Information Technology – Computer graphics and image processing
Graphical Kernel System – (GKS)

Language Bindings Part 5:
Fortran 90

Contents

Foreword

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

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

International Standard ISO/IEC 8651-1 was prepared by Joint Technical Committee ISO/IEC JTC1, Information technology.

ISO/IEC 8651 consists of the following parts, under the general title

Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) Language Bindings -

- Part 1: FORTRAN 77
- Part 2: PASCAL
- Part 3: ADA
- Part 4: C
- Part 5: Fortran 90

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

Introduction

The Graphical Kernel System (GKS), the functional description of which is given in ISO/IEC 7942-1:1994(E) , 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 7942-1:1994(E) is to define a standard binding for the Fortran 90 computer programming language.

ISO/IEC 8651-1 defines a language binding to the functionality of ISO 7942:1985. The set of functions in this Part extends ISO/IEC 8651-1 to provide access to all facilities of ISO/IEC 7942:1994.

It is anticipated that ISO 8651-1 will be considered for deletion at the next edition of ISO/IEC 8651.

1 Scope

This International Standard specifies a set of functions for computer graphics programming, the Graphical Kernel System (GKS). It provides functions for two dimensional graphical output, the storage and dynamic modification of pictures, and operator input. GKS functions and data types are specified independently of programming languages.

GKS establishes a system for device independent graphics programming by separating picture composition and interaction from the realization of the pictures on a specific output device and the input devices used by the operator.

This International Standard is applicable to a wide range of applications that produce two dimensional pictures on vector or raster graphical devices in monochrome and colour. Operator interaction is allowed with these pictures.

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 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.

References

[1] ISO/IEC 7942-1:1994(E) : Information technology
Computer graphics and image processing — Graphical Kernel System (GKS) functional description

[2] ISO/IEC 1539 : 1991 Fortran 90

3 The Fortran 90 Language Binding of GKS

3.1 Specification

The GKS language binding interface for ISO Fortran 90, see [2], shall be described as in clauses 3, 4, 5, 6, 7, 8, and 9.

3.2 Mapping of GKS Functions to Fortran 90 subroutine names

The function names of GKS are all mapped to Fortran subroutine names which start with the letter `g`. Any spaces in the subroutine name are replaced by underscores.

The remaining letters after the first one are obtained by deriving a unique acronym from the words of the function name; e.g., Primitive becomes "`prim`", Workstation becomes "`ws`". Hence, the Fortran subroutine name of GKS function Activate Workstation is `gactivate_ws`. For a list of all abbreviations, see clause 4.

3.3 Parameters

In general, the order of GKS function parameters is preserved.

In some subroutines, an argument that is an abstract data type has been used to define a generic interface.

This is described in more detail in clause 11.

3.4 Derived Data Types

A number of derived data types have been defined for the functions unique to ISO/IEC 7942-1:1994(E) . They are listed in clause 6.

3.5 Error Handling

There are two error routines in every GKS system, named `gerr_log` and `gerr_hand`. The user may replace the latter with their own subroutine using the same name, `gerr_hand`, and calling sequence. Furthermore, this user-defined error routine may call the system-defined error logging procedure `gerr_log`

4 Tables

4.1 Abbreviation Policy in Construction of Identifiers

1. The plural of an expression is constructed by adding an “s” after the abbreviation. So, for example, primitive is reduced to “prim” and primitives becomes “prims”. If an expression is reduced to NULL then so is its plural.

2. Digits are prefixed by an underscore “_”.

3. Construction of identifiers :

   - Function names — “`g`” followed by their abbreviated name all in lowercase.
   - Data types — “`G`” followed by their abbreviated name.
   - Fields of data types — redundant words (words in the field which are identical to those in the data type name) are omitted, if the context allows this. For example the colour index field in `Gcurve_bun` is abbreviated to `colr_ind` but there are two colour index fields in `Garea_bun` for both edges and interior. Therefore in `Garea_bun` the fields are called `int_colr_ind` and `edge_colr_ind`
   - Enumerated types — “`G`”, followed by their abbreviated name, ending with “`_id`”.

The following table lists the expressions to be abbreviated along with their resultant abbreviation. The word NULL indicates that the expression should be deleted when the name is being formed. Any expressions not included in this list should remain unchanged and any spaces are converted to underscores “_”.

4.2 List of Abbreviations

accumulate	→	accum
alignment	→	align
along	→	NULL
and	→	NULL
approximation	→	approx
archive	→	ar
argument	→	arg
associate(d)	→	assoc
attribute source flag	→	asf
attribute	→	attr
at	→	NULL
availability	→	avail
available	→	avail
boundary	→	bndry
bundle(d)	→	bun
centre	→	ctr
character	→	char
circular	→	circ
CIE L*u*v* (colour model)	→	cieluv
classification	→	class
clipping	→	clip
closed path closed	→	cpcl
closed path open	→	cpop

colour	→	colr
composite	→	comp
contour	→	cont
control	→	ctrl
convert	→	conv
conversion	→	conv
coordinate	→	coord
criteria	→	crit
criterion	→	crit
current	→	cur
dashed	→	dash
define	→	def
delete	→	del
deletion	→	del
description	→	desc
design primitive	→	design
detectability	→	det
detectable	→	det
device coordinate(s)	→	dc
device	→	dev
dimension	→	dim
direction	→	dir
display	→	disp
dotted	→	dot
element	→	elem
ellipse	→	ell
error state list	→	esl
error	→	err
evaluate	→	eval
expansion	→	expan
factor(s)	→	NULL
fill area	→	fill
from	→	NULL
function	→	func
generalized drawing primitive	→	gdp
generic	→	gen
gks closed	→	gkcl
gks description table	→	gdt
gks open	→	gkop
handling	→	hand
highlighted	→	highl
highlighting	→	highl
homogeneous	→	homo
horizontal	→	hor
hue lightness saturation (colour model)	→	hls
hue saturation value (colour modei	→	hsv
hyperbolic	→	hyp
identifier	→	id
index	→	ind
indicator	→	ind
indirect	→	ind
individual	→	indiv
information	→	info
initialise	→	init
initial	→	init
input	→	in
inquire	→	inq
instance	→	inst
integer	→	int

interior	→	int
interpret	→	intr
invalid	→	inval
invisible	→	invis
in	→	NULL
ISO9541	→	NULL
length	→	len
local	→	loc
locator	→	loc
logging	→	log
logical device coordinates	→	ldc
logical	→	NULL
luminance chromaticity coefficients	→	lcc
mapped	→	map
mapping	→	map
maximum	→	max
metafile	→	mf
minimum	→	min
monochrome	→	mono
nominal	→	nom
normai	→	norm
normalization	→	norm
normalized device coordinate(s)	→	ndc
number	→	num
of	→	NULL
on	→	NULL
operating	→	op
ordinate	→	ord
orientation	→	ori
output	→	out
overflow	→	overf
parabolic	→	par
parameter	→	param
pattern part closed	→	patpcl
pattern part open	→	patpop
pattern	→	pat
picture part closed	→	ppcl
picture part open	→	ppop
picture	→	pic
pointer	→	ptr
polyline	→	line
polymarker	→	marker
position	→	pos
postconcatenate	→	post
preconcatenate	→	pre
precision	→	prec
predefined	→	pred
primitive	→	prim
priority	→	pri
process	→	proc
prompt and echo type	→	pet
queue	→	NULL
radius	→	rad
range	→	ran
realized	→	real
record	→	rec
rectangle	→	rect
red green blue (colour model)	→	rgb
reference	→	ref

reject	→	rej
replication	→	replic
representation	→	rep
request(ed)	→	req
reset	→	res
retrieve	→	ret
section	→	sec
segment state list	→	ssl
segment store closed	→	sscl
segment store open	→	ssop
segment	→	seg
select(or)	→	sel
selection	→	sel
sequence	→	seq
source	→	src
specification	→	specif
specific	→	spec
specified	→	specif
state	→	st
state list	→	sl
stencil part closed	→	stcl
stencil part open	→	stop
stencil state list	→	stsl
subtract	→	sub
table	→	tab
technique	→	tech
tiling closed	→	tlcl
tiling open	→	tlop
to	→	NULL
transform(ation)	→	tran
undetectable	→	undet
used	→	NULL
valuator	→	val
value	→	val
vector	→	vec
vertical	→	vert
viewport	→	vp
visibility	→	vis
visible	→	vis
visual	→	vis
window	→	win
workstation description table	→	wdt
workstation	→	ws
world coordinate	→	wc

5 Translation of abstract GKS data types into Fortran 90 constructs

The following sections illustrate the general rules used for generating the appropriate Fortran 90 construct from the GKS abstract data type.

5.1 Enumeration Types

RouteDir == NDC | BACKDROP

All enumeration types are prefixed by a "`G`" and end with "`_id`".

```
TYPE Groute_dir_id
  INTEGER :: i
END TYPE Groute_dir_id

TYPE(Groute_dir_id), PARAMETER :: &
Groute_dir_ndc      = Groute_dir_id(0), &
Groute_dir_backdrop = Groute_dir_id(1)
```

5.2 Powerset

NameSet == P GenName

```
  TYPE Gname_set
    TYPE (Ggen_name) pointer, dimension(:) :: name_list
  END TYPE Gname_set
```

5.3 Cartesian Product

FontPrec == FontInd $\times$ Prec

```
  TYPE Gfont_prec
    INTEGER :: font_ind
    INTEGER :: prec
  END TYPE Gfont_prec
```

5.4 Set Extension

TypeName == $\{declarations \mid predicate\ o\ expression\}$

Types which consist of subranges of other types are not separately defined.

5.5 Matrices

Matrix23 == $\{i, j : \mid N_1, m : R \mid 1 <= i <= 2, 1 <= j <= 3\ o\ (i, j) \rightarrow m\}$

```
  TYPE Gmatrix_23
      REAL :: coeff(2,3)
  END TYPE Gmatrix_23
```

5.6 Function Union Types

PrimAttrValue	==	pickidentifier <<PickId>> |
		nameset <<NameSet>> |
		textheight <<TextHt>> |
		...

Fortran 90 has no mechanism to restrict which members of a `TYPE` you can access so we need to introduce an enumerated type to indicate which member is being used:

PrimAttrName == PICKIDENTIFIER | NAMESET | TEXTHEIGHT | ...

```
TYPE Gprim_attr_id
  INTEGER :: i
END TYPE Gprim_attr_id

TYPE(Gprim_attr_id), PARAMETER :: &
Gprim_attr_pickid             = Gprim_attr_id(0), &
Gprim_attr_nameset            = Gprim_attr_id(1), &
Gprim_attr_scissorset         = Gprim_attr_id(2), &
 ...

  TYPE Gprim_attr_val
    TYPE(Gprim_attr_id) :: prim_attr
    INTEGER             :: pick_id
    TYPE(Gname_set)     :: name_set
    TYPE(Gscissor_set)  :: scissor_set
      ...
  END TYPE Gprim_attr_val
```

If the data type is a parameter to a GKS function then the binding makes use of the ability to define a GENERIC interface to this function. The subroutine which is called is based on the datatypes of the arguments to the subroutine.

Set Primitive Attribute

```
INTERFACE gset_prim_attr

 SUBROUTINE gset_pa_pick_id (pick_id)
  TYPE(Gpick_id), INTENT(IN)                    :: pick_id

 SUBROUTINE gset_pa_name_set (name_set)
  TYPE(Gname_set), INTENT(IN)                   :: name_set

 SUBROUTINE gset_pa_text_height (text_height)
  TYPE(Gtext_height), INTENT(IN)                :: text_height
```

An application which makes a call to this subroutine would use the generic name `gset_prim_attr`.

```
  call gset_prim_attr (Gtext_height(0.5))
```

The datatype would indicate that the subroutine `gset_pa_text_height` should be invoked.

5.7 Partial Functions

A partial function is a mapping from a set of integers. For example, a value of type CurveBunTab is a mapping to values of type CurveBun, held as an array each of whose elements holds an integer and the corresponding value of type CurveBun. For some values of CurveInd there may be no corresponding CurveBun value.

CurveBunTab == CurveInd $\mapsto$ CurveBun

```
  TYPE Gcurve_bun_elem
    INTEGER          :: curve_ind
    TYPE(Gcurve_bun) :: curve_bun
  END TYPE Gcurve_bun_elem

  TYPE(Gcurve_bun_elem), pointer, dimension(:) :: Gcurve_bun_tab
```

5.8 Sequence Types

SeqMeasureId == seq MeasureId

```
  TYPE Gseq_measure_id
    INTEGER, pointer, dimension(:) :: measure_id
  END TYPE Gseq_measure_id
```

5.9 SelectCrit Type

The SelectCrit data type in GKS requires special treatment when binding to Fortran 90 and does not follow the rules set out in this clause. This includes the creation of an additional type `Gdyadic_args` to allow the definition of selection criteria. The type is recursive to allow the expression of selection criteria to be composed from other sub expressions. For example, consider the following expression where RED, VISIBLE and HIGHLIGHT are all namesets:

```
CONTAINS(RED) AND NOT( ISIN(VISIBLE) OR ISIN(HIGHLIGHT) )
```

```
  TYPE Gsel_crit
    TYPE(Gsel_op_id)            :: sel_op
    TYPE(Gname_set)             :: name_set
    TYPE(Gsel_crit), POINTER    :: monadic_arg
    TYPE(Gdyadic_args), POINTER :: dyadic_args
  END TYPE Gsel_crit

  TYPE Gdyadic_args
    TYPE(Gsel_crit), POINTER :: arg1
    TYPE(Gsel_crit), POINTER :: arg2
  END TYPE Gdyadic_args
```

6 Data types

In ISO/IEC 7942-1:1994(E) there are many different abstract data types that are used to specify the parameters to each function. The following shows the correspondence between the types used in ISO/IEC 7942-1:1994(E) and their realization in a Fortran 90 implementation using intrinsic and derived data types.

6.1 Data Types for GKS Control

GKS Data Type		Fortran binding data type and attributes
ErrFile	Error file	`CHARACTER, POINTER`
EscapeFuncId	Escape function identifier	`INTEGER`
EscapeInData	Escape input data record	`CHARACTER, POINTER`
EscapeOutData	Escape output data record	`CHARACTER, POINTER`
ErrNum	Error number	`INTEGER`
ErrSt	Error status	`Gerr_st_id`
FuncName	Function name	`Gfunc_name_id`
RouteDir	Route Direction	`Groute_dir_id`

6.2 Data Types for Output Primitives

GKS Data Type		Fortran binding data type and attributes
WCPoint	World Coordinate Point	`Gwc_point`
WCOrd	Ordinate value in WC	`REAL`
WCOrdPos	Ordinate value in WC	`REAL`
HWCPoint	Homogenous world coordinate point	`Ghomo_wc_point`
CharString	Character string	`CHARACTER, POINTER, DIMENSION(:)`
ColrArraySpec	Colour array specifier	`Gcolr_array_spec`
ControlPoints	Control points	`Gctrl_points`
Knots	Knots	`Gknots`
Matrix23	2 x 3 matrix	`Gmatrix_23`
Matrix33	3 x 3 matrix	`Gmatrix_33`
ParamRan	Parameter Range	`Gparam_ran`
NURB	NURB record	`Gnurb`
SplineOrder	Spline Order	`INTEGER`
ClosedNURB	Closed NURB record	`Gnurb`
ConicSection	Conic section	`Gconic_sec`
EllipticDisc	Elliptic disc	`Gmatrix_33`
TilingName	Tiling name	`INTEGER`
TilingOrigin	Tiling origin	`Gwc_point`
TilingTran	Tiling transformation	`Gmatrix_23`
PrimParam	Output primitive parameters	`Gprim_param`
SenseFlag	Sense flag	`Gsense_flag_id`
StencilName	Stencil name	`INTEGER`
StencilOrigin	Stencil origin	`Gwc_point`
StencilTran	Stencil transformation	`Gmatrix_23`
GDPData	Generalized Drawing Primitive data record	`Ggdp_data`
GDPId	Generalized Drawing Primitive identifier	`INTEGER`

The following auxiliary data types are defined:

Data Type		Fortran binding data type
indirectarray	array of colour indicies	Gind_array
rgbarray	array of rgb	Grgb_array
cieluvarray	array of cieluv	Gcieluv_array
hsvarray	array of hsv	Ghsv_array
hlsarray	array of hls	Ghls_array
hcp	sequence of homogeneous WC points	Ghcp
cp	sequence WC points	Gcp
setofpolyline	Set of polyline	Gset_line
setofnurb	Set of NURB	Gset_nurb
setofconicsection	Set of conic section	Gset_conic_sec
polymarker	Polymarker	Gmarker
setoffillarea	Set of Fillarea	Gset_fill
setofellipticsector	Set of elliptic sector	Gset_ellip_sec
setofellipticsegment	Set of elliptic segment	Gset_ellip_seg
setofellipticdisc	Set of elliptic disc	Gset_ellip_disc
setofclosednurb	Set of closed NURB	Gset_closed_nurb
text	Text	Gtext
cellarray	Cell array	Gcell_array
design	Design	Gdesign
gdp	Generalized drawing primitive	Ggdp
seq WCPoint	Sequence of WC points	Gseq_wc_point

The following list defines the derived Fortran 90 data types mentioned above:

Gwc_point

```
TYPE Gwc_point
    REAL :: x
    REAL :: y
END TYPE Gwc_point
```

Gseq_wc_point

```
TYPE Gseq_wc_point
    TYPE(Gwc_point), POINTER, DIMENSION(:) :: point
END TYPE Gseq_wc_point
```

Ghomo_wc_point

```
TYPE Ghomo_wc_point
    REAL :: x
    REAL :: y
    REAL :: w
END TYPE Ghomo_wc_point
```

Gind_array

```
TYPE Gind_array
  INTEGER                         :: dx
  INTEGER                         :: dy
  INTEGER, POINTER, DIMENSION(:,:) :: ind
END TYPE Gind_array
```

Grgb_array

```
TYPE Grgb_array
  INTEGER                                 :: dx
  INTEGER                                 :: dy
  TYPE(Gcolr_rgb), POINTER, DIMENSION(:,:) :: rgb
END TYPE Grgb_array
```

Gcieluv_array

```
TYPE Gcieluv_array
  INTEGER                                    :: dx
  INTEGER                                    :: dy
  TYPE(Gcolr_cieluv), POINTER, DIMENSION(:,:) :: cieluv
END TYPE Gcieluv_array
```

Ghsv_array

```
TYPE Ghsv_array
  INTEGER                                 :: dx
  INTEGER                                 :: dy
  TYPE(Gcolr_hsv), POINTER, DIMENSION(:,:) :: hsv
END TYPE Ghsv_array
```

Ghls_array

```
TYPE Ghls_array
  INTEGER                                 :: dx
  INTEGER                                 :: dy
  TYPE(Gcolr_hls), POINTER, DIMENSION(:,:) :: hls
END TYPE Ghls_array
```

Ghcp

```
TYPE Ghcp
  TYPE(Ghomo_wc_point), POINTER, DIMENSION(:) :: point
END TYPE Ghcp
```

Gcp

```
TYPE Gcp
  TYPE(Gwc_point), POINTER, DIMENSION(:) :: point
END TYPE Gcp
```

Gmatrix_23

```
TYPE Gmatrix_23
    REAL :: coeff(2,3)
END TYPE Gmatrix_23
```

Gmatrix_33

```
TYPE Gmatrix_33
    REAL :: coeff(3,3)
END TYPE Gmatrix_33
```

Gset_line

```
TYPE Gset_line
  TYPE(Gseq_wc_point), POINTER, DIMENSION(:) :: seq
END TYPE Gset_line
```

Gset_nurb

```
TYPE Gset_nurb
  TYPE(Gnurb), POINTER, DIMENSION(:) :: nurb
END TYPE Gset_nurb
```

Gset_conic_sec

```
TYPE Gset_conic_sec
  TYPE(Gconic_sec), POINTER, DIMENSION(:) :: sec
END TYPE Gset_conic_sec
```

Gmarker

```
TYPE Gmarker
  TYPE(Gwc_point), POINTER, DIMENSION(:) :: point
END TYPE Gmarker
```

Gset_fill

```
TYPE Gset_fill
  TYPE(Gseq_wc_point), POINTER, DIMENSION(:) :: seqs
END TYPE Gset_fill
```

