gint               num_char_hts;       /* num. of character heights          */
        Gfloat             min_char_ht;        /* min. character height              */
        Gfloat             max_char_ht;        /* max. character height              */
        Gint               num_char_expans;    /* num. of char. expansion factors    */
        Gfloat             min_char_expan;     /* min. expansion factor              */
        Gfloat             max_char_expan;     /* max. expansion factor              */
        Gint               num_pred_inds;      /* num. of predef. text indices       */
    } Gtext_facs;
/* --- Gtran  transformation --- */
    typedef struct {
        Glimit   win;    /* window     */
        Glimit   vp;     /* viewport   */
    } Gtran;
/*
 *      A.2.5  Implementation Dependent Types
 */
/* --- Gchoice_data  choice data record    --- */
    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_rep  colour representation --- */
    typedef union {
        Grgb      rgb;      /* Red Green Blue colour specification */
    } Gcolr_rep;
/* --- Gescape_in_data  escape input data record --- */
    typedef union {
        struct Gescape_in_r1 {
                     Gdata    reg_dep;     /* reg. 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    reg_dep;     /* reg. 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    reg_dep;     /* reg. dep. */
        } gdp_r1;    /* gdp 1 data */
        struct Ggdp_u1 {
                   Gdata    impl_dep;    /* impl. dep. */
        } gdp_u1;    /* gdp -1 data */
        /* etc. */
    } Ggdp_data;
/* --- Gitem_data  item data record --- */
    typedef struct {
        Gint                type;           /* item type                           */
        Gint                length;         /* item data record length             */
        union {
           Gctrl_flag       clear_ws;       /* control flag                     */
           Gupd_regen_flag  upd_ws;         /* regen. flag                      */
           struct Gdefer_st {
                            Gdefer_mode    defer_mode;
                                           /* deferral mode                    */
                            Girg_mode      irg_mode;
                                           /* irg mode                         */
           } defer_st;                     /* deferrral state                  */
           /* etc. */
           Gdata            gdp_unsupp;     /* GDPs not supported by impl.      */
           Gdata            impl_dep;       /* impl. dependent                  */
        } data;
    } Gitem_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;
                        /* polyline               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;
                        /* polyline/fill area     control flag */
                        union Gloc_attrs {
                                                  Gline_attrs    line_attrs;
                                                  /* polyline    attrs. */
                                                  Gfill_attrs    fill_attrs;
                                                  /* fill area   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 record  --- */
    typedef struct {
        union Gpick_pets {
            struct Gpick_pet_r1 {
                    Gdata    impl_dep;    /* impl. dep. */
            } pet_r1;   /* pet 1 data */
            struct Gpick_pet_r2 {
                    Gdata    impl_dep;    /* impl. dep. */
            } pet_r2;   /* pet 2 data */
            struct Gpick_pet_r3 {
                    Gdata    impl_dep;    /* impl. dep. */
            } pet_r3;   /* pet 3 data */
            struct Gpick_pet_u1 {
                    Gdata    impl_dep;    /* impl. dep. */
            } 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 [editing] position */
        union string_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 record  --- */
    typedef struct {
        Gint      in_buf_size;      /* input buffer size (nr. of points) */
        Gint      init_pos;         /* initial [editing] position */
        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;
                                  /* polymarker     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;
                                  /* polyline       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;
/*
 *      A.3  Macros
 */
/*
 *      A.3.1  Error Macros
 */
#define GE_NO_ERR                  (0)/* No Errors */
/* operating state errors */
#define GE_NOT_GKCL                (1)/* GKS not in proper state: GKS shall be in
```

```
                                    the state GKCL */
#define GE_NOT_GKOP             (2)/* GKS not in proper state: GKS shall be in
                                    the state GKOP */
#define GE_NOT_WSAC             (3)/* GKS not in proper state: GKS shall be in
                                    the state WSAC */
#define GE_NOT_SGOP             (4)/* GKS not in proper state: GKS shall be in
                                    the state SGOP */
#define GE_NOT_WSAC_SGOP        (5)/* GKS not in proper state: GKS shall be
                                    either in the state WSAC or in the state SGOP
                                    */
#define GE_NOT_WSOP_WSAC        (6)/* GKS not in proper state: GKS shall be
                                    either in the state WSOP or in the state WSAC
                                    */
#define GE_GKCL_GKOP            (7)/* GKS not in proper state: GKS shall be in
                                    one of the states WSOP, WSAC or SGOP */
#define GE_GKCL                 (8)/* GKS not in proper state: GKS shall be in
                                    one of the states GKOP, WSOP, WSAC or SGOP */

/* workstation errors */
#define GE_WS_ID_INVAL          (20)/* Specified workstation identifier is
                                    invalid */
#define GE_CONN_ID_INVAL        (21)/* Specified connection identifier is invalid
                                    */
#define GE_WS_TYPE_INVAL        (22)/* Specified workstation type is invalid */
#define GE_NO_WS_TYPE           (23)/* Specified workstation type does not exist
                                    */
#define GE_WS_OPEN              (24)/* Specified workstation is open */
#define GE_WS_NOT_OPEN          (25)/* Specified workstation is not open */
#define GE_WS_CANT_OPEN         (26)/* Specified workstation cannot be opened */
#define GE_WISS_NOT_OPEN        (27)/* Workstation Independent Segment Storage is
                                    not open */
#define GE_WISS_OPEN            (28)/* Workstation Independent Segment Storage is
                                    already open */
#define GE_WS_ACTIVE            (29)/* Specified workstation is active */
#define GE_WS_INACTIVE          (30)/* Specified workstation is not active */
#define GE_WS_MO                (31)/* Specified workstation is of category MO */
#define GE_WS_NOT_MO            (32)/* Specified workstation is not of category
                                    MO */
#define GE_WS_MI                (33)/* Specified workstation is of category MI */
#define GE_WS_NOT_MI            (34)/* Specified workstation is not of category
                                    MI */
#define GE_WS_IN                (35)/* Specified workstation is of category INPUT
                                    */
#define GE_WS_WISS              (36)/* Specified workstation is Workstation
                                    Independent Segment Storage */
#define GE_WS_NOT_OUTIN         (37)/* Specified workstation is not of category
                                    OUTIN */
#define GE_WS_NOT_IN_OUTIN      (38)/* Specified workstation is neither of
                                    category INPUT nor of category OUTIN */
#define GE_WS_NOT_OUT_OUTIN     (39)/* Specified workstation is neither of
                                    category OUTPUT nor of category OUTIN */
#define GE_WS_NO_PIXEL          (40)/* Specified workstation has no pixel store
                                    readback capability */
```

```
#define GE_WS_TYPE_NO_GDP        (41)/* Specified workstation type is not able to
                                     generate the specified generalized drawing
                                     primitive */
#define GE_WS_MAX_OPEN           (42)/* Maximum number of simultaneously open
                                     workstations would be exceeded */
#define GE_WS_MAX_ACTIVE         (43)/* Maximum number of simultaneously active
                                     workstations would be exceeded */