Gset_ellip_sec

```
TYPE Gset_ellip_sec
  TYPE(Gconic_sec), POINTER, DIMENSION(:) :: sec
END TYPE Gset_ellip_sec
```

Gset_ellip_seg

```
TYPE Gset_ellip_seg
  TYPE(Gconic_sec), POINTER, DIMENSION(:) :: seg
END TYPE Gset_ellip_seg
```

Gset_ellip_disc

```
TYPE Gset_ellip_disc
  TYPE(Gmatrix_33), POINTER, DIMENSION(:) :: disc
END TYPE Gset_ellip_disc
```

Gset_closed_nurb

```
TYPE Gset_closed_nurb
  TYPE(Gnurb), POINTER, DIMENSION(:) :: nurb
END TYPE Gset_closed_nurb
```

Gtext

```
TYPE Gtext
  TYPE(Gwc_point)    :: ref_point
  CHARACTER, POINTER :: charstring(:)
END TYPE Gtext
```

Gcolr_array_spec

```
TYPE Gcolr_array_spec
  TYPE(Gcolr_array_spec_id) :: colr_type
  TYPE(Gind_array)          :: ind_array
  TYPE(Grgb_array)          :: rgb_array
  TYPE(Gcieluv_array)       :: cieluv_array
  TYPE(Ghsv_array)          :: hsv_array
  TYPE(Ghls_array)          :: hls_array
END TYPE Gcolr_array_spec
```

Gcell_array

```
TYPE Gcell_array
  TYPE(Gwc_point)        :: p
  TYPE(Gwc_point)        :: q
  TYPE(Gcolr_array_spec) :: array_spec
END TYPE Gcell_array
```

Gdesign

```
TYPE Gdesign
  INTEGER          :: stencil_name
  TYPE(Gwc_point)  :: stencil_origin
  TYPE(Gmatrix_23) :: stencil_tran
  INTEGER          :: tiling_name
  TYPE(Gwc_point)  :: tiling_origin
  TYPE(Gmatrix_23) :: tiling_tran
END TYPE Gdesign
```

Ggdp_data

```
TYPE Ggdp_data
  CHARACTER, POINTER :: data_rec(:)
END TYPE Ggdp_data
```

Ggdp

```
TYPE Ggdp
  TYPE(Gwc_point), POINTER, DIMENSION(:) :: point
  INTEGER                                :: id
  TYPE(Ggdp_data)                        :: data
END TYPE Ggdp
```

Gctrl_points

```
TYPE Gctrl_points
  TYPE(Gctrl_points_id) :: point_type
  TYPE(Ghcp)            :: hcp
  TYPE(Gcp)             :: cp
END TYPE Gctrl_points
```

Gknots

```
TYPE Gknots
    REAL, POINTER, DIMENSION(:) :: knot
END TYPE Gknots
```

Gparam_ran

```
TYPE Gparam_ran
    REAL :: tmin
    REAL :: tmax
END TYPE Gparam_ran
```

Gnurb

```
TYPE Gnurb
    INTEGER             :: k
    TYPE(Gctrl_points)  :: cp
    TYPE(Gknots)        :: kn
    TYPE(Gparam_ran)    :: ran
END TYPE Gnurb
```

Gconic_sec

```
TYPE Gconic_sec
    TYPE(Gmatrix_33)       :: matrix
    TYPE(Gwc_point)        :: start
    TYPE(Gwc_point)        :: end
    TYPE(Gsense_flag_id)   :: sense_flag
END TYPE Gconic_sec
```

Gprim_param

```
TYPE Gprim_param
  TYPE(Gprim_param_id)   :: prim_type
  TYPE(Gset_line)        :: set_line
  TYPE(Gset_nurb)        :: set_nurb
  TYPE(Gset_conic_sec)   :: set_conic_sec
  TYPE(Gmarker)          :: marker
  TYPE(Gset_fill)        :: set_fill
  TYPE(Gset_ellip_sec)   :: set_ellip_sec
  TYPE(Gset_ellip_seg)   :: set_ellip_seg
  TYPE(Gset_ellip_disc)  :: set_ellip_disc
  TYPE(Gset_closed_nurb) :: set_closed_nurb
  TYPE(Gtext)            :: text
  TYPE(Gcell_array)      :: cell_array
  TYPE(Gdesign)          :: design
  TYPE(Ggdp)             :: gdp
END TYPE Gprim_param
```

6.3 Data Types for Output Attributes

GKS Data Type		Fortran binding data type and attributes
AreaAsfs	Area ASF's	`Garea_asfs`
AreaInd	Area index	`INTEGER`
AsfValue	ASF's value	`Gasf_val_id`
CharAsfs	Character ASFs	`Gchar_asfs`
CharExpan	Character expansion	`REAL`
CharInd	Character index	`INTEGER`
CharSpace	Character spacing	`REAL`
ClipInd	Clip indicator	`Gclip_ind_id`
ColrInd	Colour index	`INTEGER`
ColrSpec	Colour specifier	`Gcolr_spec`
CurveAsfs	Curve ASFs	`Gcurve_asfs`
CurveInd	Curve index	`INTEGER`
CurveType	Curve type	`INTEGER`
CurveWidth	Curve width	`REAL`
EdgeFlag	Edge flag	`Gedge_flag_id`
EdgeType	Edge type	`INTEGER`
EdgeWidth	Edge width	`REAL`
FontInd	Text Font index	`INTEGER`
FontPrec	Text Font and Precision	`Gfont_prec`
GenName	Generic name	`Ggen_name`
HorAlign	Horizontal alignment	`Ghor_align_id`
IntStyleInd	Interior Style Index	`INTEGER`
IntStyle	Interior style	`Gint_style_id`
MarkerAsfs	Polymarker Asf's	`Gmarker_asfs`
MarkerInd	Marker index	`INTEGER`
MarkerSize	Marker size	`REAL`
MarkerType	Marker type	`INTEGER`
Name	Name	`INTEGER`
NameSet	Nameset	`Gname_set`
NDCOrd	NDC ordinate	`REAL`
NDCRect	NDC rectangle	`Gndc_rect`
NDCRectSet	NDC rectangle set	`Gndc_rect_set`
PatSize	Pattern size	`Gpat_size`
PickId	Pick identifier	`INTEGER`
Prec	Precision	`Gprec_id`
Scissor	Scissor	`Gscissor`
ScissorId	Scissor identifier	`INTEGER`
ScissorIdSet	Set of scissor identifiers	`Gscissor_id_set`
ScissorSet	Set of scissor	`Gscissor_set`
SegName	Segment name	`INTEGER`
SheildInd	Sheild indicator	`Gshield_ind_id`
TextAlign	Text alignment	`Gtext_align`
TextHt	Text height	`REAL`
TextPath	Text path	`Gtext_path_id`
TextSkew	Text skew	`REAL`
TextUpVec	Text up vector	`Gtext_up_vec`
VertAlign	Vertical alignment	`Gvert_align_id`
PrimAttrVal	Output primitive attributes	`Gprim_attr_val`

The following auxiliary data types are defined:

Data Type		Fortran binding data type
ScissorSetElems	Scissor set elements	Gscissor_set_elem
pick_id	pick id	Gpick_id
curve_ind	curve ind	Gcurve_ind
curve_type	curve type	Gcurve_type
marker_ind	marker ind	Gmarker_ind
marker_type	marker type	Gmarker_type
area_ind	area ind	Garea_ind
int_style_ind	int style ind	Gint_style_ind
edge_type	edge type	Gedge_type
char_ind	char ind	Gchar_ind
text_height	text height	Gtext_height
text_skew	text skew	Gtext_skew
curve_width	curve width	Gcurve_width
edge_width	edge width	Gedge_width
char_expan	char expan	Gchar_expan
char_space	char space	Gchar_space
global_tran	global tran	Gglobal_tran
loc_tran	loc tran	Gloc_tran
curve_colr_spec	curve colr spec	Gcurve_colr_spec
marker_colr_spec	marker colr spec	Gmarker_colr_spec
int_colr_spec	int colr spec	Gint_colr_spec
edge_colr_spec	edge colr spec	Gedge_colr_spec
char_colr_spec	char colr spec	Gchar_colr_spec

The following list defines the derived Fortran 90 data types described above:

Garea_asfs

```
TYPE Garea_asfs
  TYPE(Gasf_val_id) :: int_style
  TYPE(Gasf_val_id) :: int_style_ind
  TYPE(Gasf_val_id) :: int_colr_spec
  TYPE(Gasf_val_id) :: edge_flag
  TYPE(Gasf_val_id) :: edgetype
  TYPE(Gasf_val_id) :: edgewidth
  TYPE(Gasf_val_id) :: edge_colr_spec
END TYPE Garea_asfs
```

Gchar_asfs

```
TYPE Gchar_asfs
  TYPE(Gasf_val_id) :: font_prec
  TYPE(Gasf_val_id) :: expan
  TYPE(Gasf_val_id) :: spacing
  TYPE(Gasf_val_id) :: colr_spec
END TYPE Gchar_asfs
```

Gcolr_cieluv

```
TYPE Gcolr_cieluv
  REAL :: u
  REAL :: v
  REAL :: Y
END TYPE Gcolr_cieluv
```

Gcolr_hls

```
TYPE Gcolr_hls
  REAL :: h
  REAL :: l
  REAL :: s
END TYPE Gcolr_hls
```

Gcolr_hsv

```
TYPE Gcolr_hsv
  REAL :: h
  REAL :: s
  REAL :: v
END TYPE Gcolr_hsv
```

Gcolr_rgb

```
TYPE Gcolr_rgb
  REAL :: r
  REAL :: g
  REAL :: b
END TYPE Gcolr_rgb
```

Gcolr_coord

```
TYPE Gcolr_coord
  TYPE(Gcolr_model_id) :: colr_model
  TYPE(Gcolr_rgb)      :: rgb
  TYPE(Gcolr_cieluv)   :: cieluv
  TYPE(Gcolr_hsv)      :: hsv
  TYPE(Gcolr_hls)      :: hls
END TYPE Gcolr_coord
```

Gcolr_char

```
TYPE Gcolr_char
  CHARACTER, POINTER :: data(:)
END TYPE Gcolr_char
```

Gcolr_spec

```
TYPE Gcolr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gcolr_spec
```

Gcurve_asfs

```
TYPE Gcurve_asfs
  TYPE(Gasf_val_id) :: type
  TYPE(Gasf_val_id) :: width
  TYPE(Gasf_val_id) :: colr_spec
END TYPE Gcurve_asfs
```

Gfont_prec

```
TYPE Gfont_prec
  INTEGER       :: font_ind
  TYPE(Gprec_id) :: prec
END TYPE Gfont_prec
```

Ggen_name

```
TYPE Ggen_name
  TYPE(Ggen_name_id) :: gen_name
  INTEGER            :: wsid
  INTEGER            :: seg_name
  INTEGER            :: name
END TYPE Ggen_name
```

Gmarker_asfs

```
TYPE Gmarker_asfs
  TYPE(Gasf_val_id) :: type
  TYPE(Gasf_val_id) :: size
  TYPE(Gasf_val_id) :: colr_spec
END TYPE Gmarker_asfs
```

Gname_set

```
TYPE Gname_set
  TYPE (Ggen_name), POINTER, DIMENSION(:) :: name
END TYPE Gname_set
```

Gndc_rect

```
TYPE Gndc_rect
  REAL :: xmin
  REAL :: xmax
  REAL :: ymin
  REAL :: ymax
END TYPE Gndc_rect
```

Gndc_rect_set

```
TYPE Gndc_rect_set
  TYPE(Gndc_rect), POINTER, DIMENSION(:) :: rect
END TYPE Gndc_rect_set
```

Gpat_size

```
TYPE Gpat_size
  REAL :: x
  REAL :: y
END TYPE Gpat_size
```

Gscissor

```
  TYPE Gscissor
    TYPE(Gclip_ind_id)     :: clip_ind
    TYPE(Gndc_rect_set)    :: clip_rect_set
    TYPE(Gshield_ind_id)   :: shield_ind
    TYPE(Gndc_rect_set)    :: shield_rect_set
  END TYPE Gscissor
```

Gscissor_id_set

```
  TYPE Gscissor_id_set
     INTEGER, POINTER, DIMENSION(:) :: id
  END TYPE Gscissor_id_set
```

Gscissor_set_elem

```
  TYPE Gscissor_set_elem
    INTEGER        :: scissor_id
    TYPE(Gscissor) :: scissor_bun
  END TYPE Gscissor_set_elem
```

Gscissor_set

```
  TYPE Gscissor_set
    TYPE (Gscissor_set_elem), POINTER, DIMENSION(:) :: ind
  END TYPE Gscissor_set
```

Gtext_align

```
  TYPE Gtext_align
    TYPE(Ghor_align_id)  :: hor_align
    TYPE(Gvert_align_id) :: vert_align
  END TYPE Gtext_align
```

Gtext_up_vec

```
  TYPE Gtext_up_vec
    REAL :: x
    REAL :: y
  END TYPE Gtext_up_vec
```

Gprim_attr_val

```
  TYPE Gprim_attr_val
    TYPE(Gprim_attr_id) :: prim_attr
    INTEGER             :: pick_id
    TYPE(Gname_set)     :: name_set
    TYPE(Gscissor_set)  :: scissor_set
    TYPE(Gmatrix_23)    :: global_tran
    TYPE(Gmatrix_23)    :: loc_tran
    TYPE(Gpat_size)     :: pat_size
    TYPE(Gwc_point)     :: pat_ref_pt
    REAL                :: text_height
    TYPE(Gtext_up_vec)  :: text_up_vec
    REAL                :: text_skew
```

```
    TYPE(Gtext_path_id) :: text_path
    TYPE(Gtext_align)   :: text_align
    INTEGER             :: curve_ind
    TYPE(Gcurve_asfs)   :: curve_asfs
    INTEGER             :: curve_type
    REAL                :: curve_width
    TYPE(Gcolr_spec)    :: curve_colr_spec
    INTEGER             :: marker_ind
    TYPE(Gmarker_asfs)  :: marker_asfs
    INTEGER             :: marker_type
    REAL                :: marker_size
    TYPE(Gcolr_spec)    :: marker_colr_spec
    INTEGER             :: area_ind
    TYPE(Garea_asfs)    :: area_asfs
    TYPE(Gint_style_id) :: int_style
    INTEGER             :: int_style_ind
    TYPE(Gcolr_spec)    :: int_colr_spec
    TYPE(Gedge_flag_id) :: edge_flag
    INTEGER             :: edge_type
    REAL                :: edge_width
    TYPE(Gcolr_spec)    :: edge_colr_spec
    INTEGER             :: char_ind
    TYPE(Gchar_asfs)    :: char_asfs
    TYPE(Gfont_prec)    :: char_font_prec
    REAL                :: char_expan
    REAL                :: char_space
    TYPE(Gcolr_spec)    :: char_colr_spec
  END TYPE Gprim_attr_val
```

Gpick_id

```
  TYPE Gpick_id
    INTEGER :: i
  END TYPE  Gpick_id
```

Gcurve_ind

```
  TYPE Gcurve_ind
    INTEGER :: i
  END TYPE  Gcurve_ind
```

Gcurve_type

```
  TYPE Gcurve_type
    INTEGER :: i
  END TYPE  Gcurve_type
```

Gmarker_ind

```
  TYPE Gmarker_ind
    INTEGER :: i
  END TYPE  Gmarker_ind
```

Gmarker_type

```
TYPE Gmarker_type
  INTEGER :: i
END TYPE  Gmarker_type
```

Garea_ind

```
TYPE Garea_ind
  INTEGER :: i
END TYPE  Garea_ind
```

Gint_style_ind

```
TYPE Gint_style_ind
  INTEGER :: i
END TYPE  Gint_style_ind
```

Gedge_type

```
TYPE Gedge_type
  INTEGER :: i
END TYPE  Gedge_type
```

Gchar_ind

```
TYPE Gchar_ind
  INTEGER :: i
END TYPE  Gchar_ind
```

Gmarker_size

```
TYPE  Gmarker_size
  REAL :: r
END TYPE Gmarker_size
```

Gtext_height

```
TYPE  Gtext_height
  REAL :: r
END TYPE Gtext_height
```

Gtext_skew

```
TYPE  Gtext_skew
  REAL :: r
END TYPE Gtext_skew
```

Gcurve_width

```
TYPE  Gcurve_width
  REAL :: r
END TYPE Gcurve_width
```

Gedge_width

```
TYPE Gedge_width
  REAL :: r
END TYPE Gedge_width
```

Gchar_expan

```
TYPE Gchar_expan
  REAL :: r
END TYPE Gchar_expan
```

Gchar_space

```
TYPE Gchar_space
  REAL :: r
END TYPE Gchar_space
```

Gmatrix_23_global

```
TYPE Gmatrix_23_global
    REAL :: coeff(2,3)
END TYPE Gmatrix_23_global
```

Gmatrix_23_local

```
TYPE Gmatrix_23_local
    REAL :: coeff(2,3)
END TYPE Gmatrix_23_local
```

Gcurve_colr_spec

```
TYPE Gcurve_colr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gcurve_colr_spec
```

Gmarker_colr_spec

```
TYPE Gmarker_colr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gmarker_colr_spec
```

Gint_colr_spec

```
TYPE Gint_colr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gint_colr_spec
```

Gedge_colr_spec

```
TYPE Gedge_colr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gedge_colr_spec
```

Gchar_colr_spec

```
TYPE Gchar_colr_spec
  TYPE(Gcolr_spec_id) :: colr_spec
  INTEGER             :: ind
  TYPE(Gcolr_rgb)     :: rgb
  TYPE(Gcolr_cieluv)  :: cieluv
  TYPE(Gcolr_hls)     :: hls
  TYPE(Gcolr_hsv)     :: hsv
END TYPE Gchar_colr_spec
```

6.4 Data Types for Transformations

GKS Data Type		Fortran binding data type and attributes
DCOrd	DC Ordinate	REAL
DCVp	DC Viewport	Gdc_vp
LDCWin	LDC window	Gldc_win
LDCOrd	LDC Ordinate	REAL
LDCRect	LDC rectangle	Gldc_rect
LDCRectSet	Set of LDC rectangle	Gldc_rect_set
LDCVp	LDC viewport	Gldc_vp
NDCWin	NDC window	Gndc_win
NDCVp	NDC viewport	Gndc_vp
NormTran	Normalization Transformation	Gnorm_tran
NormTranNum	Normalization transformation number	INTEGER
NormTranNum1	Normalization transformation number	INTEGER
Pri	Priority	Gpri_id
ScissorMode	Scissoring mode	Gscissor_mode_id
ViewInd	View index	INTEGER
ViewInd1	View index	INTEGER
ViewPri	View priority	Gview_pri
ViewScissor	View scissor	Gview_scissor
ViewUpVec	View up vector	Gview_up_vec
VpInPri	Viewport input priority	Gvp_in_pri
WCWin	World Coordinate Window	Gwc_win

The following list defines the derived Fortran 90 data types described above:

Gdc_vp

```
TYPE Gdc_vp
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gdc_vp
```

Gldc_win

```
TYPE Gldc_win
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gldc_win
```

Gldc_rect

```
TYPE Gldc_rect
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gldc_rect
```

Gldc_rect_set

```
TYPE Gldc_rect_set
  TYPE(Gldc_rect), POINTER, DIMENSION(:) :: rect
END TYPE Gldc_rect_set
```

Gldc_vp

```
TYPE Gldc_vp
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gldc_vp
```

Gndc_win

```
TYPE Gndc_win
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gndc_win
```

Gndc_vp

```
TYPE Gndc_vp
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gndc_vp
```

Gwc_win

```
TYPE Gwc_win
  REAL :: xmin
  REAL :: ymin
  REAL :: xmax
  REAL :: ymax
END TYPE Gwc_win
```

Gnorm_tran

```
TYPE Gnorm_tran
  TYPE(Gwc_win) :: wc_win
  TYPE(Gndc_vp) :: ndc_vp
END TYPE Gnorm_tran
```

Gview_pri

```
TYPE Gview_pri
  INTEGER :: pos(16)
END TYPE Gview_pri
```

Gview_scissor

```
TYPE Gview_scissor
  TYPE(Gclip_ind_id)   :: clip_ind
  TYPE(Gldc_rect_set)  :: clip_rect_set
  TYPE(Gshield_ind_id) :: shield_ind
  TYPE(Gldc_rect_set)  :: shield_rect_set
END TYPE Gview_scissor
```

Gview_up_vec

```
TYPE Gview_up_vec
  TYPE(Gwc_point) :: x
  TYPE(Gwc_point) :: y
END TYPE Gview_up_vec
```

Gvp_in_pri

```
TYPE Gvp_in_pri
  INTEGER :: pos(64)
END TYPE Gvp_in_pri
```

6.5 Data types for NDC picture

GKS Data Type		Fortran binding data type
Position	Position	`Gpos_id`

6.6 Data types for metafiles

GKS Data Type		Fortran binding data type and attributes
PicId	Picture Identifier	`INTEGER`
NDCMfSpec	NDC Metafile Specifier	`CHARACTER, POINTER`
ReifMfSpec	Reified Metafile Specifier	`CHARACTER, POINTER`

6.7 Data types for picture parts and archives

GKS Data Type		Fortran binding data type and attributes
PicPartName	Picture Part Name	`INTEGER`
ScissorSelect	Scissor Select	`Gscissor_sel_id`
TransMode	Transformation Mode	`Gtrans_mode_id`
ArName	Archive Name	`INTEGER`
ArSpec	Archive Specifier	`CHARACTER, POINTER`

6.8 Data types for utility functions

GKS Data Type		Fortran binding data type and attributes
CircularArcSpec	Circular arc specification	`Gcirc_arc_spec`
EllipseSpec	Ellipse specification	`Gellip_spec`
HyperbolaSpec	Hyperbola specification	`Ghyperbola_spec`
ParabolaSpec	Parabola specification	`Gparabola_spec`
NDCPoint	NDC Point	`Gndc_point`
OrdSelector	Ordinate Selector	`Gord_sel_id`
Rad	Radians	`REAL`
ScaleFacs	Scale Factors	`Gscale_facs`
Vec	Vector	`Gvec`
WCNDCPoint	WC or NDC Point	`Gwc_ndc_point`

The following list defines the derived Fortran 90 data types described above:

Gcirc_arc_spec

```
TYPE Gcirc_arc_spec
  REAL :: sx
  REAL :: sy
  REAL :: ix
  REAL :: iy
  REAL :: ex
  REAL :: ey
END TYPE Gcirc_arc_spec
```

Gellip_spec

```
TYPE Gellip_spec
  REAL :: px
  REAL :: py
  REAL :: cx
  REAL :: cy
  REAL :: qx
  REAL :: qy
END TYPE Gellip_spec
```

Ghyperbola_spec

```
TYPE Ghyperbola_spec
  REAL :: px
  REAL :: py
  REAL :: cx
  REAL :: cy
  REAL :: qx
  REAL :: qy
END TYPE Ghyperbola_spec
```

Gparabola_spec

```
TYPE Gparabola_spec
  REAL :: px
  REAL :: py
  REAL :: cx
```

```
    REAL :: cy
    REAL :: qx
    REAL :: qy
  END TYPE Gparabola_spec
```

Gndc_point

```
  TYPE Gndc_point
    REAL :: x
    REAL :: y
  END TYPE Gndc_point
```

Gscale_facs

```
  TYPE Gscale_facs
    REAL :: x
    REAL :: y
  END TYPE Gscale_facs
```

Gvec

```
  TYPE Gvec
    REAL :: x
    REAL :: y
  END TYPE Gvec
```

Gwc_ndc_point

```
  TYPE Gwc_ndc_point
    TYPE(Gwcndc_point_id)  :: point_type
    TYPE(Gndc_point)       :: ndc_point
    TYPE(Gwc_point)        :: wc_point
  END TYPE Gwc_ndc_point
```

6.9 Data types for segments

GKS Data Type		Fortran binding data type and attributes
SegAttrName	Segment attribute name	Gseg_attr_name_id
Det	Detectability	Gdet_id
Highl	Highlighting	Ghighl_id
SegPri	Priority	REAL
Vis	Visibility	Gvis_id
SegAttrVal	Segment attribute value	Gseg_attr_val

GSeg_attr_val

```
TYPE Gseg_attr_val
  TYPE(Gseg_attr_name_id) :: attr_name
  TYPE(Gvis_id)           :: vis
  TYPE(Ghighl_id)         :: highl
  TYPE(Gdet_id)           :: det
  TYPE(Gmatrix_23)        :: tran
  REAL                    :: pri
END TYPE Gseg_attr_val
```

6.10 Data Types for input

GKS Data Type		Fortran binding data type and attributes
ChoiceNum	Choice Number	`INTEGER`
ChoiceStatus	Choice Status	`Gchoice_st_id`
DevCompose	Device Composition	`Gdev_compose`
DevId	Device identifier	`Gdev_id`
DevIdValueSet	Device identifier value set	`Gdev_id_val_set`
EchoArea	Echo Area	`Gecho_area`
EchoSwitch	Echo Switch	`Gecho_switch_id`
InitInValue	Initial input value	`Ginit_in_val`
InitValue	Initial value	`Ginit_val`
InData	Input data	`Gin_data`
InStatus	Input status	`Gin_st_id`
InValue	Input value	`Gin_val`
MeasureClass	Input Measure Class	`Gmeasure_class_id`
MeasureId	Input Measure Identifier	`INTEGER`
OpMode	Input Mode	`Gop_mode_id`
PET	Prompt and Echo type	`INTEGER`
PickStatus	Pick status	`Gpick_st_id`
SeqMeasureId	Sequence of input measure identifier	`Gseq_measure_id`
Timeout	Timeout	`REAL`
TriggerId	Trigger identifier	`INTEGER`
TriggerSet	Trigger set	`Gtrigger_set`

The following auxiliary data types are defined:

Data Type		Fortran binding data type and attributes
locatorvalue	Locator value	`Gloc_val`
strokevalue	Stroke value	`Gstroke_val`
valuatorvalue	Valuator value	`REAL`
choicevalue	Choice value	`Gchoice_val`
pickvalue	Pick value	`Gpick_val`
stringvalue	String value	`CHARACTER, POINTER`
compvalue	Composite value	`Gcomp_val`

The following list defines the derived Fortran 90 data types mentioned above:

Gdev_compose

```
TYPE Gdev_comp
  INTEGER, POINTER, DIMENSION(:) :: measure_id
  INTEGER, POINTER, DIMENSION(:) :: trigger_id
END TYPE Gdev_comp
```

Gdev_id

```
TYPE Gdev_id
  INTEGER :: ws_id
  INTEGER :: dev_num
  INTEGER :: measure_class
END TYPE Gdev_id
```

Gloc_val

```
TYPE Gloc_val
  INTEGER          :: norm_tran_num
  TYPE(Gwc_point)  :: wc_point
END TYPE Gloc_val
```

Gstroke_val

```
TYPE Gstroke_val
  INTEGER                                :: norm_tran_num
  TYPE(Gwc_point), POINTER, DIMENSION(:) :: point
END TYPE Gstroke_val
```

Gchoice_val

```
TYPE Gchoice_val
  INTEGER             :: num
  TYPE(Gchoice_st_id) :: st
END TYPE Gchoice_val
```

Gpick_val

```
TYPE Gpick_val
  TYPE(Gpick_st_id) :: st
  TYPE(Gname_set)   :: name_set
  INTEGER           :: pick_id
END TYPE Gpick_val
```

Gcomp_val

```
TYPE Gcomp_val
  TYPE(Gin_val), POINTER, DIMENSION(:) :: in_val
END TYPE Gcomp_val
```

Gin_val

```
TYPE Gin_val
  TYPE(Gmeasure_class_id) :: measure_class
  TYPE(Gloc_val)          :: loc_val
  TYPE(Gstroke_val)       :: stroke_val
  REAL                    :: val_val
  TYPE(Gchoice_val)       :: choice_val
  TYPE(Gpick_val)         :: pick_val
  CHARACTER, POINTER      :: string_val(:)
  TYPE(Gcomp_val)         :: comp_val
END TYPE Gin_val
```

Gdev_id_val_set

```
TYPE Gdev_id_val_set
 TYPE(Gdev_id), POINTER, DIMENSION(:) :: dev_id
 TYPE(Gin_val), POINTER, DIMENSION(:) :: in_val
END TYPE Gdev_id_val_set
```

Gecho_area

```
TYPE Gecho_area
  REAL :: xmin
  REAL :: xmax
  REAL :: ymin
  REAL :: ymax
END TYPE Gecho_area
```

Gin_data

```
TYPE Gin_data
  CHARACTER, POINTER :: data(:)
END TYPE Gin_data
```

Ginit_in_val

```
TYPE Ginit_in_val
  TYPE(Gin_val)         :: in_value
  INTEGER               :: pet
  TYPE(Gecho_switch_id) :: echo_switch
  TYPE(Gecho_area)      :: echo_area
  TYPE(Gin_data)        :: in_data
END TYPE Ginit_in_val
```

Ginit_val

```
TYPE Ginit_val
  INTEGER            :: init_val
  TYPE(Gin_val)      :: in_val
  INTEGER            :: pet
  INTEGER            :: echo_flag
  TYPE(Gecho_area)   :: echo_area
  TYPE(Gin_data)     :: in_data
  TYPE(Ginit_in_val) :: all
END TYPE Ginit_val
```

Gseq_measure_id

```
TYPE Gseq_measure_id
  INTEGER, POINTER, DIMENSION(:) :: measure_id
END TYPE Gseq_measure_id
```

Gtrigger_set

```
TYPE Gtrigger_set
  INTEGER, POINTER, DIMENSION(:) :: trigger
END TYPE Gtrigger_set
```

6.11 Data Types for Workstation Control

GKS Data Type		Fortran binding data type and attributes
WsGenType	Workstation generic type	INTEGER
WsId	Workstation identifier	INTEGER
WsSpec	Workstation specifier	CHARACTER, POINTER
WsSpecInfo	Workstation specifier information	CHARACTER, POINTER
WsType	Workstation Type	Gws_type
ChangeFlag	Change flag	Gchange_flag_id
ColrAvail	Colour availability	Gcolr_avail_id
DCUnits	DC units	Gdc_units_id
DispKind	Display kind	Gdisp_kind_id
WsStatus	Workstation status	Gws_st_id
AreaBun	Area bundle	Garea_bun
CharBun	Character bundle	Gchar_bun
CurveBun	Curve bundle	Gcurve_bun
MarkerBun	Marker bundle	Gmarker_bun
ColrModel	Colour model	Gcolr_model_id
ColrCIELUV	CIELUV coordinates	Gcolr_cieluv
ColrHLS	HLS coordinates	Gcolr_hls
ColrHSV	HSV coordinates	Gcolr_hsv
ColrRGB	RGB coordinates	Gcolr_rgb
ColrCoords	Colour coordinates	Gcolr_coord
ColrCharacteristics	Colour Characteristics	Gcolr_char
ConversionFlag	Conversion flag	Gconv_flag_id
VisEffSt	Visual effect state	Gvis_eff_st_id
AreaBunTab	Area bundle table	TYPE(Garea_bun_elem), POINTER, DIMENSION(:)
CharBunTab	Char bundle table	TYPE(Gchar_bun_elem), POINTER, DIMENSION(:)
ColrTab	Colour table	TYPE(Gcolr_elem), POINTER, DIMENSION(:)
CurveBunTab	Curve bundle table	TYPE(Gcurve_bun_elem), POINTER, DIMENSION(:)
MarkerBunTab	Marker bundle table	TYPE(Gmarker_bun_elem), POINTER, DIMENSION(:)
PatTab	Pattern table	TYPE(Gpat_elem), POINTER, DIMENSION(:)
Rep	Representation	Grep
SelectCritType	Selection criteria type	Gsel_crit_id
SelCrit	Selection criteria	Gsel_crit
Selects	workstation selection criteria	Gsels

The following auxiliary data types are defined:

GKS Data Type		Fortran binding data type
AreaBunElem	area bundle element	Garea_bun_elem
CharBunElem	character bundle element	Gchar_bun_elem
ColrElem	colour element	Gcolr_elem
CurveBunElem	curve bundle element	Gcurve_bun_elem
MarkerBunElem	marker bundle element	Gmarker_bun_elem
PatElem	pattern element	Gpat_elem
DyadicArgs	Dyadic arguments	Gdyadic_args

The following list defines the derived Fortran 90 data types which are described above:

Gws_type

```
TYPE Gws_type
  INTEGER            :: ws_gen_type
  CHARACTER, POINTER :: ws_spec_info(:)
END TYPE Gws_type
```

Garea_bun

```
TYPE Garea_bun
  TYPE(Gint_style_id) :: int_style
  INTEGER             :: int_style_ind
  TYPE(Gcolr_spec)    :: int_colr_spec
  TYPE(Gedge_flag_id) :: edgeflag
  INTEGER             :: edgetype
  REAL                :: edgewidth
  TYPE(Gcolr_spec)    :: edge_colr_spec
END TYPE Garea_bun
```

Gchar_bun

```
TYPE Gchar_bun
  TYPE(Gfont_prec) :: font_prec
  REAL             :: expan
  REAL             :: space
  TYPE(Gcolr_spec) :: colr_spec
END TYPE Gchar_bun
```

Gcurve_bun

```
TYPE Gcurve_bun
  INTEGER          :: type
  REAL             :: width
  TYPE(Gcolr_spec) :: colr_spec
END TYPE Gcurve_bun
```

Gmarker_bun

```
TYPE Gmarker_bun
  INTEGER          :: type
  REAL             :: size
  TYPE(Gcolr_spec) :: colr_spec
END TYPE Gmarker_bun
```

Garea_bun_elem

```
TYPE Garea_bun_elem
  INTEGER          :: area_ind
  TYPE(Garea_bun)  :: area_bun
END TYPE Garea_bun_elem
```

Gchar_bun_elem

```
TYPE Gchar_bun_elem
  INTEGER          :: char_ind
  TYPE(Gchar_bun)  :: char_bun
END TYPE Gchar_bun_elem
```

Gcolr_elem

```
TYPE Gcolr_elem
  INTEGER          :: colr_ind
  TYPE(Gcolr_coord) :: colr_bun
END TYPE Gcolr_elem
```

Gcurve_bun_elem

```
TYPE Gcurve_bun_elem
  INTEGER         :: curve_ind
  TYPE(Gcurve_bun) :: curve_bun
END TYPE Gcurve_bun_elem
```

Gmarker_bun_elem

```
TYPE Gmarker_bun_elem
  INTEGER          :: marker_ind
  TYPE(Gmarker_bun) :: marker_bun
END TYPE Gmarker_bun_elem
```

Gpat_elem

```
TYPE Gpat_elem
  INTEGER                :: pat_ind
  TYPE(Gcolr_array_spec) :: pat_bun
END TYPE Gpat_elem
```

Grep

```
TYPE Grep
  TYPE(Grep_id)          :: prim_rep
  TYPE(Gcurve_bun_elem)  :: curve
  TYPE(Gmarker_bun_elem) :: marker
  TYPE(Garea_bun_elem)   :: area
  TYPE(Gpat_elem)        :: pat
  TYPE(Gchar_bun_elem)   :: char
  TYPE(Gcolr_elem)       :: colr
END TYPE Grep
```

Gsel_crit

```
TYPE Gsel_crit
  TYPE(Gsel_op_id)            :: sel_op
  TYPE(Gname_set)             :: name_set
  TYPE(Gsel_crit), POINTER    :: monadic_arg
  TYPE(Gdyadic_args), POINTER :: dyadic_args
END TYPE Gsel_crit
```

Gdyadic_args

```
TYPE Gdyadic_args
  TYPE(Gsel_crit), POINTER :: arg1
  TYPE(Gsel_crit), POINTER :: arg2
END TYPE Gdyadic_args
```

Gsels

```
TYPE Gsels
  TYPE(Gsel_crit_id) :: sel_crit_type
  TYPE(Gsel_crit)    :: disp_sel_crit
  TYPE(Gsel_crit)    :: highl_sel_crit
  TYPE(Gsel_crit)    :: det_sel_crit
END TYPE Gsels
```

6.12 Data types for inquiry

GKS Data Type		Fortran binding data type and attributes
ErrInd	Error indicator	INTEGER
SetReal	Set or realized values	Gset_realized_id

6.13 Data Types for operating state

GKS Data Type		Fortran binding data type
GKSOpSt	GKS operating state	Ggks_op_st_id
PicPartOpSt	Picture part operating state	Gpic_part_op_st_id
SegOpSt	segment operating state	Gseg_op_st_id
TilingOpSt	Tiling operating state	Gtiling_op_st_id
StencilOpSt	Stencil operating state	Gstencil_op_st_id

6.14 Data types for glyph functions

GKS Data Type		Fortran binding data type and attributes
GlyphName	Glyph name	CHARACTER, POINTER
ISO954FontName	ISO954 font name	CHARACTER, POINTER

6.15 Data types for paths, tilings and stencils

GKS Data Type		Fortran binding data type and attributes
AbutSpec	Abut specifier	Gabut_spec_id
Boundary	Boundary	Gbndry
Cap	Cap type	Gcap_id
ContourAttrName	Contour attribute name	Gcont_attr_name_id
ContourAttrValue	Contour attribute value	Gcont_attr_val
HLCPoint	Homogeneous local coordinate	Ghlc_point
InsideRule	Inside rule	Ginside_rule_id
InstanceNum	Number of instances	REAL
InstanceSeqSpec	Sequence of instance specifiers	Ginstance_seq_spec
InstanceSpec	Instance specifier	Ginstance_spec
Join	Join	Gjoin
LCControlPoints	Local coordinate control points	Glc_ctrl_point
LCConicSection	Local coordinate conic section	Glc_conic_sec
LCNURB	Local coordinate NURB	Glc_nurb
LCOrd	Local coordinate ordinate	REAL
LCPoint	local coordinate point	Glc_point
Path	Path	Gpath
PathSpec	Path specifier	Gpath_spec
RepTech	Replication technique	Grep_tech
SeqBoundary	Sequence of boundaries	Gseq_bndry
SeqPaths	Sequence of paths	Gseq_path
SeqSpec	Sequence of stencil specifications	Gseq_spec
StencilAttrName	Stencil attribute name	Gstencil_attr_name_id
StencilAttrValue	Stencil attribute value	Gstencil_attr_val
Style	Style	Gstyle_id
TilingParam	Tiling parameter	Gtiling_param

The following auxiliary data types are defined:

GKS Data Type		Fortran binding data type and attributes
Put	put instance	`Gput`
Align	align instance	`Galign`
Map	map instance	`Gmap`
seq HLCPoint	Homogeneous local coordinate control points	`Gseq_hlc_point`
seq LCPoint	Local coordinate control points	`Gseq_lc_point`
dxy	replication technique dxy	`Gdxy`
dyx	replication technique dyx	`Gdyx`
topy	topy position	`Gtopy`
capy	capy position	`Gcapy`
halfy	halfy position	`Ghalfy`
basey	basey position	`Gbasey`
bottomy	bottomy position	`Gbottomy`
centrey	centrey position	`Gcentrey`
leftx	leftx position	`Gleftx`
rightx	rightx position	`Grightx`
centrex	centrex position	`Gcentrex`
centre	centre position	`Glc_point_centre`
origin	origin position	`Glc_point_origin`
centretop	centretop position	`Glc_point_centretop`
centrebottom	centrebottom position	`Glc_point_centrebottom`
centreleft	centreleft position	`Glc_point_centreleft`
centreright	centreright position	`Glc_point_centreright`
topleft	topleft position	`Glc_point_topleft`
topright	topright position	`Glc_point_topright`
bottomleft	bottomleft position	`Glc_point_bottomleft`
bottomright	bottomright position	`Glc_point_bottomright`

Glc_point

```
TYPE Glc_point
  REAL :: x
  REAL :: y
END TYPE Glc_point
```

Gput

```
TYPE Gput
  TYPE(Glc_point) :: instance_pt
  TYPE(Glc_point) :: stencil_pt
END TYPE Gput
```

Galign

```
TYPE Galign
  TYPE(Glc_point) :: ref_instance_pt
  TYPE(Glc_point) :: dir_instance_pt
  TYPE(Glc_point) :: ref_stencil_pt
  TYPE(Glc_point) :: dir_stencil_pt
END TYPE Galign
```