/* transformation errors */
#define GE_TRAN_NUM_INVAL        (50)/* Transformation number is invalid */
#define GE_RECT_INVAL            (51)/* Rectangle definition is invalid */
#define GE_VP_INVAL              (52)/* Viewport is not within the Normalized Dev-
                                     ice Coordinate unit square */
#define GE_WIN_INVAL             (53)/* Workstation window is not within the Nor-
                                     malized Device Coordinate unit square */
#define GE_WS_VP_INVAL           (54)/* Workstation viewport is not within the
                                     display space */

/* output attribute errors */
#define GE_LINE_IND_INVAL        (60)/* Polyline index is invalid */
#define GE_LINE_REP_UNDEF        (61)/* A representation for the specified poly-
                                     line index has not been defined on this
                                     workstation */
#define GE_LINE_REP_NOT_PRED     (62)/* A representation for the specified poly-
                                     line index has not been predefined on this
                                     workstation */
#define GE_LINETYPE_ZERO         (63)/* Linetype is equal to zero */
#define GE_LINETYPE_NOT_WS       (64)/* Specified linetype is not supported on
                                     this workstation */
#define GE_LINEWIDTH_LT_ZERO     (65)/* Linewidth scale factor is less than zero
                                     */
#define GE_MARKER_IND_INVAL      (66)/* Polymarker index is invalid */
#define GE_MARKER_REP_UNDEF      (67)/* A representation for the specified poly-
                                     marker index has not been defined on this
                                     workstation */
#define GE_MARKER_REP_NOT_PRED (68)/* A representation for the specified poly-
                                     marker index has not been predefined on this
                                     workstation */
#define GE_MARKER_TYPE_ZERO      (69)/* Marker type is equal to zero */
#define GE_MARKER_TYPE_NOT_WS    (70)/* Specified marker type is not supported on
                                     this workstation */
#define GE_MARKER_SIZE_LT_ZERO (71)/* Marker size scale factor is less than zero
                                     */
#define GE_TEXT_IND_INVAL        (72)/* Text index is invalid */
#define GE_TEXT_REP_UNDEF        (73)/* A representation for the specified text
                                     index has not been defined on this worksta-
                                     tion */
#define GE_TEXT_REP_NOT_PRED     (74)/* A representation for the specified text
                                     index has not been predefined on this works-
                                     tation */
#define GE_FONT_ZERO             (75)/* Text font is equal to zero */
#define GE_FONT_NOT_WS           (76)/* Requested text font is not supported for
                                     the specified precision on this workstation
                                     */
```

```
#define GE_EXPAN_LE_ZERO          (77)/* Character expansion factor is less than or
                                      equal to zero */
#define GE_HT_LE_ZERO             (78)/* Character height is less than or equal to
                                      zero */
#define GE_UP_VEC_ZERO            (79)/* Length of character up vector is zero */
#define GE_FILL_IND_INVAL         (80)/* Fill area index is invalid */
#define GE_FILL_REP_UNDEF         (81)/* A representation for the specified fill
                                      area index has not been defined on this
                                      workstation */
#define GE_FILL_REP_NOT_PRED      (82)/* A representation for the specified fill
                                      area index has not been predefined on this
                                      workstation */
#define GE_INT_STYLE_NOT_WS       (83)/* Specified fill area interior style is not
                                      supported on this workstation */
#define GE_STYLE_IND_ZERO         (84)/* Style (pattern or hatch) index is equal to
                                      zero */
#define GE_PAT_IND_INVAL          (85)/* Specified pattern index is invalid */
#define GE_HATCH_STYLE_NOT_WS     (86)/* Specified hatch style is not supported on
                                      this workstation */
#define GE_PAT_SIZE_LE_ZERO       (87)/* Pattern size value is not positive */
#define GE_PAT_REP_UNDEF          (88)/* A representation for the specified pattern
                                      index has not been defined on this worksta-
                                      tion */
#define GE_PAT_REP_NOT_PRED       (89)/* A representation for the specified pattern
                                      index has not been predefined on this works-
                                      tation */
#define GE_PAT_NOT_WS             (90)/* Interior style PATTERN is not supported on
                                      this workstation */
#define GE_DIM_INVAL              (91)/* Dimensions of colour array are invalid */
#define GE_COLR_IND_LT_ZERO       (92)/* Colour index is less than zero */
#define GE_COLR_IND_INVAL         (93)/* Colour index is invalid */
#define GE_COLR_REP_UNDEF         (94)/* A representation for the specified colour
                                      index has not been defined on this worksta-
                                      tion */
#define GE_COLR_REP_NOT_PRED      (95)/* A representation for the specified colour
                                      index has not been predefined on this works-
                                      tation */
#define GE_COLR_INVAL             (96)/* Colour component is outside valid range
                                      for colour model */
#define GE_PICK_ID_INVAL          (97)/* Pick identifier is invalid */
/* output errors */
#define GE_NUM_POINT_INVAL        (100)/* Number of points is invalid */
#define GE_INVAL_CODE             (101)/* Invalid code in string */
#define GE_GDP_ID_INVAL           (102)/* Generalized drawing primitive identifier
                                      is invalid */
#define GE_GDP_DATA_INVAL         (103)/* Content of generalized drawing primitive
                                      data record is invalid */
#define GE_CANT_GEN_GDP           (104)/* At least one active workstation is not
                                      able to generate the specified generalized
                                      drawing primitive */
#define GE_CANT_GEN_GDP_CLIP      (105)/* At least one active workstation is not
                                      able to generate the specified generalized
```

```
                                   drawing primitive under the current transfor-
                                   mations and clipping conditions */
/* segment errors */
#define GE_SEG_NAME_INVAL          (120)/* Specified segment name is invalid */
#define GE_SEG_NAME_USED           (121)/* Specified segment name is already in use
                                   */
#define GE_SEG_ABSENT              (122)/* Specified segment does not exist */
#define GE_SEG_NOT_WS              (123)/* Specified segment does not exist on
                                   specified workstation */
#define GE_SEG_NOT_WISS            (124)/* Specified segment does not exist on
                                   Workstation Independent Segment Storage */
#define GE_SEG_OPEN                (125)/* Specified segment is open */
#define GE_SEG_PRI_INVAL           (126)/* Segment priority is outside the range
                                   [0,1] */
/* input errors */
#define GE_IN_DEV_NOT_WS           (140)/* Specified input device is not present on
                                   workstation */
#define GE_IN_DEV_NOT_REQ          (141)/* Input device is not in REQUEST mode */
#define GE_IN_DEV_NOT_SAMPLE       (142)/* Input device is not in SAMPLE mode */
#define GE_EV_SAMPLE_UNAVAIL       (143)/* EVENT and SAMPLE input mode are not
                                   available at this level of GKS */
#define GE_PET_NOT_WS              (144)/* Specified prompt and echo type is not
                                   supported on this workstation */
#define GE_ECHO_INVAL              (145)/* Echo area is outside display space */
#define GE_IN_DATA_INVAL           (146)/* Contents of input data record are invalid
                                   */
#define GE_QUE_OVERF               (147)/* Input queue has overflowed */
#define GE_NO_QUE_OVERF            (148)/* Input queue has not overflowed since GKS
                                   was opened or the last invocation of INQUIRE
                                   INPUT QUEUE OVERFLOW */
#define GE_ASSOC_WS_CLOSED         (149)/* Input queue has overflowed, but associ-
                                   ated workstation has been closed */
#define GE_NO_CUR_EV               (150)/* No input value of the correct class is in
                                   the current event report */
#define GE_TIMEOUT_INVAL           (151)/* Timeout is invalid */
#define GE_INIT_INVAL              (152)/* Initial value is invalid */
#define GE_INIT_STROKE_INVAL       (153)/* Number of points in the initial stroke is
                                   greater than the buffer size */
#define GE_INIT_STRING_INVAL       (154)/* Length of the initial string is greater
                                   than the buffer size */
/* GKSM errors */
#define GE_ITEM_RESERVED           (160)/* Item type is not allowed for user items
                                   */
#define GE_ITEM_LENGTH_INVAL       (161)/* Item length is invalid */
#define GE_NO_ITEM_MI              (162)/* No item is left in GKS Metafile input */
#define GE_ITEM_INVAL              (163)/* Metafile item is invalid */
#define GE_ITEM_GKS_INVAL          (164)/* Item type is not a valid GKS item */
#define GE_ITEM_DATA_INVAL         (165)/* Content of item data record is invalid
                                   for the specified item type */
#define GE_MAX_ITEM_DATA_INVAL     (166)/* Maximum item data record length is
                                   invalid */
#define GE_USER_ITEM               (167)/* User item cannot be interpreted */
```