Gmap

```
TYPE Gmap
```

```
    TYPE(Glc_point) :: instance_pt1
    TYPE(Glc_point) :: instance_pt2
    TYPE(Glc_point) :: instance_pt3
    TYPE(Glc_point) :: stencil_pt1
    TYPE(Glc_point) :: stencil_pt2
    TYPE(Glc_point) :: stencil_pt3
  END TYPE Gmap
```

Gseq_hlc_point

```
  TYPE Gseq_hlc_point
    TYPE(Ghlc_point), POINTER, DIMENSION(:) :: point
  END TYPE Gseq_hlc_point
```

Glccp

```
  TYPE Gseq_lc_point
    TYPE(Glc_point), POINTER, DIMENSION(:) :: point
  END TYPE Gseq_lc_point
```

Gdxy

```
  TYPE Gdxy
    TYPE(Glc_point) :: dx
    TYPE(Glc_point) :: dy
  END TYPE Gdxy
```

Gdxy

```
  TYPE Gdyx
    TYPE(Glc_point) :: dy
    TYPE(Glc_point) :: dx
  END TYPE Gdyx
```

Gseq_path

```
  TYPE Gseq_path
    TYPE(Gpath), POINTER, DIMENSION(:) :: path
  END TYPE Gseq_path
```

Gseq_spec

```
  TYPE Gseq_spec
    INTEGER           :: instance
    INTEGER           :: stencil
    TYPE(Glc_point)   :: ref_point
  END TYPE Gseq_spec
```

Gbndry

```
  TYPE Gbndry
    TYPE(Gbndry_id)    :: bndry
    TYPE(Gseq_path)    :: closed_path_seq
    TYPE(Gmatrix_33)   :: area
  END TYPE Gbndry
```

Gjoin

```
TYPE Gjoin
  TYPE(Gjoin_id) :: join
  REAL           :: mitre_point
END TYPE Gjoin
```

Gcont_attr_val

```
TYPE Gcont_attr_val
  TYPE(Gcont_attr_name_id) :: cont_attr_type
  TYPE(Gstyle_id)          :: style
  REAL                     :: width
  TYPE(Gcap_id)            :: cap
  TYPE(Gjoin)              :: join
END TYPE Gcont_attr_val
```

Ghlc_point

```
TYPE Ghlc_point
  REAL :: x
  REAL :: y
  REAL :: w
END TYPE Ghlc_point
```

Ginstance_seq_spec

```
TYPE Ginstance_seq_spec
  TYPE(Gseq_spec), POINTER, DIMENSION(:) :: seq_spec
END TYPE Ginstance_seq_spec
```

Ginstance_spec

```
TYPE Ginstance_spec
  TYPE(Ginstance_id) :: instance_type
  TYPE(Gput)         :: put
  TYPE(Galign)       :: align
  TYPE(Gmap)         :: map
END TYPE Ginstance_spec
```

Glc_ctrl_points

```
TYPE Glc_ctrl_points
  TYPE(Glc_ctrl_point_id) :: point_type
  TYPE(Gseq_hlc_point)    :: hlccp
  TYPE(Gseq_lc_point)     :: lccp
END TYPE Glc_ctrl_points
```

Glc_conic_sec

```
TYPE Glc_conic_sec
  TYPE(Gmatrix_33)     :: matrix_33
  TYPE(Glc_point)      :: start_point
  TYPE(Glc_point)      :: end_point
  TYPE(Gsense_flag_id) :: sense_flag
END TYPE Glc_conic_sec
```

Glc_nurb

```
TYPE Glc_nurb
    INTEGER              :: k
    TYPE(Glc_ctrl_points) :: cp
    TYPE(Gknots)          :: kn
    TYPE(Gparam_ran)      :: ran
END TYPE Glc_nurb
```

Gpath

```
TYPE Gpath
  TYPE(Gpath_id)      :: path_type
  TYPE(Gseq_lc_point) :: line
  TYPE(Glc_nurb)      :: nurb
  TYPE(Glc_conic_sec) :: conic_sec
END TYPE Gpath
```

Gpath_spec

```
TYPE Gpath_spec
  TYPE(Gabut_spec_id) :: abut_spec
  TYPE(Gpath)         :: path
  TYPE(Glc_point)     :: lc_point
END TYPE Gpath_spec
```

Grep_tech

```
TYPE Grep_tech
  TYPE(Greplic_tech_id) rep_type
  REAL       :: dx
  REAL       :: dy
  TYPE(Gdxy) :: dxy
  TYPE(Gdyx) :: dyx
END TYPE Grep_tech
```

Gseq_bndry

```
TYPE Gseq_bndry
  TYPE(Gbndry), POINTER, DIMENSION(:) :: bndry
END TYPE Gseq_bndry
```

Gtopy

```
TYPE Gtopy
  REAL :: p
END TYPE Gtopy
```

Gcapy

```
TYPE Gcapy
  REAL :: p
END TYPE Gcapy
```

Ghalfy

```
TYPE Ghalfy
  REAL :: p
END TYPE Ghalfy
```

Gbasey

```
TYPE Gbasey
  REAL :: p
END TYPE Gbasey
```

Gbottomy

```
TYPE Gbottomy
  REAL :: p
END TYPE Gbottomy
```

Gcentrey

```
TYPE Gcentrey
  REAL :: p
END TYPE Gcentrey
```

Gleftx

```
TYPE Gleftx
  REAL :: p
END TYPE Gleftx
```

Grightx

```
TYPE Grightx
  REAL :: p
END TYPE Grightx
```

Gcentrex

```
TYPE Gcentrex
  REAL :: p
END TYPE Gcentrex
```

Glc_point_centre

```
TYPE Glc_point_centre
  REAL :: x
  REAL :: y
END TYPE Glc_point_centre
```

Glc_point_origin

```
TYPE Glc_point_origin
  REAL :: x
  REAL :: y
END TYPE Glc_point_origin
```

Glc_point_centretop

```
TYPE Glc_point_centretop
  REAL :: x
  REAL :: y
END TYPE Glc_point_centretop
```

Glc_point_centrebottom

```
TYPE Glc_point_centrebottom
  REAL :: x
  REAL :: y
END TYPE Glc_point_centrebottom
```

Glc_point_centreleft

```
TYPE Glc_point_centreleft
  REAL :: x
  REAL :: y
END TYPE Glc_point_centreleft
```

Glc_point_centreright

```
TYPE Glc_point_centreright
  REAL :: x
  REAL :: y
END TYPE Glc_point_centreright
```

Glc_point_topleft

```
TYPE Glc_point_topleft
  REAL :: x
  REAL :: y
END TYPE Glc_point_topleft
```

Glc_point_topright

```
TYPE Glc_point_topright
  REAL :: x
  REAL :: y
END TYPE Glc_point_topright
```

Glc_point_bottomleft

```
TYPE Glc_point_bottomleft
  REAL :: x
  REAL :: y
END TYPE Glc_point_bottomleft
```

Glc_point_bottomright

```
TYPE Glc_point_bottomright
  REAL :: x
  REAL :: y
END TYPE Glc_point_bottomright
```

Gstencil_attr_val

```
TYPE Gstencil_attr_val
  TYPE(Gstencil_attr_name_id) attr_type
  TYPE(Gtopy)                 :: topy
  TYPE(Gcapy)                 :: capy
  TYPE(Ghalfy)                :: halfy
  TYPE(Gbasey)                :: basey
  TYPE(Gbottomy)              :: bottomy
  TYPE(Gcentrey)              :: centrey
  TYPE(Gleftx)                :: leftx
  TYPE(Grightx)               :: rightx
  TYPE(Gcentrex)              :: centrex
  TYPE(Glc_point_centre)      :: centre
  TYPE(Glc_point_origin)      :: origin
  TYPE(Glc_point_centretop)   :: centretop
  TYPE(Glc_point_centrebottom) :: centrebottom
  TYPE(Glc_point_centreleft)  :: centreleft
  TYPE(Glc_point_centreright) :: centreright
  TYPE(Glc_point_topleft)     :: topleft
  TYPE(Glc_point_topright)    :: topright
  TYPE(Glc_point_bottomleft)  :: bottomleft
  TYPE(Glc_point_bottomright) :: bottomright
END TYPE Gstencil_attr_val
```

Gtiling_param

```
TYPE Gtiling_param
  TYPE(Gtiling_param_id) :: tiling_prim
  TYPE(Gset_line)        :: set_line
  TYPE(Gset_nurb)        :: set_nurb
  TYPE(Gset_conic_sec)   :: set_conic_sec
  TYPE(Gset_fill)        :: set_fill
  TYPE(Gset_ellip_sec)   :: set_ellip_sec
  TYPE(Gset_ellip_seg)   :: set_ellip_seg
  TYPE(Gset_ellip_disc)  :: set_ellip_disc
  TYPE(Gset_closed_nurb) :: set_closed_nurb
  TYPE(Gdesign)          :: design
END TYPE Gtiling_param
```

6.16 Data types for state lists and description tables

GKS Data Type		Fortran binding data type
OpSt	Operating State list	`Gop_st_list`
GDT	GKS description table	`Ggdt`
GSL	GKS state list	`Ggsl`
WSL	Workstation state list	`Gwsl`
WDT	Workstation description table	`Gwdt`
STSL	Stencil state list	`Gstsl`

The following auxiliary data types are defined for the GKS Description Table:

GKS Data Type		Fortran binding data type
P WsGenTypes	set of generic workstation types	`Gset_ws_gen_type`
P ISO9541FontName	available fonts	`Gset_font_name`
FontInd → ISO9541FontName	list of font mappings	`Gfont_tab`

The following auxiliary data types are defined for the GKS State List:

GKS Data Type		Fortran binding data type
P WsId	set of workstation identifiers	`Gset_ws_id`
P (AuditId *x* RecAuditFlag)	set of audit information	`Gset_audit_info`
P AuditId	Set of audit identifiers	`Gset_audit_id`
seq (DevId *x* InValue)	Input queue information	`Gseq_in`
FontId → ISO9541FontName	Font bundle table	`TYPE(Gfont_tab),POINTER,DIMENSION(:)`
NormTranNum → NormTran	list of normalisation trans	`Gnorm_tran_list`
OpenPicPart	picture part currently open	`Gopen_pic_part_name`
P PicPartNames	set of picture part names	`Gpic_part_name_set`
OpenSegName	name of segment currently open	`Gopen_seg_name`
P SegName	set of segment names	`Gseg_name_set`
P SSL	set of segment state lists	`Gssl_set`
OpenStencilName	name of stencil currently open	`Gopen_stencil_name`
P StencilName	set of stencil names	`Gstencil_name_set`
P STSL	set of stencil state lists	`Gstsl_set`
CurContAttrs	current contour attributes	`Gcur_cont_attr`
CurPrimAttrs	current contour attributes	`Gcur_prim_attr`
OpenTilingName	name of tiling currently open	`Gopen_tiling_name`
P TilingName	set of tiling names	`Gtiling_name_set`

The following auxiliary data types are defined for Workstation State List:

GKS Data Type		Fortran binding data type
DevOpMode	Device operating modes	`Gdev_op_mode_bun`
DevOpModeElem	Device operating mode	`Gdev_op_mode_elem`
DevInVal	Device initial values	`Gdev_in_val_bun`
DevInValElem	Device initial value	`Gdev_in_val_elem`
ViewReps	View representations	`Gview_rep`

The following auxiliary data types are defined for the Workstation Description Table:

GKS Data Type		Fortran binding data type
DispSize	Display size	`Gdisp_size`
P CurveType	set of curve types	`Gcurve_type_set`
P MarkerType	set of marker types	`Gmarker_type_set`
P IntStyle	set of interior styles	`Gint_style_set`
P IntStyleInt	set of interior style indexes	`Gint_style_ind_set`
P EdgeType	set of edge types	`Gedge_type_set`
P FontPrec	set of font precision	`Gfont_prec_set`
P ColrModel	set of colour model	`Gcolr_model_set`
P GDPId	set of GDP identifiers	`Ggdp_id_set`
P DevId	set of device identifiers	`Gdev_id_set`
P TriggerId	set of trigger identifiers	`Gtrigger_id_set`
P MeasureId	set of measure identifiers	`Gmeasure_id_set`
DevIdDevCompose	device compositions	`Gdev_id_compose`

The following auxiliary data types are defined for the Error State List:

GKS Data Type		Fortran binding data type
ErrorFile	Error file	`Gerr_file`
DevIdOverflow	Device identifier causing queue overflow	`Gdev_id_overflow`

The following auxiliary data types are defined for the Segment State List:

GKS Data Type		Fortran binding data type
CurSegAttr	current segment attributes	`Gcur_seg_attr`

The following auxiliary data types are defined for the Stencil State List:

GKS Data Type		Fortran binding data type
CurStencilAttr	current stencil attributes	`Gcur_stencil_attr`

6.16.1 Operating State Lists

Gop_st_list

```
TYPE Gop_st_list
  TYPE(Gop_st_list_id)       :: op_st
  TYPE(Ggks_op_st_id)        :: gks_op_st
  TYPE(Gpic_part_op_st_id)   :: pic_part_op_st
  TYPE(Gseg_op_st_id)        :: seg_op_st
  TYPE(Gstencil_op_st_id)    :: stencil_op_st
  TYPE(Gtiling_op_st_id)     :: tiling_op_st
END TYPE Gop_st_list
```

6.16.2 GKS description table

Gws_gen_type_name

```
TYPE Gws_gen_type_name
  CHARACTER, POINTER :: type_name(:)
END TYPE Gws_gen_type_name
```

Gws_gen_type_set

```
TYPE Gws_gen_type_set
  TYPE(Gws_gen_type_name), POINTER, DIMENSION(:) :: gen_type
END TYPE Gws_gen_type_set
```

Gfont_name

```
TYPE Gfont_name
  CHARACTER, POINTER :: font_name(:)
END TYPE Gfont_name
```

Gfont_name_set

```
TYPE Gfont_name_set
  TYPE(Gfont_name), POINTER, DIMENSION(:) :: font
END TYPE Gfont_name_set
```

Gfont_tab

```
TYPE Gfont_tab
  INTEGER, POINTER, DIMENSION(:)          :: font_ind
  TYPE(Gfont_name), POINTER, DIMENSION(:) :: font
END TYPE Gfont_tab
```

Ggdt

```
TYPE Ggdt
  TYPE(Ggdt_id)          :: entry_id
  TYPE(Gws_gen_type_set) :: avail_ws_gen_types
  TYPE(Gfont_name_set)   :: avail_font
  TYPE(Gfont_tab)        :: def_font_ind_map
END TYPE Ggdt
```

6.16.3 GKS State List

Gws_id_set

```
TYPE Gws_id_set
  INTEGER, POINTER, DIMENSION(:) :: id
END TYPE Gws_id_set
```

Gaudit_info_set

```
TYPE Gaudit_info_set
  INTEGER, POINTER, DIMENSION(:) :: id
  INTEGER, POINTER, DIMENSION(:) :: flag
END TYPE Gaudit_info_set
```

Gaudit_id_set

```
TYPE Gaudit_id_set
  INTEGER, POINTER, DIMENSION(:) :: id
END TYPE Gaudit_id_set
```

Gseq_in

```
TYPE Gseq_in
  TYPE(Gdev_id), POINTER, DIMENSION(:) :: dev_id
  TYPE(Gin_val), POINTER, DIMENSION(:) :: in_val
END TYPE Gseq_in
```

Gnorm_tran_list

```
TYPE Gnorm_tran_list
  TYPE(Gnorm_tran), POINTER, DIMENSION(:) :: tran
END TYPE Gnorm_tran_list
```

Gopen_picture_part

```
TYPE Gopen_picture_part
  TYPE(Gname_id) :: avail
  INTEGER        :: name
END TYPE Gopen_picture_part
```

Gpic_part_name_set

```
TYPE Gpic_part_name_set
  INTEGER, POINTER, DIMENSION(:) :: part_name
END TYPE Gpic_part_name_set
```

Gopen_seg_name

```
TYPE Gopen_seg_name
  TYPE(Gname_id) :: avail
  INTEGER        :: name
END TYPE Gopen_seg_name
```

Gseg_name_set

```
TYPE Gseg_name_set
  INTEGER, POINTER, DIMENSION(:) :: seg_name
END TYPE Gseg_name_set
```

Gssl_set

```
TYPE Gssl_set
  TYPE(Gssl), POINTER, DIMENSION(:) :: ssl
END TYPE Gssl_set
```

Gopen_stencil_name

```
TYPE Gopen_stencil_name
  TYPE(Gname_id) :: avail
  INTEGER        :: name
END TYPE Gopen_stencil_name
```

Gstencil_name_set

```
TYPE Gstencil_name_set
  INTEGER, POINTER, DIMENSION(:) :: stencil_name
END TYPE Gstencil_name_set
```

Gstsl_set

```
TYPE Gstsl_set
  TYPE(Gstsl), POINTER, DIMENSION(:) :: stsl
END TYPE Gstsl_set
```

Gcur_cont_attr

```
TYPE Gcur_cont_attr
  TYPE(Gstyle_id) :: style
  REAL            :: width
  TYPE(Gcap_id)   :: cap
  TYPE(Gjoin)     :: join
END TYPE Gcur_cont_attr
```

Gcur_prim_attr

```
TYPE Gcur_prim_attr
  INTEGER             :: pick_id
  TYPE(Gname_set)     :: name_set
  TYPE(Gscissor_set)  :: scissor_set
  TYPE(Gmatrix_23)    :: global_tran
  TYPE(Gmatrix_23)    :: loc_tran
  TYPE(Gpat_size)     :: pat_size
  TYPE(Gwc_point)     :: pat_ref_pt
  REAL                :: text_height
  TYPE(Gtext_up_vec)  :: text_up_vec
  REAL                :: text_skew
  TYPE(Gtext_path_id) :: text_path
  TYPE(Gtext_align)   :: text_align
  INTEGER             :: curve_ind
  TYPE(Gcurve_asfs)   :: curve_asfs
  INTEGER             :: curve_type
  REAL                :: curve_width
  TYPE(Gcolr_spec)    :: curve_colr_spec
  INTEGER             :: marker_ind
  TYPE(Gmarker_asfs)  :: marker_asfs
  INTEGER             :: marker_type
  REAL                :: marker_size
  TYPE(Gcolr_spec)    :: marker_colr_spec
  INTEGER             :: area_ind
  TYPE(Garea_asfs)    :: area_asfs
  TYPE(Gint_style_id) :: int_style
  INTEGER             :: int_style_ind
```

```
    TYPE(Gcolr_spec)    :: int_colr_spec
    TYPE(Gedge_flag_id) :: edge_flag
    INTEGER             :: edge_type
    REAL                :: edge_width
    TYPE(Gcolr_spec)    :: edge_colr_spec
    INTEGER             :: char_ind
    TYPE(Gchar_asfs)    :: char_asfs
    TYPE(Gfont_prec)    :: char_font_prec
    REAL                :: char_expan
    REAL                :: char_space
    TYPE(Gcolr_spec)    :: char_colr_spec
  END TYPE Gcur_prim_attr
```

Gopen_tiling_name

```
  TYPE Gopen_tiling_name
    TYPE(Gname_id) :: avail
    INTEGER        :: name
  END TYPE Gopen_tiling_name
```

Gtiling_name_set

```
  TYPE Gtiling_name_set
    INTEGER, POINTER, DIMENSION(:) :: tiling_name
  END TYPE Gtiling_name_set
```

Ggsl

```
  TYPE Ggsl
    TYPE(Ggsl_id)                           :: entry_type
    TYPE(Groute_dir_id)                     :: route_dir
    TYPE(Gscissor_mode_id)                  :: scissor_mode
    TYPE(Gws_id_set)                        :: open_ws
    TYPE(Gws_id_set)                        :: active_ws
    TYPE(Gaudit_info_set)                   :: open_audits
    TYPE(Gaudit_id_set)                     :: open_playbacks
    TYPE(Gseq_in)                           :: in_queue
    TYPE(Gfont_tab), POINTER, DIMENSION(:)  :: font_ind_map
    TYPE(Gcur_prim_attr)                    :: cur_prim_attr
    INTEGER                                 :: cur_norm_tran
    TYPE(Gvp_in_pri)                        :: vp_in_pri
    TYPE(Gopen_picture_part)                :: open_pic_part
    TYPE(Gpic_part_name_set)                :: pic_part_names
    TYPE(Gopen_seg_name)                    :: open_seg_name
    TYPE(Gseg_name_set)                     :: seg_names
    TYPE(Gssl_set)                          :: ssl
    TYPE(Gopen_stencil_name)                :: open_stencil_name
    TYPE(Gstencil_name_set)                 :: stencil_names
    TYPE(Gstsl_set)                         :: stsl
    TYPE(Gcur_cont_attr)                    :: cur_cont_attr
    TYPE(Gopen_tiling_name)                 :: tiling_name
    TYPE(Gtiling_name_set)                  :: tiling_names
  END TYPE Ggsl
```