```
#define GE_FUNC_UNAVAIL          (168)/* Specified function is not supported in
                                     this level of GKS */
/* escape errors */
#define GE_ESCAPE_FUNC_UNAVAIL (180)/* Specified escape function is not sup-
                                     ported */
#define GE_ESCAPE_ID_INVAL       (181)/* Specified escape function identification
                                     is invalid */
#define GE_ESCAPE_DATA_INVAL     (182)/* Contents of escape data record are
                                     invalid */
/* error file errors */
#define GE_ERR_FILE_INVAL        (200)/* Specified error file is invalid */
/* I/O errors */
#define GE_MEM_OVERF             (300)/* Storage overflow has occurred in GKS */
#define GE_SEG_MEM_OVERF         (301)/* Storage overflow has occurred in segment
                                     storage */
#define GE_IO_ERR_READ           (302)/* Input/Output error has occurred while
                                     reading */
#define GE_IO_ERR_WRITE          (303)/* Input/Output error has occurred while
                                     writing */
#define GE_IO_ERR_WRITE_WS       (304)/* Input/Output error has occurred while
                                     sending data to a workstation */
#define GE_IO_ERR_READ_WS        (305)/* Input/Output error has occurred while
                                     receiving data from a workstation */
#define GE_IO_ERR_LIB            (306)/* Input/Output error has occurred during
                                     program library management */
#define GE_IO_ERR_WS_TABLE       (307)/* Input/Output error has occurred while
                                     reading workstation description table */
#define GE_ARITH_ERR             (308)/* Arithmetic error has occurred */
/* binding specific errors */
#define GE_START_IND_INVAL       (2200)/* Start index out of range*/
#define GE_APPL_LIST_LENGTH_LT_ZERO (2201)/* Length of application list is nega-
                                     tive */
#define GE_ENUM_TYPE_INVAL       (2202)/* Enumeration type out of range*/
#define GE_ALLOC_STORE           (2203)/* Error while allocating Store */
#define GE_ALLOC_MEM_STORE       (2204)/* Error while allocating memory for Store
                                     */

/*
 *      A.3.2   Function Macros
 */
#define Gfn_open_gks                 (0)
#define Gfn_close_gks                (1)
#define Gfn_open_ws                  (2)
#define Gfn_close_ws                 (3)
#define Gfn_activate_ws              (4)
#define Gfn_deactivate_ws            (5)
#define Gfn_clear_ws                 (6)
#define Gfn_redraw_all_segs_ws       (7)
#define Gfn_upd_ws                   (8)
#define Gfn_set_defer_st             (9)
#define Gfn_message                  (10)
#define Gfn_escape                   (11)
#define Gfn_polyline                 (12)
```

```
#define Gfn_polymarker                    (13)
#define Gfn_text                          (14)
#define Gfn_fill_area                     (15)
#define Gfn_cell_array                    (16)
#define Gfn_gdp                           (17)
#define Gfn_set_line_ind                  (18)
#define Gfn_set_linetype                  (19)
#define Gfn_set_linewidth                 (20)
#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_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_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_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_pat_size                  (39)
#define Gfn_set_pat_ref_point             (40)
#define Gfn_set_asfs                      (41)
#define Gfn_set_pick_id                   (42)
#define Gfn_set_line_rep                  (43)
#define Gfn_set_marker_rep                (44)
#define Gfn_set_text_rep                  (45)
#define Gfn_set_fill_rep                  (46)
#define Gfn_set_pat_rep                   (47)
#define Gfn_set_colr_rep                  (48)
#define Gfn_set_win                       (49)
#define Gfn_set_vp                        (50)
#define Gfn_set_vp_in_pri                 (51)
#define Gfn_sel_norm_tran                 (52)
#define Gfn_set_clip_ind                  (53)
#define Gfn_set_ws_win                    (54)
#define Gfn_set_ws_vp                     (55)
#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_tran                  (64)
```

```
#define Gfn_set_vis                        (65)
#define Gfn_set_highl                      (66)
#define Gfn_set_seg_pri                    (67)
#define Gfn_set_det                        (68)
#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_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_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_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_event                    (93)
#define Gfn_flush_events                   (94)
#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_write_item                     (101)
#define Gfn_get_item_type                  (102)
#define Gfn_read_item                      (103)
#define Gfn_interpret_item                 (104)
#define Gfn_eval_tran_matrix               (105)
#define Gfn_accum_tran_matrix              (106)
#define Gfn_emergency_close_gks            (153)
#define Gfn_err_hand                       (154)
#define Gfn_err_log                        (155)
#define Gfn_set_err_hand                   (156)
/*
 *      A.3.3  Miscellaneous Macros
 */
#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 line-
type */
#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 */
#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 GSTRING_DEF     (1)   /* String default */
#define GDEF_MEM_SIZE   ((size_t) (-1))/* Default memory size
*/
#define GDEF_ERR_FILE   ((char *) (""))/* Default error file
name*/
/*
 *       A.3.4  GKSM Item Macros
 */
/*
 * control items
 */
#define Gksm_end_item              ( 0) /*  end item */
#define Gksm_clear_ws              ( 1) /*  clear workstation */
#define Gksm_redraw_all_segs_ws    ( 2) /*  redraw all segments */
#define Gksm_upd_ws                ( 3) /*  update workstation */
#define Gksm_defer_st              ( 4) /*  deferral state */
#define Gksm_message               ( 5) /*  message */
#define Gksm_escape                ( 6) /*  escape */
/*
 * output items
 */
#define Gksm_polyline                (11) /*  polyline */
#define Gksm_polymarker              (12) /*  polymarker */
#define Gksm_text                    (13) /*  text */
#define Gksm_fill_area               (14) /*  fill area */
#define Gksm_cell_array              (15) /*  cell array */
#define Gksm_gdp                     (16) /*  gdp */
/*
 * output attribute items
 */
#define Gksm_line_ind                (21) /*  polyline index */
#define Gksm_linetype                (22) /*  linetype */
#define Gksm_linewidth               (23) /*  linewidth scale
#define Gksm_line_colr_ind           (24) /*  polyline colour index */
#define Gksm_marker_ind              (25) /*  polymarker index */
#define Gksm_marker_type             (26) /*  marker type */
#define Gksm_marker_size             (27) /*  marker size scale
#define Gksm_marker_colr_ind         (28) /*  polymarker colour
#define Gksm_text_ind                (29) /*  text index */
#define Gksm_text_font_prec          (30) /*  text font
#define Gksm_char_expan              (31) /*  character expansion
#define Gksm_char_space              (32) /*  character spacing */
#define Gksm_text_colr_ind           (33) /*  text colour index */
#define Gksm_char_vec                (34) /*  character vectors */
#define Gksm_text_path               (35) /*  text path */
#define Gksm_text_align              (36) /*  text alignment */
#define Gksm_fill_ind                (37) /*  fill area index */
#define Gksm_fill_int_style          (38) /*  fill area interior style */
#define Gksm_fill_style_ind          (39) /*  fill area style index */
#define Gksm_fill_colr_ind           (40) /*  fill area colour index */
#define Gksm_pat_ref_point           (41) /*  pattern reference point  */
#define Gksm_pat_vecs                (42) /*  pattern vectors */
#define Gksm_asf                     (43) /*  aspect source flags */
#define Gksm_pick_id                 (44) /*  pick identifier */
/*
 * workstation attribute items
 */
#define Gksm_line_rep                (51) /*  polyline repr. */
#define Gksm_marker_rep              (52) /*  polymarker repr. */
#define Gksm_text_rep                (53) /*  text repr. */
#define Gksm_fill_rep                (54) /*  fill area repr. */
#define Gksm_pat_rep                 (55) /*  pattern repr. */
#define Gksm_colr_rep                (56) /*  colour repr. */
/*
 * transformation items
 */
#define Gksm_clip_rect               (61) /*  clipping rectangle */
#define Gksm_ws_win                  (71) /*  workstation window */
#define Gksm_ws_vp                   (72) /*  workstation viewport */
/*
 * segment items
 */
```