6.16.4 Workstation state list

Gdev_op_mode_bun

```
TYPE Gdev_op_mode_bun
  TYPE(Gdev_op_mode_elem), POINTER, DIMENSION(:) :: ind
END TYPE Gdev_op_mode_bun
```

Gdev_op_mode_elem

```
TYPE Gdev_op_mode_elem
  TYPE(Gdev_id)     :: dev_id
  TYPE(Gop_mode_id) :: op_mode
END TYPE Gdev_op_mode_elem
```

Gdev_in_val_bun

```
TYPE Gdev_in_val_bun
  TYPE(Gdev_in_val_elem), POINTER, DIMENSION(:) :: ind
END TYPE Gdev_in_val_bun
```

Gdev_in_val_elem

```
TYPE Gdev_in_val_elem
  TYPE(Gdev_id)  dev_id
  TYPE(Gin_val)  in_val
END TYPE Gdev_in_val_elem
```

Gview_rep

```
TYPE Gview_rep
  INTEGER, POINTER, DIMENSION(:)              :: ind
  TYPE(Gsel_crit), POINTER, DIMENSION(:)      :: sel_crit
  TYPE(Gmatrix_23), POINTER, DIMENSION(:)     :: ori
  TYPE(Gmatrix_23), POINTER, DIMENSION(:)     :: map
  TYPE(Gview_scissor), POINTER, DIMENSION(:)  :: view_scissor
END TYPE Gview_rep
```

Gwsl

```
TYPE Gwsl
  TYPE(Gwsl_id)                                    :: entry_type
  TYPE(Gcurve_bun_elem), POINTER, DIMENSION(:)     :: curve_bun_tab
  TYPE(Gmarker_bun_elem), POINTER, DIMENSION(:)    :: marker_bun_tab
  TYPE(Garea_bun_elem), POINTER, DIMENSION(:)      :: area_bun_tab
  TYPE(Gpat_elem), POINTER, DIMENSION(:)           :: pat_tab
  TYPE(Gchar_bun_elem), POINTER, DIMENSION(:)      :: char_bun_tab
  TYPE(Gcolr_elem), POINTER, DIMENSION(:)          :: colr_tab
  TYPE(Gldc_win)                                   :: ws_win
  TYPE(Gdc_vp)                                     :: ws_vp
  TYPE(Gvis_eff_st_id)                             :: vis_eff_st
  TYPE(Gdev_op_mode_bun)                           :: dev_op_modes
  TYPE(Gdev_in_val_bun)                            :: dev_in_vals
  TYPE(Gsels)                                      :: sel_crit
  TYPE(Gseg_name_set)                              :: stored_segs
  TYPE(Gview_rep)                                  :: view_reps
  TYPE(Gview_pri)                                  :: view_pris
END TYPE Gwsl
```

6.16.5 Workstation description table

Gdev_unit

```
TYPE Gdev_unit
  REAL :: x
  REAL :: y
END TYPE Gdev_unit
```

Graster_unit

```
TYPE Graster_unit
  INTEGER :: x
  INTEGER :: y
END TYPE Graster_unit
```

Gdisp_size

```
TYPE Gdisp_size
  TYPE(Gdev_unit)    :: dev_unit
  TYPE(Graster_unit) :: raster_unit
END TYPE Gdisp_size
```

Gdev_id_compose

```
TYPE Gdev_id_compose
  TYPE(Gdev_id), POINTER, DIMENSION(:)   :: dev_id
  TYPE(Gdev_comp), POINTER, DIMENSION(:) :: dev_comp
END TYPE Gdev_id_compose
```

Gwdt

```
TYPE Gwdt
  TYPE(Gwdt_id)                                     :: entry_type
  TYPE(Gws_type)                                    :: wstype
  TYPE(Gws_st_id)                                   :: ws_st
  TYPE(Gdc_units_id)                                :: dc_units
  TYPE(Gdisp_size)                                  :: disp_size
  TYPE(Gdisp_kind_id)                               :: disp_kind
  INTEGER, POINTER, DIMENSION(:)                    :: curve_types
  INTEGER                                           :: num_curve_widths
  REAL                                              :: nom_curve_width
  REAL                                              :: min_curve_width
  REAL                                              :: max_curve_width
  TYPE(Gcurve_bun_elem), POINTER, DIMENSION(:)      :: curve_bun_tab
  INTEGER, POINTER, DIMENSION(:)                    :: marker_types
  INTEGER                                           :: num_marker_widths
  REAL                                              :: nom_marker_width
  REAL                                              :: min_marker_width
  REAL                                              :: max_marker_width
  TYPE(Gmarker_bun_elem), POINTER, DIMENSION(:)     :: marker_bun_tab
  TYPE(Gint_style_id), POINTER, DIMENSION(:)        :: int_styles
  INTEGER, POINTER, DIMENSION(:)                    :: int_style_inds
  TYPE(Garea_bun_elem), POINTER, DIMENSION(:)       :: area_bun_tab
```

```
    INTEGER, POINTER, DIMENSION(:)                      :: edge_types
    INTEGER                                             :: num_edge_widths
    REAL                                                :: nom_edge_width
    REAL                                                :: min_edge_width
    REAL                                                :: max_edge_width
    TYPE(Gpat_elem), POINTER, DIMENSION(:)              :: pat_tab
    TYPE(Gfont_prec), POINTER, DIMENSION(:)             :: font_precs
    INTEGER                                             :: num_char_expans
    REAL                                                :: min_char_expan
    REAL                                                :: max_char_expan
    INTEGER                                             :: num_text_heights
    REAL                                                :: min_text_height
    REAL                                                :: max_text_height
    TYPE(Gchar_bun_elem), POINTER, DIMENSION(:)         :: char_bun_tab
    INTEGER, POINTER, DIMENSION(:)                      :: colr_models
    CHARACTER, POINTER                                  :: colr_chars(:)
    INTEGER                                             :: num_colrs
    CHARACTER, POINTER                                  :: colr_avail(:)
    TYPE(Gcolr_elem), POINTER, DIMENSION(:)             :: colr_tab
    INTEGER, POINTER, DIMENSION(:)                      :: gdp_ids
    INTEGER, POINTER, DIMENSION(:)                      :: dev_ids
    TYPE(Gdev_in_val_bun)                               :: dev_in_val
    INTEGER, POINTER, DIMENSION(:)                      :: measure_ids
    INTEGER, POINTER, DIMENSION(:)                      :: trigger_ids
    TYPE(Gdev_id_compose)                               :: dev_comp
  END TYPE Gwdt
```

6.16.6 Error state list

Gerr_file

```
  TYPE Gerr_file
    TYPE(Gname_id)     :: st
    CHARACTER, POINTER :: file_name(:)
  END TYPE Gerr_file
```

Gdev_id_overflow

```
  TYPE Gdev_id_overflow
    TYPE(Gname_id) :: st
    INTEGER        :: id
  END TYPE Gdev_id_overflow
```

6.16.7 Segment State List

Gcur_seg_attr

```
  TYPE Gcur_seg_attr
    TYPE(Gvis_id)          :: vis
    TYPE(Ghighl_id)        :: highl
    TYPE(Gdet_id)          :: det
    TYPE(Gmatrix_23)       :: tran
    REAL                   :: pri
  END TYPE Gcur_seg_attr
```

Gssl

```
TYPE Gssl
  TYPE(Gssl_id)       :: entry_val
  INTEGER             :: seg_name
  TYPE(Gws_id_set)    :: ws_ids
  TYPE(Gcur_seg_attr) :: seg_attrs
END TYPE Gssl
```

6.16.8 Stencil State List

Gcur_stencil_attr

```
TYPE Gcur_stencil_attr
  TYPE(Gstencil_attr_val), POINTER, DIMENSION(:) :: stencil_attr
END TYPE Gcur_stencil_attr
```

Gstsl

```
TYPE Gstsl
  TYPE(Gstsl_id)          :: entry_type
  INTEGER                 :: stencil_name
  TYPE(Gcur_stencil_attr) :: stencil_attrs
END TYPE Gstsl
```

6.17 Data Types for Audit Trails

GKS Data Type		Fortran binding data type and attributes
AuditId	Audit Identifier	INTEGER
AuditOps	Audit Operations	Gaudit_ops
AuditSpec	Audit Specifier	CHARACTER, POINTER
AuditUserData	Audit user data	Gaudit_user_data
ProcessOps	Process Operations	Gprocess_op_id
PlaybackOps	Playback Operations	Gplayback_ops
RecAuditFlag	Record audit flag	Grec_audit_flag_id
FuncParams	Function Parameters	Gfunc_params

The following list defines the derived Fortran 90 data types which are mentioned above:

Gaudit_ops

```
TYPE Gaudit_ops
  TYPE(Gaudit_op_id) :: audit_op
  CHARACTER, POINTER :: audit_spec(:)
END TYPE Gaudit_ops
```

Gaudit_user_data

```
TYPE Gaudit_user_data
  CHARACTER, POINTER :: data(:)
END TYPE Gaudit_user_data
```

Gplayback_ops

```
TYPE Gplayback_ops
  TYPE(Gplayback_op_id) :: audit_op
  CHARACTER, POINTER    :: audit_spec(:)
END TYPE Gplayback_ops
```

Gfa_open_gks

```
TYPE Gfa_open_gks
  CHARACTER, POINTER ::  err_file(:)
END TYPE Gfa_open_gks
```

Gfa_close_gks

```
TYPE Gfa_close_gks
  INTEGER :: NULL
END TYPE Gfa_close_gks
```

Gfa_save_gsl

```
TYPE Gfa_save_gsl
  INTEGER                                     :: wsid
  TYPE(Ggsl_id), POINTER, DIMENSION (:)       :: ent_names
  TYPE(Ggsl), POINTER, DIMENSION (:)          :: ent_vals
END TYPE Gfa_save_gsl
```

Gfa_restore_gsl

```
TYPE Gfa_restore_gsl
  INTEGER                            :: wsid
  TYPE(Ggsl), POINTER, DIMENSION (:) :: ent_vals
END TYPE Gfa_restore_gsl
```

Gfa_escape

```
TYPE Gfa_escape
  INTEGER                               :: escape_id
  CHARACTER, POINTER                    :: in_data
  CHARACTER, POINTER                    :: out_data
END TYPE Gfa_escape
```

Gfa_emergency_close_gks

```
TYPE Gfa_emergency_close_gks
  INTEGER :: NULL
END TYPE Gfa_emergency_close_gks
```

Gfa_route

```
TYPE Gfa_route
  TYPE(Groute_dir_id) :: dir
END TYPE Gfa_route
```

Gfa_create_out_prim

```
TYPE Gfa_create_out_prim
  TYPE(Gprim_param_id)         :: prim
  TYPE(Gset_line)              :: line
  TYPE(Gset_nurb)              :: nurb
  TYPE(Gset_conic_sec)         :: conic_sec
  TYPE(Gmarker)                :: marker
  TYPE(Gset_fill)              :: fill
  TYPE(Gset_ellip_sec)         :: ellip_sec
  TYPE(Gset_ellip_seg)         :: ellip_seg
  TYPE(Gset_ellip_disc)        :: ellip_disc
  TYPE(Gset_closed_nurb)       :: closed_nurb
  TYPE(Gtext)                  :: text
  TYPE(Gcell_array)            :: cell_array
  TYPE(Gdesign)                :: design
  TYPE(Ggdp)                   :: gdp
END TYPE Gfa_create_out_prim
```

Gfa_open_stencil

```
TYPE Gfa_open_stencil
  INTEGER              :: stencil_name
  TYPE(Ginside_rule_id) :: inside_rule
END TYPE Gfa_open_stencil
```

Gfa_close_stencil

```
TYPE Gfa_close_stencil
  INTEGER :: NULL
END TYPE Gfa_close_stencil
```

Gfa_rename_stencil

```
TYPE Gfa_rename_stencil
  INTEGER          :: old_stencil_name
  INTEGER          :: new_stencil_name
END TYPE Gfa_rename_stencil
```

Gfa_del_stencil

```
TYPE Gfa_del_stencil
   INTEGER          :: stencil_name
END TYPE Gfa_del_stencil
```

Gfa_create_stencil_bndry

```
TYPE Gfa_create_stencil_bndry
  INTEGER              :: stencil_name
  TYPE(Ginside_rule_id) :: inside_rule
  TYPE(Gseq_bndry)      :: bndry
END TYPE Gfa_create_stencil_bndry
```

Gfa_create_stencil_conts

```
TYPE Gfa_create_stencil_conts
  INTEGER               :: stencil_name
  TYPE(Ginside_rule_id) :: inside_rule
  TYPE(Gseq_path)       :: path
END TYPE Gfa_create_stencil_conts
```

Gfa_set_cont_attr

```
TYPE Gfa_set_cont_attr
  TYPE(Gcont_attr_name_id) :: attr
  TYPE(Gstyle_id)          :: style
  REAL                     :: width
  TYPE(Gcap_id)            :: cap
  TYPE(Gjoin)              :: join
END TYPE Gfa_set_cont_attr
```

Gfa_set_stencil_attr

```
TYPE Gfa_set_stencil_attr
    TYPE(Gstencil_attr_name_id) :: attr
    REAL                        :: topy
    REAL                        :: capy
    REAL                        :: halfy
    REAL                        :: basey
    REAL                        :: bottomy
    REAL                        :: centrey
    REAL                        :: leftx
    REAL                        :: rightx
    REAL                        :: centrex
    TYPE(Glc_point)             :: centre
    TYPE(Glc_point)             :: origin
    TYPE(Glc_point)             :: centretop
    TYPE(Glc_point)             :: centrebottom
    TYPE(Glc_point)             :: centreleft
    TYPE(Glc_point)             :: centreright
    TYPE(Glc_point)             :: topleft
    TYPE(Glc_point)             :: topright
    TYPE(Glc_point)             :: bottomleft
    TYPE(Glc_point)             :: bottomright
END TYPE Gfa_set_stencil_attr
```

Gfa_get_stencil_attr

```
TYPE Gfa_get_stencil_attr
  INTEGER                          :: stencil_name
  TYPE(Gstencil_attr_name_id)      :: attr_name
  INTEGER                          :: error_ind
  TYPE(Gstencil_attr_val)          :: attr_val
END TYPE Gfa_get_stencil_attr
```

Gfa_inst_stencil

```
TYPE Gfa_inst_stencil
  INTEGER                :: stencil_name
  INTEGER                :: instance_name
```

```
    TYPE(Gmatrix_23)       :: stencil_tran
    TYPE(Ginstance_spec)   :: instance_spec
  END TYPE Gfa_inst_stencil
```

Gfa_inst_stencil_path

```
  TYPE Gfa_inst_stencil_path
    INTEGER          :: stencil_name
    INTEGER          :: instance_name
    TYPE(Gwc_point)  :: start_map
    TYPE(Gwc_point)  :: end_map
    REAL             :: instance_num
    TYPE(Gpath)      :: path
  END TYPE Gfa_inst_stencil_path
```

Gfa_inst_stencil_seq_path

```
  TYPE Gfa_inst_stencil_seq_path
    TYPE(Ginstance_seq_spec)  :: instance_seq_spec
    TYPE(Gpath_spec)          :: path
    TYPE(Gmatrix_23)          :: stencil_tran
  END TYPE Gfa_inst_stencil_seq_path
```

Gfa_open_tiling

```
  TYPE Gfa_open_tiling
    INTEGER   :: tiling_name
  END TYPE Gfa_open_tiling
```

Gfa_close_tiling

```
  TYPE Gfa_close_tiling
    INTEGER :: NULL
  END TYPE Gfa_close_tiling
```

Gfa_rename_tiling

```
  TYPE Gfa_rename_tiling
    INTEGER :: old_tiling_name
    INTEGER :: new_tiling_name
  END TYPE Gfa_rename_tiling
```

Gfa_del_tiling

```
  TYPE Gfa_del_tiling
    INTEGER :: tiling_name
  END TYPE Gfa_del_tiling
```

Gfa_create_tiling_comp

```
TYPE Gfa_create_tiling_comp
  TYPE(Gtiling_param_id)        :: comp_name
  TYPE(Gset_line)               :: set_line
  TYPE(Gset_nurb)               :: set_nurb
  TYPE(Gset_conic_sec)          :: set_conic_sec
  TYPE(Gset_fill)               :: set_fill
  TYPE(Gset_ellip_sec)          :: set_ellip_sec
  TYPE(Gset_ellip_seg)          :: set_ellip_seg
  TYPE(Gset_ellip_disc)         :: set_ellip_disc
  TYPE(Gset_closed_nurb)        :: set_closed_nurb
  TYPE(Gdesign)                 :: design
  TYPE(Gwc_point)               :: origin
  TYPE(Greplic_tech_id)         :: rep_tech
END TYPE Gfa_create_tiling_comp
```

Gfa_set_prim_attr

```
TYPE Gfa_set_prim_attr
  TYPE(Gprim_attr_id)        :: name
  TYPE(Gpick_id)             :: pick_id
  TYPE(Gname_set)            :: name_set
  TYPE(Gscissor_set)         :: scissor_set
  TYPE(Gmatrix_23)           :: global_tran
  TYPE(Gmatrix_23)           :: loc_tran
  TYPE(Gpat_size)            :: pat_size
  TYPE(Gwc_point)            :: pat_ref_pt
  TYPE(Gtext_height)         :: text_height
  TYPE(Gtext_up_vec)         :: text_up_vec
  TYPE(Gtext_skew)           :: text_skew
  TYPE(Gtext_path_id)        :: text_path
  TYPE(Gtext_align)          :: text_align
  TYPE(Gcurve_ind)           :: curve_ind
  TYPE(Gcurve_asfs)          :: curve_asfs
  TYPE(Gcurve_type)          :: curve_type
  TYPE(Gcurve_width)         :: curve_width
  TYPE(Gcolr_spec)           :: curve_colr_spec
  TYPE(Gmarker_ind)          :: marker_ind
  TYPE(Gmarker_asfs)         :: marker_asfs
  TYPE(Gmarker_type)         :: marker_type
  TYPE(Gmarker_size)         :: marker_size
  TYPE(Gcolr_spec)           :: marker_colr_spec
  TYPE(Garea_ind)            :: area_ind
  TYPE(Garea_asfs)           :: area_asfs
  TYPE(Gint_style_id)        :: int_style
  TYPE(Gint_style_ind)       :: int_style_ind
  TYPE(Gcolr_spec)           :: int_colr_spec
  TYPE(Gedge_flag_id)        :: edge_flag
  TYPE(Gedge_type)           :: edge_type
  TYPE(Gedge_width)          :: edge_width
  TYPE(Gcolr_spec)           :: edge_colr_spec
  TYPE(Gchar_ind)            :: char_ind
  TYPE(Gchar_asfs)           :: char_asfs
  TYPE(Gfont_prec)           :: char_font_prec
  TYPE(Gchar_expan)          :: char_expan
  TYPE(Gchar_space)          :: char_space
  TYPE(Gchar_colr_spec)      :: char_colr_spec
END TYPE Gfa_set_prim_attr
```

Gfa_add_set_names_nameset

```
TYPE Gfa_add_set_names_nameset
  TYPE (Gname_set) :: nameset
END TYPE Gfa_add_set_names_nameset
```

Gfa_remove_set_names_nameset

```
TYPE Gfa_remove_set_names_nameset
  TYPE (Gname_set) :: nameset
END TYPE Gfa_remove_set_names_nameset
```

Gfa_add_set_scissor_set

```
TYPE Gfa_add_set_scissor_set
  TYPE (Gscissor_set) :: scissor_set
END TYPE Gfa_add_set_scissor_set
```

Gfa_remove_set_scissor_set

```
TYPE Gfa_remove_set_scissor_set
  TYPE (Gscissor_id_set) :: scissor_id_set
END TYPE Gfa_remove_set_scissor_set
```