```
#define Gksm_create_seg          (81) /*  create segment */
#define Gksm_close_seg           (82) /*  close segment */
#define Gksm_rename_seg          (83) /*  rename segment */
#define Gksm_del_seg             (84) /*  delete segment */
#define Gksm_set_seg_tran        (91) /*  segment transformation */
#define Gksm_set_vis             (92) /*  visibility  */
#define Gksm_set_highl           (93) /*  highlighting */
#define Gksm_set_seg_pri         (94) /*  segment priiority */
#define Gksm_set_det             (95) /*  detectability */
/*
 * user items
 */
#define Gksm_user_item_start     (100)  /* user item start */
/*
 *       A.4  GKS/C Function Specification
 */
/*
 *       A.4.1  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 gopen_ws(
     Gint         ws_id,       /* workstation identifier    */
     const void   *conn_id,    /* connection identifier     */
     Gint         ws_type      /* workstation type          */);
extern void gclose_ws(
     Gint     ws_id    /* workstation identifier    */);
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              */);
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     */);
extern void gmessage(
     Gint         ws_id,      /* workstation identifier    */
     const char   *message    /* message string            */);
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   */);
/*
 *      A.4.2  Output Functions
 */
extern void gpolyline(
     const Gpoint_list   *point_list   /* list of points   */);
extern void gpolymarker(
     const Gpoint_list   *point_list   /* list of points   */);
extern void gtext(
     const Gpoint    *text_pos,      /* text position       */
     const char      *char_string    /* character string    */);
extern void gfill_area(
     const Gpoint_list   *point_list   /* list of points   */);
extern void gcell_array(
     const Grect       *rect,          /* cell rectangle    */
     const Gpat_rep    *colr_array     /* colour array      */);
extern void ggdp(
     const Gpoint_list   *point_list,    /* list of points     */
     Gint                gdp_id,         /* gdp identifier     */
     const Ggdp_data     *gdp_data       /* gdp data record    */);
/*
 *      A.4.3  Output Attribute Functions
 */
extern void gset_line_ind(
     Gint    line_ind    /* polyline index    */);
extern void gset_linetype(
     Gint    linetype    /* linetype  */);
extern void gset_linewidth(
     Gfloat    linewidth    /* linewidth scale factor   */);
extern void gset_line_colr_ind(
     Gint    line_colr_ind  /* polyline colour index   */);
extern void gset_marker_ind(
     Gint    marker_ind   /* polymarker 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_ind(
     Gint    marker_colr_ind    /* polymarker 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_ind(
     Gint    text_colr_ind    /* text colour index    */);
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_path(
     Gtext_path   text_path   /* text path   */);
extern void gset_text_align(
     const Gtext_align   *text_align   /* text alignment   */);
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_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_asfs(
     const Gasfs   *list_asf   /* list of aspect source flags   */);
extern void gset_pick_id(
     Gint   pick_id   /* pick identifier   */);
extern void gset_line_rep(
     Gint                   ws_id,          /* workstation identifier     */
     Gint                   line_ind,       /* polyline index             */
     const Gline_bundle     *line_bundle    /* polyline representation    */);
extern void gset_marker_rep(
     Gint                     ws_id,           /* workstation identifier      */
     Gint                     marker_ind,      /* polymarker index            */
     const Gmarker_bundle     *marker_bundle   /* polymarker 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_fill_rep(
     Gint                   ws_id,          /* workstation identifier      */
     Gint                   fill_ind,       /* fill area index             */
     const Gfill_bundle     *fill_bundle    /* fill 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    */);
/*
 *      A.4.4   Transformation Functions
 */
extern void gset_win(
     Gint            tran_num,      /* transformation number   */
     const Glimit   *win_limits   /* window limits           */);
extern void gset_vp(
     Gint           tran_num,    /* transformation number   */
     const Glimit   *vp_limits   /* 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 gsel_norm_tran(
     Gint     tran_num   /* transformation number   */);
extern void gset_clip_ind(
     Gclip_ind    clip_ind   /* clipping indicator   */);
extern void gset_ws_win(
     Gint           ws_id,          /* workstation identifier       */
     const Glimit   *ws_win_limits  /* workstation window limits    */);
extern void gset_ws_vp(
     Gint           ws_id,         /* workstation identifier        */
     const Glimit   *ws_vp_limits  /* workstation viewport limits   */);
/*
 *      A.4.5   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              */
     Gtran_matrix   tran_matrix   /* transformation matrix     */);
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_seg_pri(
     Gint      seg_name,    /* segment name       */
     Gfloat    seg_pri      /* segment priority   */);
extern void gset_det(
     Gint    seg_name,    /* segment name    */
     Gdet    det          /* detectability   */);
/*
 *      A.4.6  Input Functions
 */
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_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_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_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_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 gflush_events(
     Gint         ws_id,    /* workstation identifier          */
     Gin_class    class,    /* device class                    */
     Gint         in_num    /* 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    */);
/*
 *      A.4.7   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          */);
/*
 *      A.4.8   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_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_type,              /* OUT list of avalailable ws types  */
     Gint         *num_elems_impl_list   /* OUT length of impl. list         */);
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_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_cur_prim_attrs(
     Gint           *err_ind,      /* OUT error indicator     */
     Gprim_attrs    *prim_attrs    /* OUT current primitive
                                      attribute structure     */);
extern void ginq_line_ind(
     Gint    *err_ind,    /* OUT error indicator          */
     Gint    *line_ind    /* OUT current polyline index   */);
extern void ginq_marker_ind(
     Gint    *err_ind,      /* OUT error indicator            */
     Gint    *marker_ind    /* OUT current polymarker index   */);
extern void ginq_text_ind(
     Gint    *err_ind,    /* OUT error indicator      */
     Gint    *text_ind    /* OUT current text index   */);
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_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_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_fill_ind(
     Gint    *err_ind,    /* OUT error indicator          */
     Gint    *fill_ind    /* OUT current fill area index   */);
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_pat_ref_point(
     Gint      *err_ind,          /* OUT error indicator                    */
     Gpoint    *pat_ref_point     /* OUT current pattern reference point    */);
extern void ginq_cur_pick_id(
     Gint    *err_ind,    /* OUT error indicator          */
     Gint    *pick_id     /* OUT current pick identifier   */);
extern void ginq_cur_indiv_attrs(
     Gint            *err_ind,      /* OUT error indicator         */
     Gindiv_attrs    *indiv_attr    /* OUT current individual
                                       attribute  structure     */);
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_ind(
     Gint    *err_ind,          /* OUT error indicator                     */
     Gint    *line_colr_ind     /* OUT current polyline colour 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_ind(
     Gint    *err_ind,            /* OUT error indicator                        */
     Gint    *marker_colr_ind     /* OUT current polymarker colour 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_ind(
     Gint    *err_ind,          /* OUT error indicator              */
     Gint    *text_colr_ind     /* OUT current text colour 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_asfs(
     Gint     *err_ind,    /* OUT error indicator                 */
     Gasfs    *list_asf    /* OUT current aspect 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_tran_nums(
     Gint         num_elems_appl_list,    /* length of application list   */
     Gint         start_ind,              /* starting index               */
     Gint         *err_ind,               /* OUT error indicator          */
     Gint_list    *norm_tran_num,         /* OUT list of normalization
                                             transformation numbers       */
     Gint         *num_elems_impl_list    /* OUT length of impl. list     */);
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_clip(
     Gint     *err_ind,         /* OUT error indicator                */
     Gclip    *clip_ind_rect    /* OUT current clipping indicator
                                   and rectangle                      */);
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_more_simult_events(
     Gint                  *err_ind,          /* OUT error indicator                   */
     Gmore_simult_events   *simult_events     /* OUT [more] simultaneous events        */);
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_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 implicit regeneration 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_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 polyline indices  */
     Gint          *num_elems_impl_list    /* OUT length of impl. list              */
extern void ginq_line_rep(
     Gint            ws_id,      /* workstation identifier       */
     Gint            line_ind,   /* polyline index               */
     Ginq_type       type,       /* type of returned values      */
     Gint            *err_ind,   /* OUT error indicator          */
     Gline_bundle    *line_rep   /* OUT polyline 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 polymarker indices
     Gint          *num_elems_impl_list    /* OUT length of impl. list
extern void ginq_marker_rep(
     Gint              ws_id,        /* workstation identifier         */
     Gint              marker_ind,   /* polymarker index               */
     Ginq_type         type,         /* type of returned values        */
     Gint              *err_ind,     /* OUT error indicator            */
     Gmarker_bundle    *marker_rep   /* OUT polymarker 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_text_extent(
     Gint            ws_id,      /* workstation identifier        */
     const Gpoint    *pos,       /* text position                 */
     const char      *str,       /* text string                   */
     Gint            *err_ind,   /* OUT error indicator           */
     Gtext_extent    *extent      /* OUT concatenation point and
                                     text extent parallelogram     */);
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_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_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_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_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_ws_cat(
     Gint      ws_type,     /* workstation type            */
     Gint      *err_ind,    /* OUT error indicator         */
     Gws_cat   *cat         /* OUT workstation category    */);
extern void ginq_ws_class(
     Gint        ws_type,     /* workstation type        */
     Gint        *err_ind,    /* OUT error indicator     */
     Gws_class   *class       /* OUT workstation class   */);
extern void ginq_disp_space_size(
     Gint                ws_type,      /* workstation type            */
     Gint                *err_ind,     /* OUT error indicator         */
     Gdisp_space_size    *disp_size    /* OUT display [space] size    */);
extern void ginq_dyn_mod_ws_attrs(
     Gint                 ws_type,     /* workstation 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_type,        /* workstation 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_line_facs(
     Gint          ws_type,               /* workstation 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 polyline facilities         */
     Gint          *num_elems_impl_list   /* OUT length of linetype
                                             list in impl.                   */);
extern void ginq_pred_line_rep(
     Gint            ws_type,     /* workstation type              */
     Gint            ind,         /* predefined index              */
     Gint            *err_ind,    /* OUT error indicator           */
     Gline_bundle    *line_rep    /* OUT predefined polyline rep.  */);
extern void ginq_marker_facs(
     Gint            ws_type,               /* workstation 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 polymarker facilities     */
     Gint            *num_elems_impl_list   /* OUT length of marker type
                                               list in impl.                 */);
extern void ginq_pred_marker_rep(
     Gint              ws_type,       /* workstation type                */
     Gint              ind,           /* predefined index                */
     Gint              *err_ind,      /* OUT error indicator             */
     Gmarker_bundle    *marker_rep    /* OUT predefined polymarker rep.  */);
extern void ginq_text_facs(
     Gint          ws_type,               /* workstation 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_type,     /* workstation type          */
     Gint            ind,         /* predefined index          */
     Gint            *err_ind,    /* OUT error indicator       */
     Gtext_bundle    *text_rep    /* OUT predefined text rep.  */);
extern void ginq_fill_facs(
     Gint          ws_type,               /* workstation 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_type,     /* workstation type                */
     Gint            ind,         /* predefined index                */
     Gint            *err_ind,    /* OUT error indicator             */
     Gfill_bundle    *fill_rep    /* OUT predefined fill area rep.   */);
extern void ginq_pat_facs(
     Gint    ws_type,          /* workstation 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_type,     /* workstation 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_type,      /* workstation type      */
     Gint          *err_ind,     /* OUT error indicator   */
     Gcolr_facs    *colr_facs    /* OUT colour facilities */);
extern void ginq_pred_colr_rep(
     Gint         ws_type,     /* workstation 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_type,                /* workstation 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_gdp(
     Gint      ws_type,     /* workstation 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_type,     /* workstation 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_type,          /* workstation 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_type,     /* workstation type                    */
     Gint                  *err_ind,    /* OUT error indicator                 */
     Gdyn_mod_seg_attrs    *dyn_mod     /* OUT dynamic modification of
                                           segment attributess                 */);
extern void ginq_num_avail_in(
     Gint      ws_type,     /* workstation type           */
     Gint      *err_ind,    /* OUT error indicator        */
     Gnum_in   *num_in      /* OUT num. of input devices  */);
extern void ginq_def_loc_data(
     Gint        ws_type,      /* workstation 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_type,           /* workstation 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_type,       /* workstation 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_type,               /* workstation 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_type,        /* workstation 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_type,           /* workstation 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_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    *ws,                  /* OUT list of workstations     */
     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_attr   /* OUT segment attribute structure    */);
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 index 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         */);
extern void ginq_in_overf(
     Gint         *err_ind,     /* OUT error indicator                */
     Gint         *ws_id,       /* OUT workstation identifier         */
     Gin_class    *in_class,    /* OUT input class                    */
     Gint         *in_num       /* OUT logical input device number    */);
/*
 *      A.4.9  Utility Functions
 */
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(
     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 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 pointer to allocated storage     */);
extern void gdel_store(
     Gint      *err_ind,    /* OUT error indicator              */
     Gstore    *store       /* IN/OUT storage to be deleted     */);
/*
 *      A.4.10  Error Handling Functions
 */
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                          */);
```

Annex B
(informative)

Sample Programs

B.1 STAR Program