Gfa_set_win_vp

```
TYPE Gfa_set_win_vp
  INTEGER        :: normtran
  TYPE (Gwc_win) :: win_limits
  TYPE (Gndc_vp) :: vp_limits
END TYPE Gfa_set_win_vp
```

Gfa_set_vp_in_pri

```
TYPE Gfa_set_vp_in_pri
  INTEGER       :: normtran
  INTEGER       :: ref_tran_num
  TYPE(Gpri_id) :: rel_pri
END TYPE Gfa_set_vp_in_pri
```

Gfa_set_scissor_mode

```
TYPE Gfa_set_scissor_mode
  TYPE(Gscissor_mode_id) :: mode
END TYPE Gfa_set_scissor_mode
```

Gfa_set_norm_tran_num

```
TYPE Gfa_set_norm_tran_num
  INTEGER :: normtran
END TYPE Gfa_set_norm_tran_num
```

Gfa_del_prims_ndc_pic

```
TYPE Gfa_del_prims_ndc_pic
  TYPE(Gsel_crit) :: selcrit
END TYPE Gfa_del_prims_ndc_pic
```

Gfa_add_set_names_ndc_pic

```
TYPE Gfa_add_set_names_ndc_pic
  TYPE(Gname_set)   :: name_set
  TYPE (Gsel_crit)  :: selcrit
END TYPE Gfa_add_set_names_ndc_pic
```

Gfa_remove_set_names_ndc_pic

```
TYPE Gfa_remove_set_names_ndc_pic
  TYPE(Gname_set)   name_set
  TYPE(Gsel_crit)   selcrit
END TYPE Gfa_remove_set_names_ndc_pic
```

Gfa_set_ndc_pic_prim_attr

```
TYPE Gfa_set_ndc_pic_prim_attr
  TYPE(Gsel_crit)              :: selcrit
  TYPE(Gprim_attr_id)          :: name
  INTEGER                      :: pick_id
  TYPE(Gname_set)              :: name_set
  TYPE(Gscissor_set)           :: scissor_set
  TYPE(Gmatrix_23)             :: global_tran
  TYPE(Gmatrix_23)             :: loc_tran
  TYPE(Gpat_size)              :: pat_size
  TYPE(Gwc_point)              :: pat_ref_pt
  REAL                         :: text_height
  TYPE(Gtext_up_vec)           :: text_up_vec
  REAL                         :: text_skew
  TYPE(Gtext_path_id)          :: text_path
  TYPE(Gtext_align)            :: text_align
  INTEGER                      :: curve_ind
  TYPE(Gcurve_asfs)            :: curve_asfs
  INTEGER                      :: curve_type
  REAL                         :: curve_width
  TYPE(Gcolr_spec)             :: curve_colr_spec
  INTEGER                      :: marker_ind
  TYPE(Gmarker_asfs)           :: marker_asfs
  INTEGER                      :: marker_type
  REAL                         :: marker_size
  TYPE(Gcolr_spec)             :: marker_colr_spec
  INTEGER                      :: area_ind
  TYPE(Garea_asfs)             :: area_asfs
  TYPE(Gint_style_id)          :: int_style
  INTEGER                      :: int_style_ind
  TYPE(Gcolr_spec)             :: int_colr_spec
  TYPE(Gedge_flag_id)          :: edge_flag
  INTEGER                      :: edge_type
  REAL                         :: edge_width
```

```
    TYPE(Gcolr_spec)            :: edge_colr_spec
    INTEGER                     :: char_ind
    TYPE(Gchar_asfs)            :: char_asfs
    TYPE(Gfont_prec)            :: char_font_prec
    REAL                        :: char_expan
    REAL                        :: char_space
    TYPE(Gcolr_spec)            :: char_colr_spec
  END TYPE Gfa_set_ndc_pic_prim_attr
```

Gfa_remove_set_scissor_ndc_pic

```
  TYPE Gfa_remove_set_scissor_ndc_pic
    TYPE(Gscissor_id_set) :: scissor_id_set
    TYPE(Gsel_crit)       :: selcrit
  END TYPE Gfa_remove_set_scissor_ndc_pic
```

Gfa_add_set_scissor_ndc_pic

```
  TYPE Gfa_add_set_scissor_ndc_pic
    TYPE(Gscissor_set) :: scissor_set
    TYPE(Gsel_crit)    :: selcrit
  END TYPE Gfa_add_set_scissor_ndc_pic
```

Gfa_reorder_ndc_pic

```
  TYPE Gfa_reorder_ndc_pic
    TYPE(Gsel_crit)  :: src_sel
    TYPE(Gsel_crit)  :: ref_sel
    TYPE(Gpos_id)    :: rel_pos
  END TYPE Gfa_reorder_ndc_pic
```

Gfa_copy_ndc_pic_ndc_mf

```
  TYPE Gfa_copy_ndc_pic_ndc_mf
    CHARACTER, POINTER :: ndc_mf_spec(:)
    INTEGER            :: picid
    TYPE(Gsel_crit)    :: selcrit
    TYPE(Gname_set)    :: nameset
  END TYPE Gfa_copy_ndc_pic_ndc_mf
```

Gfa_copy_ndc_mf_pic_ndc_pic

```
  TYPE Gfa_copy_ndc_mf_pic_ndc_pic
    CHARACTER, POINTER :: ndc_mf_spec(:)
    INTEGER            :: picid
    TYPE(Gname_set)    :: nameset
  END TYPE Gfa_copy_ndc_mf_pic_ndc_pic
```

Gfa_open_pic_part

```
  TYPE Gfa_open_pic_part
    INTEGER :: partname
  END TYPE Gfa_open_pic_part
```

Gfa_close_pic_part

```
TYPE Gfa_close_pic_part
 INTEGER :: NULL
END TYPE Gfa_close_pic_part
```

Gfa_ar_pic_part

```
TYPE Gfa_ar_pic_part
  INTEGER            :: part_name
  CHARACTER, POINTER :: ar_spec(:)
  INTEGER            :: ar_name
END TYPE Gfa_ar_pic_part
```

Gfa_ret_pic_part_ar

```
TYPE Gfa_ret_pic_part_ar
  CHARACTER, POINTER :: ar_spec(:)
  INTEGER            :: ar_name
  INTEGER            :: part_name
END TYPE Gfa_ret_pic_part_ar
```

Gfa_reopen_pic_part

```
TYPE Gfa_reopen_pic_part
  INTEGER :: part_name
END TYPE Gfa_reopen_pic_part
```

Gfa_append_pic_part

```
TYPE Gfa_append_pic_part
  INTEGER             :: src_partname
  INTEGER             :: sink_partname
  TYPE(Gmatrix_23)    :: globaltran
  TYPE(Gtran_mode_id) :: globalmode
  TYPE(Gmatrix_23)    :: localtran
  TYPE(Gtran_mode_id) :: localmode
  TYPE(Gname_set)     :: nameset
END TYPE Gfa_append_pic_part
```

Gfa_rename_pic_part

```
TYPE Gfa_rename_pic_part
  INTEGER :: old_name
  INTEGER :: new_name
END TYPE Gfa_rename_pic_part
```

Gfa_del_pic_part

```
TYPE Gfa_del_pic_part
  INTEGER :: part_name
END TYPE Gfa_del_pic_part
```

Gfa_copy_pic_part_pps

```
TYPE Gfa_copy_pic_part_pps
  INTEGER               :: part_name
  TYPE(Gsel_crit)       :: sel_crit
  TYPE(Gmatrix_23)      :: global_tran
  TYPE(Gtran_mode_id)   :: global_mode
  TYPE(Gmatrix_23)      :: local_tran
  TYPE(Gtran_mode_id)   :: local_mode
  TYPE(Gname_set)       :: name_set
  TYPE(Gscissor_sel_id) ::  scissor_sel
END TYPE Gfa_copy_pic_part_pps
```

Gfa_copy_ndc_pic_pps

```
TYPE Gfa_copy_ndc_pic_pps
  TYPE (Gsel_crit)  :: sel_crit
  INTEGER           :: part_name
  TYPE (Gname_set)  :: name_set
END TYPE Gfa_copy_ndc_pic_pps
```

Gfa_set_in_dev_mode

```
TYPE Gfa_set_in_dev_mode
  TYPE(Gdev_id)     :: dev_id
  TYPE(Gop_mode_id) :: mode
END TYPE Gfa_set_in_dev_mode
```

Gfa_req_in

```
TYPE Gfa_req_in
  TYPE(Gdev_id)   :: dev_id
  TYPE(Gin_st_id) :: in_st
  TYPE(Gin_val)   :: in_val
END TYPE Gfa_req_in
```

Gfa_sample_in

```
TYPE Gfa_sample_in
  TYPE(Gdev_id)             :: dev_id
  TYPE(Gin_val)             :: in_val
END TYPE Gfa_sample_in
```

Gfa_await_in

```
TYPE Gfa_await_in
  REAL                              :: timeout
  TYPE(Gdev_id_val_set)             :: events
END TYPE Gfa_await_in
```

Gfa_flush_dev_events

```
  TYPE Gfa_flush_dev_events
    TYPE(Gdev_id) :: dev_id
  END TYPE Gfa_flush_dev_events
```

Gfa_set_font_ind_map

```
  TYPE Gfa_set_font_ind_map
    INTEGER            :: font_ind
    CHARACTER, POINTER :: font_name (:)
  END TYPE Gfa_set_font_ind_map
```

Gfa_get_glyph_name

```
  TYPE Gfa_get_glyph_name
    INTEGER            :: font_index
    CHARACTER(LEN=1)   :: spec_char
    CHARACTER, POINTER :: spec_glyph(:)
  END TYPE Gfa_get_glyph_name
```

Gfa_set_char_code

```
  TYPE Gfa_set_char_code
    INTEGER            :: font_index
    CHARACTER(LEN=1)   :: spec_char
    CHARACTER, POINTER :: spec_glyph(:)
  END TYPE Gfa_set_char_code
```

Gfa_audit

```
  TYPE Gfa_audit
    INTEGER            :: audit_id
    TYPE(Gaudit_op_id) :: audit_op
  END TYPE Gfa_audit
```

Gfa_write_user_rec_audit

```
  TYPE Gfa_write_user_rec_audit
    INTEGER                :: audit_id
    TYPE(Gaudit_user_data) :: rec
  END TYPE Gfa_write_user_rec_audit
```

Gfa_playback

```
  TYPE Gfa_playback
    INTEGER             :: audit_id
    TYPE(Gplayback_ops) :: playback_op
  END TYPE Gfa_playback
```

Gfa_read_item_func_name_audit

```
  TYPE Gfa_read_item_func_name_audit
    INTEGER         :: audit_id
    TYPE(Gfn_id)    :: func_name
  END TYPE Gfa_read_item_func_name_audit
```

Gfa_proc_audit_item

```
TYPE Gfa_proc_audit_item
  INTEGER              :: audit_id
  TYPE(Gprocess_op_id) :: intr_op
END TYPE Gfa_proc_audit_item
```

Gfa_inq_op_st_entry

```
TYPE Gfa_inq_op_st_entry
  TYPE(Gop_st_list_id)         :: entry_type
  INTEGER                      :: error_ind
  TYPE(Gop_st_list)            :: entry_value
END TYPE Gfa_inq_op_st_entry
```

Gfa_inq_gdt_entry

```
TYPE Gfa_inq_gdt_entry
  TYPE(Ggdt_id)          :: ent_name
  INTEGER                :: err_ind
  TYPE(Ggdt)             :: ent_val
END TYPE Gfa_inq_gdt_entry
```

Gfa_inq_gsl_entry

```
TYPE Gfa_inq_gsl_entry
  TYPE(Ggsl_id)          :: ent_name
  INTEGER                :: err_ind
  TYPE(Ggsl)             :: ent_val
END TYPE Gfa_inq_gsl_entry
```

Gfa_inq_in_overf

```
TYPE Gfa_inq_in_overf
  INTEGER                  :: err_ind
  TYPE(Gdev_id)            :: dev_id
END TYPE Gfa_inq_in_overf
```

Gfa_eval_tran_matrix

```
TYPE Gfa_eval_tran_matrix
  TYPE(Gwc_ndc_point)        :: fixed_pt
  TYPE(gwc_ndc_point)        :: shift_vec
  REAL                       :: rot_angle
  TYPE(Gscale_facs)          :: scale_facs
  TYPE(Gmatrix_23)           :: tran_mat
END TYPE Gfa_eval_tran_matrix
```

Gfa_accum_tran_matrix

```
TYPE Gfa_accum_tran_matrix
  TYPE(Gmatrix_23)              :: in_mat
  TYPE(Gwc_ndc_point)           :: fixed_pt
  TYPE(Gwc_ndc_point)           :: shift_vec
  REAL                          :: rot_angle
  TYPE(Gscale_facs)             :: scale_facs
  TYPE(Gmatrix_23)              :: tran_mat
END TYPE Gfa_accum_tran_matrix
```

Gfa_eval_circle

```
TYPE Gfa_eval_circle
  TYPE(Gwc_point)            :: centre
  REAL                       :: radius
  TYPE(Gmatrix_33)           :: circle
END TYPE Gfa_eval_circle
```

Gfa_eval_ell

```
TYPE Gfa_eval_ell
  TYPE(Gellip_spec)          :: ellip_spec
  TYPE(Gmatrix_33)           :: ellip
END TYPE Gfa_eval_ell
```

Gfa_eval_circ_arc_3_point

```
TYPE Gfa_eval_circ_arc_3_point
  TYPE(Gwc_point)            :: start_pt
  TYPE(Gwc_point)            :: imm_pt
  TYPE(Gwc_point)            :: end_pt
  TYPE(Gconic_sec)           :: circular_arc
END TYPE Gfa_eval_circ_arc_3_point
```

Gfa_eval_circ_arc_ctr

```
TYPE Gfa_eval_circ_arc_ctr
  TYPE(Gwc_point)               :: centre
  TYPE(Gvec)                    :: start_vec
  TYPE(Gvec)                    :: end_vec
  TYPE(Gsense_flag_id)          :: sense_flag
  REAL                          :: radius
  TYPE(Gconic_sec)              :: circular_arc
END TYPE Gfa_eval_circ_arc_ctr
```

Gfa_eval_ell_arc

```
TYPE Gfa_eval_ell_arc
  TYPE(Gwc_point)               :: centre
  TYPE(Gellip_spec)             :: ellip_spec
  TYPE(Gvec)                    :: start_vec
  TYPE(Gvec)                    :: end_vec
  TYPE(Gconic_sec)              :: conic_arc
END TYPE Gfa_eval_ell_arc
```

Gfa_eval_hyp_arc

```
TYPE Gfa_eval_hyp_arc
  TYPE(Ghyperbola_spec)          :: hyperbola_spec
  TYPE(Gvec)                     :: start_vec
  TYPE(Gvec)                     :: end_vec
  TYPE(Gconic_sec)               :: conic_arc
END TYPE Gfa_eval_hyp_arc
```

Gfa_eval_par_arc

```
TYPE Gfa_eval_par_arc
  TYPE(Gparabola_spec)        :: parabola_spec
  TYPE(Gconic_sec)            :: conic_arc
END TYPE Gfa_eval_par_arc
```

Gfa_eval_wc_ord

```
TYPE Gfa_eval_wc_ord
  REAL              :: wc_ordinate
  TYPE(Gord_sel_id) :: ord_sel
  REAL              :: ndc_val
END TYPE Gfa_eval_wc_ord
```

Gfa_open_ws

```
TYPE Gfa_open_ws
  INTEGER            :: wsid
  CHARACTER, POINTER :: wsspec (:)
  TYPE(Gws_type)     :: wstype
END TYPE Gfa_open_ws
```

Gfa_close_ws

```
TYPE Gfa_close_ws
  INTEGER :: wsid
END TYPE Gfa_close_ws
```

Gfa_save_wsl

```
TYPE Gfa_save_wsl
  INTEGER                               :: wsid
  TYPE(Gwsl_id), POINTER, DIMENSION (:) :: ent_names
  TYPE(Gwsl), POINTER, DIMENSION (:)    :: ent_vals
END TYPE Gfa_save_wsl
```

Gfa_restore_wsl

```
TYPE Gfa_restore_wsl
  INTEGER                            :: wsid
  TYPE(Gwsl), POINTER, DIMENSION (:) :: ent_vals
END TYPE Gfa_restore_wsl
```

Gfa_get_ws_status

```
TYPE Gfa_get_ws_status
 INTEGER         :: wsid
 TYPE(Gws_st_id) :: wsst
END TYPE Gfa_get_ws_status
```

Gfa_remove_backdrop

```
TYPE Gfa_remove_backdrop
  INTEGER :: wsid
END TYPE Gfa_remove_backdrop
```

Gfa_set_rep

```
TYPE Gfa_set_rep
  INTEGER                         :: wsid
  TYPE(Grep_id)                   :: rep_type
  INTEGER                         :: rep_ind
  TYPE(Gcurve_bun)                :: curv_bun
  TYPE(Gmarker_bun)               :: mark_bun
  TYPE(Garea_bun)                 :: area_bun
  TYPE(Gchar_bun)                 :: char_bun
  TYPE(Gcolr_array_spec)          :: patt_array
  TYPE(Gcolr_coord)               :: colr
END TYPE Gfa_set_rep
```

Gfa_set_view_pri

```
TYPE Gfa_set_view_pri
  INTEGER       :: wsid
  INTEGER       :: view_ind
  INTEGER       :: ref_view_ind
  TYPE(Gpri_id) :: rel_pri
END TYPE Gfa_set_view_pri
```

Gfa_set_view_sel_crit

```
TYPE Gfa_set_view_sel_crit
  INTEGER         :: wsid
  INTEGER         :: view_ind
  TYPE(Gsel_crit) :: sel_crit
END TYPE Gfa_set_view_sel_crit
```

Gfa_set_view

```
TYPE Gfa_set_view
  INTEGER             :: wsid
  INTEGER             :: view_ind
  TYPE(Gmatrix_23)    :: view_ori_matrix
  TYPE(Gmatrix_23)    :: view_map_matrix
  TYPE(Gview_scissor) :: view_scissor
END TYPE Gfa_set_view
```

Gfa_set_ws_vis_effects

```
TYPE Gfa_set_ws_vis_effects
  INTEGER              :: wsid
  TYPE(Gvis_eff_st_id) :: vis_eff
END TYPE Gfa_set_ws_vis_effects
```

Gfa_set_ws_win_vp

```
TYPE Gfa_set_ws_win_vp
  INTEGER        :: wsid
  TYPE(Gndc_win) :: ldc_win
  TYPE(Gdc_vp)   :: dc_vp
END TYPE Gfa_set_ws_win_vp
```

Gfa_def_in_dev

```
TYPE Gfa_def_in_dev
  TYPE(Gdev_id)         :: dev_id
  TYPE(Gseq_measure_id) :: seq_measure_id
  TYPE(Gtrigger_set)    :: trigger_set
  TYPE(Ginit_val)       :: init_val
END TYPE Gfa_def_in_dev
```

Gfa_init_in_dev

```
TYPE Gfa_init_in_dev
  TYPE(Gdev_id)    :: dev_id
  TYPE(Ginit_val)  :: init_val
END TYPE Gfa_init_in_dev
```

Gfa_set_ws_sel_crit

```
TYPE Gfa_set_ws_sel_crit
  INTEGER            :: wsid
  TYPE(Gsel_crit_id) :: sel_type
  TYPE(Gsel_crit)    :: sel_crit
END TYPE Gfa_set_ws_sel_crit
```

Gfa_copy_real_pic_real_mf

```
TYPE Gfa_copy_real_pic_real_mf
  INTEGER            :: wsid
  CHARACTER, POINTER :: rld_mf_spec (:)
  INTEGER            :: picid
END TYPE Gfa_copy_real_pic_real_mf
```

Gfa_copy_blank_pic_real_mf

```
TYPE Gfa_copy_blank_pic_real_mf
  INTEGER            :: wsid
  CHARACTER, POINTER :: rld_mf_spec (:)
  INTEGER            :: picid
END TYPE Gfa_copy_blank_pic_real_mf
```

Gfa_copy_real_mf_pic_backdrop

```
TYPE Gfa_copy_real_mf_pic_backdrop
  INTEGER            :: wsid
  CHARACTER, POINTER :: rld_mf_spec (:)
  INTEGER            :: picid
END TYPE Gfa_copy_real_mf_pic_backdrop
```