```
/*      Program:        STAR
 *      File:           star.c          Version 4.4
 *      Date:           6/11/91         16:35:11
 *
 *      Description
 *
 *      This program draws a yellow star on a blue background and writes
 *      The title "STAR" in white under the star.
 *
 *      Conformance
 *
 *      GKS level 0a
 */

/*
 *      Include files
 */
#include "gks.h"                        /* GKS definitions */
/*
 *      Local definitions (implementation and site dependent)
 */
#define GKS_ERR         "gks.err"       /* Name of the GKS error file */
#define DEF_BUF         GDEF_MEM_SIZE   /* Let GKS choose buffer size */
#define CONN_ID         "mo_star"       /* Connection identifier */
#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              ((double) 3.14159265)       /* The well-known number */
#define LARGE_TEXT      ((Gfloat) 0.15)             /* Size of large characters */

/*
 *      State Variables:
 *
 *                      These variables are used to set the
 *                      corresponding GKS state
 */

static Gasfs            asf = {          /* Aspect Source Flags */
```

```
        GASF_INDIV, GASF_INDIV, GASF_INDIV,
        GASF_INDIV, GASF_INDIV, GASF_INDIV,
        GASF_INDIV, GASF_INDIV, GASF_INDIV, GASF_INDIV,
        GASF_INDIV, GASF_INDIV, GASF_INDIV};

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 */
        double sin(), cos();        /* the trigonometric functions */
        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 = (Gpoint *)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,DEF_BUF);
            gset_asfs(&asf);
            gopen_ws(MY_WS,(void *)CONN_ID,WS_TYPE);
            gactivate_ws(MY_WS);

/*
 *          Centre the window about the origin
 */
            gset_win(TRAN_NUM,&win);
            gsel_norm_tran(TRAN_NUM);
/*
 *          Define the colours used
 */
            for (colr_ind=0, colrptr = colr; colr_ind<3; colr_ind++, colrptr++){
                        gset_colr_rep(MY_WS,colr_ind,colrptr);
            }
/*
 *          Fill the star with solid yellow
 */
            gset_fill_int_style(GSTYLE_SOLID);
            gset_fill_colr_ind(YELLOW);
/*
 *          Draw the star
 */
            gfill_area(&star);
/*
 *          Select large characters centred under the star
 */
            gset_char_ht(LARGE_TEXT);
            gset_text_align(&ct_half);
            gset_text_colr_ind(WHITE);
/*
 *          Draw the title
 */
            gtext(&title_pos, title);
/*
 *          Close everything
 */
            gdeactivate_ws(MY_WS);
            gclose_ws(MY_WS);
            gclose_gks();
            exit(GE_NO_ERR);
}
```

B.2 IRON Program

```
/*      Program:        IRON
 *
 *      File:           iron.c          Version 4.6
 *      Date:           9/5/91          16:03:48
 *
 *      Description
 *
 *      This program draws a horizontal bar chart illustrating costs
 *      within the iron industry. The user can select the data to be
 *      displayed using a GKS choice device. The plot is adapted from
 *      Scientific America, May 1984, page 139.
 *
 *      Conformance
 *
 *      GKS level 0b
 *
 *      Choice devices shall support prompt and echo type 3;
 */
/*
 *      Include files
 */
#include "gks.h"                        /* GKS definitions */

/*
 *      Local definitions (implementation and site dependent)
 */
#define GKS_ERR         "gks.err"       /* Name of the GKS error file */
#define DEF_BUF         GDEF_MEM_SIZE   /* Let GKS choose buffer size */
#define CONN_ID         "/tmp/tty"      /* Connection identifier */
#define WS_TYPE         (20)            /* Workstation type */
/*
 *      Convenient definitions
 */
#define MY_WS           (1)             /* The workstation we'll be using */
#define TRAN_NUM        (1)             /* Normalization transformation  number */
#define DEF_BUF         GDEF_MEM_SIZE   /* let GKS choose buffer size */
#define CHOICE_DEV      (1)             /* the choice device number */
#define CHOICE_PET      (3)             /* the choice prompt/echo type */
#define GREEN           (1)
#define RED             (2)

#define NUM_COUNTRIES   (3)
#define NUM_DATA        (6)
typedef struct {
        char            *name;
        Gfloat          data[2][NUM_DATA];
} Data_set;

void bar( Gfloat length, Gfloat  pos);
void ticks( Gfloat tstart, Gfloat tend,
        const Gtext_align            *alignment);
void border( const char              *nation);
void draw( const Data_set            *nation);

Data_set data_base[] = {
        {"United States",
                       {{69.0, 50.0, 15.0, 53.0, 57.0, 150.0},
                        {72.0, 50.0, 103.0, 0.0, 0.0, 56.0}}
        },
        {"West Germany",
                       {{65.0, 42.0, 3.0, 89.0, 52.0, 93.0},
                        {70.0, 53.0, 102.0, 0.0, 0.0, 49.0}}
        },
        {"Japan",
                       {{65.0, 47.0, 2.0, 60.0, 52.0, 55.0},
                        {70.0, 57.0, 105.0, 0.0, 0.0, 41.0}}
        }
};
char    *labels[NUM_DATA] = {
        "Labour",
        "Iron Ore",
        "Coke or Coal",
        "Purchased Scrap",
        "Other Costs",
        "Other Energy"
};

int main()
{
/*
 *      State Variables:
 *
 *                     These variables are used to set the
 *                     corresponding GKS state
 */
        static Glimit  win = {-115.0, 160., -2.0, 14.0};
        Gcolr_rep red,green ;
        static Gasfs   asf = {
                       GASF_INDIV, GASF_INDIV, GASF_INDIV,        /* polyline */
                       GASF_INDIV, GASF_INDIV, GASF_INDIV,        /* polymarker */
                       GASF_INDIV, GASF_INDIV, GASF_INDIV, GASF_INDIV, /* text */
                       GASF_BUNDLED, GASF_BUNDLED, GASF_INDIV     /* fill area */
        };
        Gint           err_ind;
        Gint           i;
        Gop_mode       op_mode;                  /* operating mode */
        Gecho_switch   echo_switch;              /* echo switch */
        Gin_status     init_choice_status;       /* initial choice status */
        Gint           init_choice;              /* initial choice value */
        Gin_status     choice_status;            /* choice status */
        Gint           choice;                   /* choice value */
        Gpoint_list    init_value;               /* initial value */
        Gint            pet;                        /* prompt/echo type */
        Glimit          echo_area;                  /* echo area */
        Gstore          store;                      /* storage for choice data record
        Gchoice_data    *choice_data;               /* choice data record */
        char            *prompts[NUM_COUNTRIES + 1];
                                                    /* prompts for choice */


/*
 *      open GKS and open/activate a ws.
 */
        gopen_gks(GKS_ERR,DEF_BUF);
        gopen_ws(MY_WS,(void *)CONN_ID,WS_TYPE);
        gactivate_ws(MY_WS);

/*
 *      specify the window onto chart
 */
        gset_win(TRAN_NUM,&win);
        gsel_norm_tran(TRAN_NUM);
/*
 *      define the colours used
 */
        green.rgb.red = green.rgb.blue = 0.0;
        green.rgb.green = 1.0;
        red.rgb.green = red.rgb.blue = 0.0;
        red.rgb.red = 1.0;
        gset_colr_rep(MY_WS,GREEN,&green);
        gset_colr_rep(MY_WS,RED,&red);
/*
 *      use the individual attributes, except the fill area
 *      interior style and style index
 */
        gset_asfs(&asf);

        gcreate_store(&err_ind, &store);

        ginq_choice_st( MY_WS, CHOICE_DEV, store, &err_ind,
                        &op_mode, &echo_switch, &init_choice_status,
                        &init_choice, &pet, &echo_area, &choice_data );
        if (err_ind != GE_NO_ERR) exit(err_ind);
        choice_data->pets.pet_r3.num_strings = NUM_COUNTRIES + 1;
        for (i=0; i<NUM_COUNTRIES; i++) {
                        prompts[i] = data_base[i].name;
        }
        prompts[NUM_COUNTRIES] = "EXIT";
        choice_data->pets.pet_r3.strings = prompts;
        ginit_choice(MY_WS, CHOICE_DEV,             /* set up this device */
                        init_choice_status,         /* initial choice status */
                        init_choice,                /* initial choice */
                        CHOICE_PET,                 /* prompt/echo type */
                        &echo_area,                 /* echo area */
                        choice_data);               /* choice data record */
```

```
/*
 *      get user's choice and disply data until told to quit
 */
        for (;;) {
                        greq_choice(MY_WS, CHOICE_DEV, &choice_status,
                        &choice);
                        if (choice_status != GIN_STATUS_OK /* choice failed */ ||
                            choice > NUM_COUNTRIES /* user selected exit */)
                            break;
                        draw(&data_base[choice-1]);
        }
/*
 *      delete the store
 */
        gdel_store(&err_ind,&store);
/*
 *      deactivate and close the ws and close GKS
 */
        gdeactivate_ws(MY_WS);
        gclose_ws(MY_WS);
        gclose_gks();
}
void draw( const Data_set              *nation)
/*
 *      draw the border of the graph followed by the two sets of data
 *      associated with this country
 */
{
        int             i, colr_ind;
        Gfloat          offset;

        border(nation->name);

        /* draw the data bars */
        for (colr_ind=GREEN, offset = 1.6;
                        colr_ind<=RED; colr_ind++, offset -=0.8)
        {
                        gset_fill_colr_ind(colr_ind);
                        gset_text_colr_ind(colr_ind);
                        gset_fill_ind(colr_ind);
                        for (i =0; i<NUM_DATA; i++){
                                        bar((Gfloat)nation->data[colr_ind-1][i],
                                                        (Gfloat)(2.0*i+offset));
                        }
        }
}
void border( const char               *nation)
/*
 *      draw the border surrounding the data
 */
{
        static Gpoint corner[5] = {
```

```
                        {0.0, 0.0},
                        {150.0, 0.0},
                        {150.0, 12.0},
                        {0.0, 12.0},
                        {0.0, 0.0}
        };
        Gpoint_list box;
        static Gtext_align            ctr_bot = {GHOR_CTR, GVERT_BOTTOM};
        static Gtext_align            ctr_top = {GHOR_CTR, GVERT_TOP};
        static Gtext_align            left_half = {GHOR_LEFT, GVERT_HALF};

        static Gpoint pos = {-114.0, 0.0};
        static Gpoint prod_pos = {0.0, -2.0};
        static Gpoint nation_pos = {0.0, 14.0};
        int           i;
/*
 *      clear the screen
 */
        gclear_ws(MY_WS, GFLAG_COND);
/*
 *      draw the box surrounding the chart area
 */
        box.num_points = 5;
        box.points = corner;
        gpolyline(&box);
/*
 *      draw the labels centred on the bar and flush left
 */
        gset_text_align(&left_half);
        gset_char_ht(0.5);
        gset_text_colr_ind(GREEN);
        for (i=0; i<NUM_DATA; i++){
                        pos.y = 1.2 + i*2.0;
                        gtext(&pos,labels[i]);
        }
/*
 *      draw the top and bottom tick marks
 */

        ticks(12.0, 12.2, &ctr_bot);
        gset_text_colr_ind(RED);
        ticks(0.0, -0.2, &ctr_top);
/*
 *      draw the title text in bigger characters
 */

        gset_text_colr_ind(GREEN);
        gtext(&prod_pos,"Production Cost");
        gset_char_ht(0.7);
        gset_text_align(&ctr_top);
        gtext(&nation_pos, nation);
}
```

```
void ticks( Gfloat tstart, Gfloat tend,
        const Gtext_align            *alignment)
/*
 *      draw the tickmarks along the edges
 */
{
#define NUM_TICKS     (4)
        char           label[5];
        Gpoint         xy[2];
        Gpoint_list    Pxy;
        int            i;
/*
 *      set up first tick mark
 */
        Pxy.num_points = 2;
        Pxy.points = xy;

        xy[0].y = tstart;
        xy[1].y = tend;
/*
 *      align above and below tick mark
 */
        gset_text_align(alignment);
/*
 *      draw the ticks; add a label to the end of each
 */
        for (i=0; i<NUM_TICKS; i++){
                       xy[0].x = xy[1].x = 50.0*i;
                       gpolyline(&Pxy);
                       sprintf(label,"%d",(int)xy[0].x);
                       gtext(&xy[1], label);
        }
}
void bar( Gfloat length, Gfloat  pos)
/*
 *      draw a horizontal bar of the specified 'length' and at
 *      position 'pos'
 */
 {
        static Gtext_align           alignment = {GHOR_LEFT, GVERT_HALF};
        Gpoint         xy[4];
        Gpoint_list    Pxy;