Gfa_message

```
TYPE Gfa_message
  INTEGER            :: wsid
  CHARACTER, POINTER :: char_str   (:)
END TYPE Gfa_message
```

Gfa_inq_wsl_entry

```
TYPE Gfa_inq_wsl_entry
  INTEGER                :: wsid
  TYPE(Gwsl_id)          :: ent_name
  TYPE(Gset_realized_id) :: ent_type
  INTEGER                :: err_ind
  TYPE(Gwsl)             :: ent_val
END TYPE Gfa_inq_wsl_entry
```

Gfa_inq_gen_wdt_entry

```
TYPE Gfa_inq_gen_wdt_entry
  INTEGER                 :: wstype
  TYPE(Gwdt_id)           :: entry_name
  INTEGER                 :: error_ind
  TYPE(Gwdt)              :: entry_val
END TYPE Gfa_inq_gen_wdt_entry
```

Gfa_inq_spec_wdt_entry

```
TYPE Gfa_inq_spec_wdt_entry
  INTEGER                 :: wsid
  TYPE(Gwdt_id)           :: entry_name
  INTEGER                 :: error_ind
  TYPE(Gwdt)              :: entry_val
END TYPE Gfa_inq_spec_wdt_entry
```

Gfa_inq_spec_wdt_entry_st

```
TYPE Gfa_inq_spec_wdt_entry_st
  INTEGER               :: wsid
  TYPE(Gwdt_id)         :: entry_name
  INTEGER               :: error_ind
  TYPE(Gchange_flag_id) :: entry_state
END TYPE Gfa_inq_spec_wdt_entry_st
```

Gfa_res_spec_wdt_entry_st

```
TYPE Gfa_res_spec_wdt_entry_st
  INTEGER              wsid
  TYPE(Gwdt_id)        entry_name
END TYPE Gfa_res_spec_wdt_entry_st
```

Gfa_get_text_extent

```
TYPE Gfa_get_text_extent
  INTEGER                       :: wsid
  TYPE(Gwc_point)               :: text_pos
  CHARACTER, POINTER            :: char_str(:)
  INTEGER                       :: err_ind
  TYPE(Gwc_point)               :: concat_pt
  TYPE(Gwc_point)               :: xmin
  TYPE(Gwc_point)               :: xmax
  TYPE(Gwc_point)               :: ymin
  TYPE(Gwc_point)               :: ymax
END TYPE Gfa_get_text_extent
```

Gfa_eval_view_ori_matrix

```
TYPE Gfa_eval_view_ori_matrix
  TYPE(Gndc_point)                :: view_ref_pt
  TYPE(Gview_up_vec)              :: view_up_vec
  TYPE(Gmatrix_23)                :: view_ori_mat
END TYPE Gfa_eval_view_ori_matrix
```

Gfa_eval_view_map_matrix

```
TYPE Gfa_eval_view_map_matrix
  TYPE(Gndc_win)                  :: win_lim
  TYPE(Gldc_vp)                   :: vp_lim
  TYPE(Gmatrix_23)                :: view_map_mat
END TYPE Gfa_eval_view_map_matrix
```

Gfa_conv_colr

```
TYPE Gfa_conv_colr
  INTEGER                         :: wsid
  TYPE(Gcolr_spec)                :: colr_spec
  TYPE(Gcolr_model_id)            :: colr_model
  TYPE(Gcolr_spec)                :: colr_val
  TYPE(Gconv_flag_id)             :: conv_flag
END TYPE Gfa_conv_colr
```

Gfa_create_seg

```
TYPE Gfa_create_seg
  INTEGER :: seg_name
END TYPE Gfa_create_seg
```

Gfa_close_seg

```
TYPE Gfa_close_seg
  INTEGER :: NULL
END TYPE Gfa_close_seg
```

Gfa_rename_seg

```
TYPE Gfa_rename_seg
  INTEGER :: old_name
  INTEGER :: new_name
END TYPE Gfa_rename_seg
```

Gfa_del_seg

```
TYPE Gfa_del_seg
  INTEGER :: seg_name
END TYPE Gfa_del_seg
```

Gfa_del_seg_ws

```
TYPE Gfa_del_seg_ws
  INTEGER :: wsid
  INTEGER :: seg_name
END TYPE Gfa_del_seg_ws
```

Gfa_assoc_seg_ws

```
TYPE Gfa_assoc_seg_ws
  INTEGER :: wsid
  INTEGER :: seg_name
END TYPE Gfa_assoc_seg_ws
```

Gfa_copy_seg_ws

```
TYPE Gfa_copy_seg_ws
  INTEGER  :: wsid
  INTEGER  :: seg_name
END TYPE Gfa_copy_seg_ws
```

Gfa_insert_seg

```
TYPE Gfa_insert_seg
  INTEGER          :: seg_name
  TYPE(Gmatrix_23) :: tran_mat
END TYPE Gfa_insert_seg
```

Gfa_set_seg_attr

```
TYPE Gfa_set_seg_attr
  INTEGER             :: seg_name
  TYPE(Gseg_attr_val) :: seg_attr_val
END TYPE Gfa_set_seg_attr
```

Gfa_activate_ws

```
  TYPE Gfa_activate_ws
    INTEGER :: wsid
  END TYPE Gfa_activate_ws
```

Gfa_deactivate_ws

```
  TYPE Gfa_deactivate_ws
    INTEGER :: wsid
  END TYPE Gfa_deactivate_ws
```

Gfa_clear_ws

```
  TYPE Gfa_clear_ws
    INTEGER :: wsid
  END TYPE Gfa_clear_ws
```

Gfa_get_map_seg_name

```
  TYPE Gfa_get_map_seg_name
    INTEGER              :: seg_name
    INTEGER              :: map_name
  END TYPE Gfa_get_map_seg_name
```

Gfa_get_map_ws_id

```
  TYPE Gfa_get_map_ws_id
    INTEGER                      :: wsid
    TYPE(Ggen_name)              :: map_name
  END TYPE Gfa_get_map_ws_id
```

Gfunc_params

```
TYPE Gfunc_params
  TYPE(Gfn_id)                                   :: func_name
  TYPE(Gfa_open_gks), POINTER                    :: Gf_open_gks
  TYPE(Gfa_close_gks), POINTER                   :: Gf_close_gks
  TYPE(Gfa_save_gsl), POINTER                    :: Gf_save_gsl
  TYPE(Gfa_restore_gsl), POINTER                 :: Gf_restore_gsl
  TYPE(Gfa_escape), POINTER                      :: Gf_escape
  TYPE(Gfa_emergency_close_gks), POINTER         :: Gf_emergency_close_gks
  TYPE(Gfa_route), POINTER                       :: Gf_route
  TYPE(Gfa_create_out_prim), POINTER             :: Gf_create_out_prim
  TYPE(Gfa_open_stencil), POINTER                :: Gf_open_stencil
  TYPE(Gfa_close_stencil), POINTER               :: Gf_close_stencil
  TYPE(Gfa_rename_stencil), POINTER              :: Gf_rename_stencil
  TYPE(Gfa_del_stencil), POINTER                 :: Gf_del_stencil
  TYPE(Gfa_create_stencil_bndry), POINTER        :: Gf_create_stencil_bndry
  TYPE(Gfa_create_stencil_conts), POINTER        :: Gf_create_stencil_conts
  TYPE(Gfa_set_cont_attr), POINTER               :: Gf_set_cont_attr
  TYPE(Gfa_set_stencil_attr), POINTER            :: Gf_set_stencil_attr
  TYPE(Gfa_get_stencil_attr), POINTER            :: Gf_get_stencil_attr
  TYPE(Gfa_inst_stencil), POINTER                :: Gf_inst_stencil
  TYPE(Gfa_inst_stencil_path), POINTER           :: Gf_inst_stencil_path
  TYPE(Gfa_inst_stencil_seq_path), POINTER       :: Gf_inst_stencil_seq_path
```

```
TYPE(Gfa_open_tiling), POINTER              :: Gf_open_tiling
TYPE(Gfa_close_tiling), POINTER             :: Gf_close_tiling
TYPE(Gfa_rename_tiling), POINTER            :: Gf_rename_tiling
TYPE(Gfa_del_tiling), POINTER               :: Gf_del_tiling
TYPE(Gfa_create_tiling_comp), POINTER       :: Gf_create_tiling_comp
TYPE(Gfa_set_prim_attr), POINTER            :: Gf_set_prim_attr
TYPE(Gfa_add_set_names_nameset), POINTER    :: Gf_add_set_names_nameset
TYPE(Gfa_remove_set_names_nameset), POINTER :: Gf_remove_set_names_nameset
TYPE(Gfa_add_set_scissor_set), POINTER      :: Gf_add_set_scissor_set
TYPE(Gfa_remove_set_scissor_set), POINTER   :: Gf_remove_set_scissor_set
TYPE(Gfa_set_win_vp), POINTER               :: Gf_set_win_vp
TYPE(Gfa_set_vp_in_pri), POINTER            :: Gf_set_vp_in_pri
TYPE(Gfa_set_scissor_mode), POINTER         :: Gf_set_scissor_mode
TYPE(Gfa_set_norm_tran_num), POINTER        :: Gf_set_norm_tran_num
TYPE(Gfa_del_prims_ndc_pic), POINTER        :: Gf_del_prims_ndc_pic
TYPE(Gfa_add_set_names_ndc_pic), POINTER    :: Gf_add_set_names_ndc_pic
TYPE(Gfa_remove_set_names_ndc_pic), POINTER :: Gf_remove_set_names_ndc_pic
TYPE(Gfa_set_ndc_pic_prim_attr), POINTER    :: Gf_set_ndc_pic_prim_attr
TYPE(Gfa_remove_set_scissor_ndc_pic), POINTER :: Gf_remove_set_scissor_ndc_pic
TYPE(Gfa_add_set_scissor_ndc_pic), POINTER  :: Gf_add_set_scissor_ndc_pic
TYPE(Gfa_reorder_ndc_pic), POINTER          :: Gf_reorder_ndc_pic
TYPE(Gfa_copy_ndc_pic_ndc_mf ), POINTER     :: Gf_copy_ndc_pic_ndc_mf
TYPE(Gfa_copy_ndc_mf_pic_ndc_pic), POINTER  :: Gf_copy_ndc_mf_pic_ndc_pic
TYPE(Gfa_open_pic_part), POINTER            :: Gf_open_pic_part
TYPE(Gfa_close_pic_part), POINTER           :: Gf_close_pic_part
TYPE(Gfa_ar_pic_part), POINTER              :: Gf_ar_pic_part
TYPE(Gfa_ret_pic_part_ar), POINTER          :: Gf_ret_pic_part_ar
TYPE(Gfa_reopen_pic_part), POINTER          :: Gf_reopen_pic_part
TYPE(Gfa_append_pic_part), POINTER          :: Gf_append_pic_part
TYPE(Gfa_rename_pic_part), POINTER          :: Gf_rename_pic_part
TYPE(Gfa_del_pic_part), POINTER             :: Gf_del_pic_part
TYPE(Gfa_copy_pic_part_pps), POINTER        :: Gf_copy_pic_part_pps
TYPE(Gfa_copy_ndc_pic_pps), POINTER         :: Gf_copy_ndc_pic_pps
TYPE(Gfa_set_in_dev_mode), POINTER          :: Gf_set_in_dev_mode
TYPE(Gfa_req_in), POINTER                   :: Gf_req_in
TYPE(Gfa_sample_in), POINTER                :: Gf_sample_in
TYPE(Gfa_await_in), POINTER                 :: Gf_await_in
TYPE(Gfa_flush_dev_events ), POINTER        :: Gf_flush_dev_events
TYPE(Gfa_set_font_ind_map), POINTER         :: Gf_set_font_ind_map
TYPE(Gfa_get_glyph_name), POINTER           :: Gf_get_glyph_name
TYPE(Gfa_set_char_code), POINTER            :: Gf_set_char_code
TYPE(Gfa_audit), POINTER                    :: Gf_audit
TYPE(Gfa_write_user_rec_audit), POINTER     :: Gf_write_user_rec_audit
TYPE(Gfa_playback ), POINTER                :: Gf_playback
TYPE(Gfa_read_item_func_name_audit), POINTER :: Gf_read_item_func_name_audit
TYPE(Gfa_proc_audit_item), POINTER          :: Gf_proc_audit_item
TYPE(Gfa_inq_op_st_entry), POINTER          :: Gf_inq_op_st_entry
TYPE(Gfa_inq_gdt_entry), POINTER            :: Gf_inq_gdt_entry
TYPE(Gfa_inq_gsl_entry), POINTER            :: Gf_inq_gsl_entry
TYPE(Gfa_inq_in_overf), POINTER             :: Gf_inq_in_overf
TYPE(Gfa_eval_tran_matrix), POINTER         :: Gf_eval_tran_matrix
TYPE(Gfa_accum_tran_matrix), POINTER        :: Gf_accum_tran_matrix
TYPE(Gfa_eval_circle), POINTER              :: Gf_eval_circle
TYPE(Gfa_eval_ell), POINTER                 :: Gf_eval_ell
TYPE(Gfa_eval_circ_arc_3_point), POINTER    :: Gf_eval_circ_arc_3_point
TYPE(Gfa_eval_circ_arc_ctr), POINTER        :: Gf_eval_circ_arc_ctr
TYPE(Gfa_eval_ell_arc), POINTER             :: Gf_eval_ell_arc
```

```
  TYPE(Gfa_eval_hyp_arc), POINTER              :: Gf_eval_hyp_arc
  TYPE(Gfa_eval_par_arc), POINTER              :: Gf_eval_par_arc
  TYPE(Gfa_eval_wc_ord), POINTER               :: Gf_eval_wc_ord
  TYPE(Gfa_open_ws), POINTER                   :: Gf_open_ws
  TYPE(Gfa_close_ws), POINTER                  :: Gf_close_ws
  TYPE(Gfa_save_wsl), POINTER                  :: Gf_save_wsl
  TYPE(Gfa_restore_wsl), POINTER               :: Gf_restore_wsl
  TYPE(Gfa_get_ws_status), POINTER             :: Gf_get_ws_status
  TYPE(Gfa_remove_backdrop), POINTER           :: Gf_remove_backdrop
  TYPE(Gfa_set_rep), POINTER                   :: Gf_set_rep
  TYPE(Gfa_set_view_pri), POINTER              :: Gf_set_view_pri
  TYPE(Gfa_set_view_sel_crit), POINTER         :: Gf_set_view_sel_crit
  TYPE(Gfa_set_view), POINTER                  :: Gf_set_view
  TYPE(Gfa_set_ws_vis_effects), POINTER        :: Gf_set_ws_vis_effects
  TYPE(Gfa_set_ws_win_vp), POINTER             :: Gf_set_ws_win_vp
  TYPE(Gfa_def_in_dev), POINTER                :: Gf_def_in_dev
  TYPE(Gfa_init_in_dev), POINTER               :: Gf_init_in_dev
  TYPE(Gfa_set_ws_sel_crit), POINTER           :: Gf_set_ws_sel_crit
  TYPE(Gfa_copy_real_pic_real_mf), POINTER     :: Gf_copy_real_pic_real_mf
  TYPE(Gfa_copy_blank_pic_real_mf), POINTER    :: Gf_copy_blank_pic_real_mf
  TYPE(Gfa_copy_real_mf_pic_backdrop), POINTER :: Gf_copy_real_mf_pic_backdrop
  TYPE(Gfa_message), POINTER                   :: Gf_message
  TYPE(Gfa_inq_wsl_entry), POINTER             :: Gf_inq_wsl_entry
  TYPE(Gfa_inq_gen_wdt_entry), POINTER         :: Gf_inq_gen_wdt_entry
  TYPE(Gfa_inq_spec_wdt_entry), POINTER        :: Gf_inq_spec_wdt_entry
  TYPE(Gfa_inq_spec_wdt_entry_st), POINTER     :: Gf_inq_spec_wdt_entry_st
  TYPE(Gfa_res_spec_wdt_entry_st), POINTER     :: Gf_res_spec_wdt_entry_st
  TYPE(Gfa_get_text_extent), POINTER           :: Gf_get_text_extent
  TYPE(Gfa_eval_view_ori_matrix), POINTER      :: Gf_eval_view_ori_matrix
  TYPE(Gfa_eval_view_map_matrix), POINTER      :: Gf_eval_view_map_matrix
  TYPE(Gfa_conv_colr), POINTER                 :: Gf_conv_colr
  TYPE(Gfa_create_seg), POINTER                :: Gf_create_seg
  TYPE(Gfa_close_seg), POINTER                 :: Gf_close_seg
  TYPE(Gfa_rename_seg), POINTER                :: Gf_rename_seg
  TYPE(Gfa_del_seg), POINTER                   :: Gf_del_seg
  TYPE(Gfa_del_seg_ws), POINTER                :: Gf_del_seg_ws
  TYPE(Gfa_assoc_seg_ws), POINTER              :: Gf_assoc_seg_ws
  TYPE(Gfa_copy_seg_ws), POINTER               :: Gf_copy_seg_ws
  TYPE(Gfa_insert_seg), POINTER                :: Gf_insert_seg
  TYPE(Gfa_set_seg_attr), POINTER              :: Gf_set_seg_attr
  TYPE(Gfa_activate_ws), POINTER               :: Gf_activate_ws
  TYPE(Gfa_deactivate_ws), POINTER             :: Gf_deactivate_ws
  TYPE(Gfa_clear_ws), POINTER                  :: Gf_clear_ws
  TYPE(Gfa_get_map_seg_name), POINTER          :: Gf_get_map_seg_name
  TYPE(Gfa_get_map_ws_id), POINTER             :: Gf_get_map_ws_id
END TYPE Gfunc_params
```

7 Enumeration Types

The following section contains a method of mapping GKS enumeration types to Fortran variable names. A type is only an enumeration type if and only if its name is prefixed by "G" and ends with "_id".