        Pxy.num_points = 4;
        Pxy.points = xy;
        if (length <= 0.0) {
                       gset_text_align(&alignment);
                       xy[0].x = 0.0;
                       xy[0].y = (Gfloat)pos;
                       gtext(xy, "0");
        }
        else {
```

```
                        xy[0].x = xy[3].x = 0.0;
                        xy[1].x = xy[2].x = length;

                        xy[0].y = xy[1].y = pos + 0.4;
                        xy[2].y = xy[3].y = pos - 0.4;

                        gfill_area(&Pxy);
                }
}
```

B.3 MAP Program

```
/*      Program:        MAP
 *      File:           map.c           Version 4.5
 *      Date:           6/11/91         16:35:15
 *
 *      Description
 *
 *      This program reads a GKSM metafile to draw a map. The primitives
 *      in each country are in a separate segment. The user can use a pick
 *      device to select various countries. A sampled choice determines
 *      the action taken with the selected country.
 *      The choice assignments are:
 *
 *      1)              highlight the state;
 *      2)              turn off highlighting;
 *      3)              make the state visible;
 *      4)              make the state invisible;
 *      5)              exit.
 *
 *      Conformance
 *
 *      GKS level 1c
 *
 *      the implementation shall support at least one workstation
 *      of category OUTIN and one of category MI. The default choice
 *      device shall support at least five choices;
 */
/*
 *      Include files
 */
#include "gks.h"                        /* GKS definitions */

/*
 *      Local definitions (implementation and site dependent)
 */
#define GKS_ERR         "gks.err"       /* name of the GKS error file */
#define DEF_BUF         GDEF_MEM_SIZE   /* let GKS choose buffer size */
#define MY_WS           (1)             /* the workstation we'll be using */
#define CONN_ID         "/dev/tty"      /* connection identifier */
```

```
#define  WS_TYPE        (19)          /* workstation type */
#define  META_WS        (2)           /* metafile workstation to read countries */
#define  META_CONN      "countries"   /* metafile file name */
#define  META_TYPE      (5)           /* metafile input type */
#define  CHOICE_DEV     (1)           /* the choice device number */
#define  PICK_DEV       (1)           /* the pick device number */

int main()
{
        Gitem_data      item_data; /* GKSM item data */
        Gint            item_length, item_type;
        Gpick           pick_data;
        Gint            choice;
        Gin_status      choice_status, pick_status;

/*
 *      open GKS and open/activate a ws.
 */
        gopen_gks(GKS_ERR,DEF_BUF);
        gopen_ws(MY_WS, (void *)CONN_ID,WS_TYPE);
        gactivate_ws(MY_WS);
/*
 *      set the choice device in the sample mode
 */
        gset_choice_mode(MY_WS, CHOICE_DEV, GOP_SAMPLE, GSWITCH_NO_ECHO);

/*
 *      open the MI workstation
 */
        gopen_ws(META_WS, (void *) META_CONN, META_TYPE);
/*
 *      interpret items until EOF is read;
 */
        for (;;) {

                gget_item_type(META_WS, &item_type, &item_length);
                if (item_type == Gksm_end_item) {
                        break;          /* terminate the loop */
                }

                item_data.length = item_length;
                item_data.length = item_type;

                gread_item(META_WS, item_length, &item_data);
                ginterpret_item(item_type, item_length, &item_data);
        }
        gclose_ws(META_WS);
/*
 *      allow the user to select countries until EXIT is
 *      selected
 */
```

```
        for(;;) {
                        greq_pick(MY_WS, PICK_DEV, &pick_status, &pick_data);
                        if (pick_status != GIN_STATUS_OK) break;
/*
 *      see which choice is in effect
 */
                        gsample_choice(MY_WS, CHOICE_DEV, &choice_status, &choice);
                        switch (choice) {
                        case 1:        gset_highl(pick_data.seg_name, GSEG_HIGHL);
                                       break;
                        case 2:        gset_highl(pick_data.seg_name, GSEG_NORM);
                                       break;
                        case 3:        gset_vis(pick_data.seg_name, GSEG_INVIS);
                                       break;
                        case 4:        gset_vis(pick_data.seg_name, GSEG_VIS);
                                       break;
                        default:
                                       break;
                        }
        }
}
```

B.4 MANIPULATE Program

```
/*      Program:        MANIPULATE
 *      File:           manipulate.c    Version 4.4
 *      Date:           9/5/91          16:03:51
 *
 *      Description
 *
 *      The program allows the user to create an object and then
 *      manipulate the object by changing the segment transformation.
 *
 *      Conformance
 *
 *      GKS level 2b
 */
/*
 *      Include files
 */
#include "gks.h"                        /* GKS definitions */
/*
 *      Local definitions (implementation and site dependent)
 */
#define GKS_ERR         "gks.err"       /* name of the GKS error file */
#define DEF_BUF         GDEF_MEM_SIZE   /* let GKS choose buffer size */
#define DISPLAY         1               /* the workstation we'll be using */
#define DISPL_CONN      "/dev/win8"     /* connection identifier */
#define DISPL_TYPE      (19)            /* workstation type */
```

```
#define  PLOTTER        (2)            /* plotter workstation */
#define  PLOT_CONN      "plotter_file" /* plotter file name */
#define  PLOT_TYPE      (17)           /* plotter type */
#define  SEG_STORE      (3)            /* segment workstation */
#define  SEG_CONN       "\0"           /* segment file name */
#define  SEG_TYPE       (0)            /* segment type */
#define  CHOICE_DEV     (1)            /* the choice device number */
#define  PICK_DEV       (1)            /* the pick device number */
#define  STROKE_DEV     (1)            /* the stroke device number */
#define  VAL_DEV        (1)            /* the valuator device number */
#define  LOC_DEV        (1)            /* the locator device number */
#define  STROKE_DEV     (1)            /* the stroke device number */
#define  MY_FONT        (-201)         /* an impl. dep. text font */

typedef  enum {
         FALSE,
         TRUE
} Bool;
Gtran_matrix           matrix;         /* global transformation matrix */

void init_gks(void)     /* initialize GKS and workstations */
{
         gopen_gks(GKS_ERR,DEF_BUF);
         gopen_ws(DISPLAY, (void *)DISPL_CONN,DISPL_TYPE);
         gactivate_ws(DISPLAY);
         gopen_ws(SEG_STORE, (void *)SEG_CONN,SEG_TYPE);
         gactivate_ws(SEG_STORE);
         gopen_ws(PLOTTER, (void *)PLOT_CONN,PLOT_TYPE);
}

void shut_all(void)     /* shut down GKS and close workstations */
{
         gdeactivate_ws(DISPLAY);
         gdeactivate_ws(SEG_STORE);
         gclose_ws(PLOTTER);
         gclose_ws(SEG_STORE);
         gclose_ws(DISPLAY);
         gclose_gks();
}
void set_up_tran(void)
{
         static Glimit  win = {0.0, 100.0, 0.0, 100.0};
         static Glimit  vp = {0.05, 0.95, 0.05, 0.95};

/*
 *       set window, viewport , and viewport input priority
 */
         gset_win(1, &win);
         gset_vp(1, &vp);

         gset_vp_in_pri(1, 0, GPRI_HIGHER);
}
```