Mnemonic Fortran names and their values for GKS ENUMERATION type values:

7.1 Data types for GKS control

Error status

```
TYPE Gerr_st_id
  INTEGER :: i
END TYPE Gerr_st_id

TYPE(Gerr_st_id), PARAMETER :: &
Gerr_st_on                       = Gerr_st_id(0), &
Gerr_st_off                      = Gerr_st_id(1)
```

Function name

```
TYPE Gfn_id
  INTEGER :: i
END TYPE Gfn_id

TYPE(Gfn_id), PARAMETER :: &
Gfn_gopen_gks                        = Gfn_id(1), &
Gfn_gclose_gks                       = Gfn_id(2), &
Gfn_gsave_gsl                        = Gfn_id(3), &
Gfn_grestore_gsl                     = Gfn_id(4), &
Gfn_gescape                          = Gfn_id(5), &
Gfn_gemergency_close_gks             = Gfn_id(6), &
Gfn_groute                           = Gfn_id(7), &
Gfn_gcreate_out_prim                 = Gfn_id(8), &
Gfn_gopen_stencil                    = Gfn_id(9), &
Gfn_gclose_stencil                   = Gfn_id(10), &
Gfn_grename_stencil                  = Gfn_id(11), &
Gfn_gdel_stencil                     = Gfn_id(12), &
Gfn_gcreate_stencil_bndry            = Gfn_id(13), &
Gfn_gcreate_stencil_conts            = Gfn_id(14), &
Gfn_gset_cont_attr                   = Gfn_id(15), &
Gfn_gset_stencil_attr                = Gfn_id(16), &
Gfn_gget_stencil_attr                = Gfn_id(17), &
Gfn_ginst_stencil                    = Gfn_id(18), &
Gfn_ginst_stencil_path               = Gfn_id(19), &
Gfn_ginst_stencil_seq_path           = Gfn_id(20), &
Gfn_gopen_tiling                     = Gfn_id(21), &
Gfn_gclose_tiling                    = Gfn_id(22), &
Gfn_grename_tiling                   = Gfn_id(23), &
Gfn_gdel_tiling                      = Gfn_id(24), &
Gfn_gcreate_tiling_comp              = Gfn_id(25), &
Gfn_gset_prim_attr                   = Gfn_id(26), &
Gfn_gadd_set_names_nameset           = Gfn_id(27), &
Gfn_gremove_set_names_nameset        = Gfn_id(28), &
```

```
Gfn_gadd_set_scissor_set          = Gfn_id(29), &
Gfn_grem_set_scissor_set          = Gfn_id(30), &
Gfn_gremove_set_scissor_set       = Gfn_id(31), &
Gfn_gset_win_vp                   = Gfn_id(32), &
Gfn_gset_vp_in_pri                = Gfn_id(33), &
Gfn_gset_scissor_mode             = Gfn_id(34), &
Gfn_gset_norm_tran_num            = Gfn_id(35), &
Gfn_gdel_prims_ndc_pic            = Gfn_id(36), &
Gfn_gadd_set_names_ndc_pic        = Gfn_id(37), &
Gfn_gremove_set_names_ndc_pic     = Gfn_id(38)

TYPE(Gfn_id), PARAMETER :: &
Gfn_gset_ndc_pic_prim_attr      = Gfn_id(39), &
Gfn_gremove_set_scissor_ndc_pic = Gfn_id(40), &
Gfn_gadd_set_scissor_ndc_pic    = Gfn_id(41), &
Gfn_greorder_ndc_pic            = Gfn_id(42), &
Gfn_gcopy_ndc_pic_ndc_mf        = Gfn_id(43), &
Gfn_gcopy_ndc_mf_pic_ndc_pic    = Gfn_id(44), &
Gfn_gopen_pic_part              = Gfn_id(45), &
Gfn_gclose_pic_part             = Gfn_id(46), &
Gfn_gar_pic_part                = Gfn_id(47), &
Gfn_gret_pic_part_ar            = Gfn_id(48), &
Gfn_greopen_pic_part            = Gfn_id(49), &
Gfn_gappend_pic_part            = Gfn_id(50), &
Gfn_grename_pic_part            = Gfn_id(51), &
Gfn_gdel_pic_part               = Gfn_id(52), &
Gfn_gcopy_pic_part_pps          = Gfn_id(53), &
Gfn_gcopy_ndc_pic_pps           = Gfn_id(54), &
Gfn_gset_in_dev_mode            = Gfn_id(55), &
Gfn_greq_in                     = Gfn_id(56), &
Gfn_gsample_in                  = Gfn_id(57), &
Gfn_gawait_in                   = Gfn_id(58), &
Gfn_gflush_dev_events           = Gfn_id(59), &
Gfn_gset_font_ind_map           = Gfn_id(60), &
Gfn_gget_glyph_name             = Gfn_id(61), &
Gfn_gset_char_code              = Gfn_id(62), &
Gfn_gaudit                      = Gfn_id(63), &
Gfn_gwrite_user_rec_audit       = Gfn_id(64), &
Gfn_gplayback                   = Gfn_id(65), &
Gfn_gread_item_func_name_audit  = Gfn_id(66), &
Gfn_gread_item_audit            = Gfn_id(67), &
Gfn_gproc_audit_item            = Gfn_id(68), &
Gfn_ginq_op_st_entry            = Gfn_id(69), &
Gfn_ginq_gdt_entry              = Gfn_id(70), &
Gfn_ginq_gsl_entry              = Gfn_id(71), &
Gfn_ginq_in_overf               = Gfn_id(72), &
Gfn_geval_tran_matrix           = Gfn_id(73), &
Gfn_gaccum_tran_matrix          = Gfn_id(74), &
Gfn_geval_circle                = Gfn_id(75), &
Gfn_geval_ell                   = Gfn_id(76)

TYPE(Gfn_id), PARAMETER :: &
Gfn_geval_circ_arc_3_point     = Gfn_id(77), &
Gfn_geval_circ_arc_ctr         = Gfn_id(78), &
Gfn_geval_ell_arc              = Gfn_id(79), &
Gfn_geval_hyp_arc              = Gfn_id(80), &
Gfn_geval_par_arc              = Gfn_id(81), &
```

```
Gfn_geval_wc_ord                = Gfn_id(82), &
Gfn_gopen_ws                    = Gfn_id(83), &
Gfn_gclose_ws                   = Gfn_id(84), &
Gfn_gsave_wsl                   = Gfn_id(85), &
Gfn_grestore_wsl                = Gfn_id(86), &
Gfn_gget_ws_status              = Gfn_id(87), &
Gfn_gremove_backdrop            = Gfn_id(88), &
Gfn_gset_rep                    = Gfn_id(89), &
Gfn_gset_view_pri               = Gfn_id(90), &
Gfn_gset_view_sel_crit          = Gfn_id(91), &
Gfn_gset_view                   = Gfn_id(92), &
Gfn_gset_ws_vis_effects         = Gfn_id(93), &
Gfn_gset_ws_win_vp              = Gfn_id(94), &
Gfn_gdef_in_dev                 = Gfn_id(95), &
Gfn_ginit_in_dev                = Gfn_id(96), &
Gfn_gset_ws_sel_crit            = Gfn_id(97), &
Gfn_gcopy_real_pic_real_mf      = Gfn_id(98), &
Gfn_gcopy_blank_pic_real_mf     = Gfn_id(99), &
Gfn_gcopy_real_mf_pic_backdrop  = Gfn_id(100)

TYPE(Gfn_id), PARAMETER :: &
Gfn_gmessage                    = Gfn_id(101), &
Gfn_ginq_wsl_entry              = Gfn_id(102), &
Gfn_ginq_gen_wdt_entry          = Gfn_id(103), &
Gfn_ginq_spec_wdt_entry         = Gfn_id(104), &
Gfn_ginq_spec_wdt_entry_st      = Gfn_id(105), &
Gfn_gres_spec_wdt_entry_st      = Gfn_id(106), &
Gfn_gget_text_extent            = Gfn_id(107), &
Gfn_geval_view_ori_matrix       = Gfn_id(108), &
Gfn_geval_view_map_matrix       = Gfn_id(109), &
Gfn_gconv_col                   = Gfn_id(110), &
Gfn_gcreate_seg                 = Gfn_id(111), &
Gfn_gclose_seg                  = Gfn_id(112), &
Gfn_grename_seg                 = Gfn_id(113), &
Gfn_gdel_seg                    = Gfn_id(114), &
Gfn_gdel_seg_ws                 = Gfn_id(115), &
Gfn_gassoc_seg_ws               = Gfn_id(116), &
Gfn_gcopy_seg_ws                = Gfn_id(117), &
Gfn_ginsert_seg                 = Gfn_id(118), &
Gfn_gset_seg_attr               = Gfn_id(119), &
Gfn_gactivate_ws                = Gfn_id(120), &
Gfn_gdeactivate_ws              = Gfn_id(121), &
Gfn_gclear_ws                   = Gfn_id(122), &
Gfn_gget_map_seg_name           = Gfn_id(123), &
Gfn_gget_map_ws_id              = Gfn_id(124)
```

Routing of output primitives

```
TYPE Groute_dir_id
  INTEGER :: i
END TYPE Groute_dir_id

TYPE(Groute_dir_id), PARAMETER :: &
Groute_dir_ndc          = Groute_dir_id(0), &
Groute_dir_backdrop     = Groute_dir_id(1)
```

7.2 Data types for output primitives

Colour array specifier

```
TYPE Gcolr_array_spec_id
  INTEGER :: i
END TYPE Gcolr_array_spec_id

TYPE(Gcolr_array_spec_id), PARAMETER :: &
Gcolr_array_spec_indirectarray  = Gcolr_array_spec_id(0), &
Gcolr_array_spec_rgbarray       = Gcolr_array_spec_id(1), &
Gcolr_array_spec_cieluvarray    = Gcolr_array_spec_id(2), &
Gcolr_array_spec_hsvarray       = Gcolr_array_spec_id(3), &
Gcolr_array_spec_hlsarray       = Gcolr_array_spec_id(4)
```

Control points

```
TYPE Gctrl_points_id
  INTEGER :: i
END TYPE Gctrl_points_id

TYPE(Gctrl_points_id), PARAMETER :: &
Gctrl_points_hcp = Gctrl_points_id(0), &
Gctrl_points_cp  = Gctrl_points_id(1)
```

Output Primitive type

```
TYPE Gprim_param_id
  INTEGER :: i
END TYPE Gprim_param_id

TYPE(Gprim_param_id), PARAMETER :: &
Gprim_param_setline       = Gprim_param_id(0), &
Gprim_param_setnurb       = Gprim_param_id(1), &
Gprim_param_conicsec      = Gprim_param_id(2), &
Gprim_param_marker        = Gprim_param_id(3), &
Gprim_param_setfill       = Gprim_param_id(4), &
Gprim_param_setellipsec   = Gprim_param_id(5), &
Gprim_param_setellipseg   = Gprim_param_id(6), &
Gprim_param_setellipdisc  = Gprim_param_id(7), &
Gprim_param_setclosednurb = Gprim_param_id(8), &
Gprim_param_text          = Gprim_param_id(9), &
Gprim_param_cellarray     = Gprim_param_id(10), &
Gprim_param_design        = Gprim_param_id(11), &
Gprim_param_gdp           = Gprim_param_id(12)
```

Sense Flag

```
TYPE Gsense_flag_id
  INTEGER :: i
END TYPE Gsense_flag_id

TYPE(Gsense_flag_id), PARAMETER :: &
Gsense_flag_clockwise      = Gsense_flag_id(0), &
Gsense_flag_anticlockwise  = Gsense_flag_id(1)
```

7.3 Data types for output attributes

Output Attribute Type

```
TYPE Gprim_attr_id
  INTEGER :: i
END TYPE Gprim_attr_id

TYPE(Gprim_attr_id), PARAMETER :: &
Gprim_attr_pickid             = Gprim_attr_id(0), &
Gprim_attr_nameset            = Gprim_attr_id(1), &
Gprim_attr_scissorset         = Gprim_attr_id(2), &
Gprim_attr_globaltran         = Gprim_attr_id(3), &
Gprim_attr_loctran            = Gprim_attr_id(4), &
Gprim_attr_patsize            = Gprim_attr_id(5), &
Gprim_attr_patrefpt           = Gprim_attr_id(6), &
Gprim_attr_textheight         = Gprim_attr_id(7), &
Gprim_attr_textupvec          = Gprim_attr_id(8), &
Gprim_attr_textskew           = Gprim_attr_id(9), &
Gprim_attr_textpath           = Gprim_attr_id(10), &
Gprim_attr_textalign          = Gprim_attr_id(11), &
Gprim_attr_curveind           = Gprim_attr_id(12), &
Gprim_attr_curveasfs          = Gprim_attr_id(13), &
Gprim_attr_curvetype          = Gprim_attr_id(14), &
Gprim_attr_curvewidth         = Gprim_attr_id(15), &
Gprim_attr_curvecolrspec      = Gprim_attr_id(16), &
Gprim_attr_markerind          = Gprim_attr_id(17), &
Gprim_attr_markerasfs         = Gprim_attr_id(18), &
Gprim_attr_markertype         = Gprim_attr_id(19), &
Gprim_attr_markersize         = Gprim_attr_id(20), &
Gprim_attr_markercolrspec     = Gprim_attr_id(21), &
Gprim_attr_areaind            = Gprim_attr_id(22), &
Gprim_attr_areaasfs           = Gprim_attr_id(23), &
Gprim_attr_intstyle           = Gprim_attr_id(24), &
Gprim_attr_intstyleind        = Gprim_attr_id(25), &
Gprim_attr_intcolrspec        = Gprim_attr_id(26), &
Gprim_attr_edgeflag           = Gprim_attr_id(27), &
Gprim_attr_edgetype           = Gprim_attr_id(28), &
Gprim_attr_edgewidth          = Gprim_attr_id(29), &
Gprim_attr_edgecolrspec       = Gprim_attr_id(30), &
Gprim_attr_charind            = Gprim_attr_id(31), &
Gprim_attr_charasfs           = Gprim_attr_id(32), &
Gprim_attr_charfontprec       = Gprim_attr_id(33), &
Gprim_attr_charexpan          = Gprim_attr_id(34), &
Gprim_attr_charspace          = Gprim_attr_id(35), &
Gprim_attr_charcolrspec       = Gprim_attr_id(36)
```

Aspect Source Flag

```
TYPE Gasf_val_id
  INTEGER :: i
END TYPE Gasf_val_id

TYPE(Gasf_val_id), PARAMETER :: &
Gasf_val_bun   = Gasf_val_id(0), &
Gasf_val_indiv = Gasf_val_id(1)
```

Clipping Indicator

```
TYPE Gclip_ind_id
  INTEGER :: i
END TYPE Gclip_ind_id

TYPE(Gclip_ind_id), PARAMETER :: &
Gclip_ind_clip   = Gclip_ind_id(0), &
Gclip_ind_noclip = Gclip_ind_id(1)
```

Colour specifier

```
TYPE Gcolr_spec_id
  INTEGER :: i
END TYPE Gcolr_spec_id

TYPE(Gcolr_spec_id), PARAMETER :: &
Gcolr_spec_indirect = Gcolr_spec_id(0), &
Gcolr_spec_rgb      = Gcolr_spec_id(1), &
Gcolr_spec_cieluv   = Gcolr_spec_id(2), &
Gcolr_spec_hsv      = Gcolr_spec_id(3), &
Gcolr_spec_hls      = Gcolr_spec_id(4)
```

Edge Flag

```
TYPE Gedge_flag_id
  INTEGER :: i
END TYPE Gedge_flag_id

TYPE(Gedge_flag_id), PARAMETER :: &
Gedge_flag_off = Gedge_flag_id(0), &
Gedge_flag_on  = Gedge_flag_id(1)
```

Generic name

```
TYPE Ggen_name_id
  INTEGER :: i
END TYPE Ggen_name_id
```

```
TYPE(Ggen_name_id), PARAMETER :: &
Ggen_name_wsid    = Ggen_name_id(0), &
Ggen_name_segname = Ggen_name_id(1), &
Ggen_name_name    = Ggen_name_id(2)
```

Horizontal Alignment

```
TYPE Ghor_align_id
  INTEGER :: i
END TYPE Ghor_align_id

TYPE(Ghor_align_id), PARAMETER :: &
Ghor_align_norm   = Ghor_align_id(0), &
Ghor_align_left   = Ghor_align_id(1), &
Ghor_align_centre = Ghor_align_id(2), &
Ghor_align_right  = Ghor_align_id(3)
```

Interior Style

```
TYPE Gint_style_id
  INTEGER :: i
END TYPE Gint_style_id

TYPE(Gint_style_id), PARAMETER :: &
Gint_style_hollow  = Gint_style_id(0), &
Gint_style_solid   = Gint_style_id(1), &
Gint_style_pat     = Gint_style_id(2), &
Gint_style_hatch   = Gint_style_id(3), &
Gint_style_empty   = Gint_style_id(4)
```

Precision

```
TYPE Gprec_id
  INTEGER :: i
END TYPE Gprec_id

TYPE(Gprec_id), PARAMETER :: &
Gprec_string = Gprec_id(0), &
Gprec_char   = Gprec_id(1), &
Gprec_stroke = Gprec_id(2)
```

Shield Indicator

```
TYPE Gshield_ind_id
  INTEGER :: i
END TYPE Gshield_ind_id

TYPE(Gshield_ind_id), PARAMETER :: &
Gshield_ind_shield   = Gshield_ind_id(0), &
Gshield_ind_noshield = Gshield_ind_id(1)
```

Text Path

```
TYPE Gtext_path_id
  INTEGER :: i
END TYPE Gtext_path_id

TYPE(Gtext_path_id), PARAMETER :: &
Gtext_path_right = Gtext_path_id(0), &
Gtext_path_left  = Gtext_path_id(1), &
Gtext_path_up    = Gtext_path_id(2), &
Gtext_path_down  = Gtext_path_id(3)
```

Vertical Alignment

```
TYPE Gvert_align_id
  INTEGER :: i
END TYPE Gvert_align_id

TYPE(Gvert_align_id), PARAMETER :: &
Gvert_align_norm   = Gvert_align_id(0), &
Gvert_align_top    = Gvert_align_id(1), &
Gvert_align_cap    = Gvert_align_id(2), &
Gvert_align_half   = Gvert_align_id(3), &
Gvert_align_base   = Gvert_align_id(4), &
Gvert_align_bottom = Gvert_align_id(5)
```

8 Data types for transformations

Priority

```
TYPE Gpri_id
  INTEGER :: i
END TYPE Gpri_id

TYPE(Gpri_id), PARAMETER :: &
Gpri_higher = Gpri_id(0), &
Gpri_lower  = Gpri_id(1)
```

Scissor mode

```
TYPE Gscissor_mode_id
  INTEGER :: i
END TYPE Gscissor_mode_id

TYPE(Gscissor_mode_id), PARAMETER :: &
Gscissor_mode_locus = Gscissor_mode_id(0), &
Gscissor_mode_shape = Gscissor_mode_id(1)
```

8.1 Data types for NDC picture

Position

```
TYPE Gpos_id
  INTEGER :: i
END TYPE Gpos_id

TYPE(Gpos_id), PARAMETER :: &
Gpos_front = Gpos_id(0), &
Gpos_back  = Gpos_id(1)
```

8.2 Data types for picture parts and archives

Scissor Select

```
TYPE Gscissor_sel_id
  INTEGER :: i
END TYPE Gscissor_sel_id

TYPE(Gscissor_sel_id), PARAMETER :: &
Gscissor_sel_change = Gscissor_sel_id(0), &
Gscissor_sel_leave  = Gscissor_sel_id(1)
```

Transformation Mode

```
TYPE Gtran_mode_id
  INTEGER :: i
END TYPE Gtran_mode_id

TYPE(Gtran_mode_id), PARAMETER :: &
Gtran_mode_replace = Gtran_mode_id(0), &
Gtran_mode_pre     = Gtran_mode_id(1), &
Gtran_mode_post    = Gtran_mode_id(2)
```

8.3 Data types for utility functions

Ordinate selector

```
TYPE Gord_sel_id
  INTEGER :: i
END TYPE Gord_sel_id

TYPE(Gord_sel_id), PARAMETER :: &
Gord_sel_x = Gord_sel_id(0), &
Gord_sel_y = Gord_sel_id(1)
```

WCNDC Point

```
TYPE Gwcndc_point_id
  INTEGER :: i
END TYPE Gwcndc_point_id

TYPE(Gwcndc_point_id), PARAMETER :: &
Gwcndc_point_wc  = Gwcndc_point_id(0), &
Gwcndc_point_ndc = Gwcndc_point_id(1)
```

8.4 Data types for segments

Segment attribute name

```
TYPE Gseg_attr_name_id
  INTEGER :: i
END TYPE Gseg_attr_name_id

TYPE(Gseg_attr_name_id), PARAMETER :: &
Gseg_attr_name_vis     = Gseg_attr_name_id(0), &
Gseg_attr_name_highl   = Gseg_attr_name_id(1), &
Gseg_attr_name_det     = Gseg_attr_name_id(2), &
Gseg_attr_name_segtran = Gseg_attr_name_id(3), &
Gseg_attr_name_segpri  = Gseg_attr_name_id(4)
```

Detectability

```
TYPE Gdet_id
  INTEGER :: i
END TYPE Gdet_id

TYPE(Gdet_id), PARAMETER :: &
Gdet_undet = Gdet_id(0), &
Gdet_det   = Gdet_id(1)
```

Highlighting

```
TYPE Ghighl_id
  INTEGER :: i
END TYPE Ghighl_id

TYPE(Ghighl_id), PARAMETER :: &
Ghighl_norm  = Ghighl_id(0), &
Ghighl_highl = Ghighl_id(1)
```

Visibility

```
TYPE Gvis_id
  INTEGER :: i
END TYPE Gvis_id

TYPE(Gvis_id), PARAMETER :: &
Gvis_vis   = Gvis_id(0), &
Gvis_invis = Gvis_id(1)
```

8.5 Data types for input

Choice Status

```
TYPE Gchoice_st_id
  INTEGER :: i
END TYPE Gchoice_st_id

TYPE(Gchoice_st_id), PARAMETER :: &
Gchoice_st_ok       = Gchoice_st_id(0), &
Gchoice_st_nochoice = Gchoice_st_id(1)
```

Device identifier value set

```
TYPE Gdev_id_val_set_id
  INTEGER :: i
END TYPE Gdev_id_val_set_id

TYPE(Gdev_id_val_set_id), PARAMETER :: &
Gdev_id_val_set_devidval = Gdev_id_val_set_id(0), &
Gdev_id_val_set_none     = Gdev_id_val_set_id(1)
```

Echo Switch

```
TYPE Gecho_switch_id
  INTEGER :: i
END TYPE Gecho_switch_id

TYPE(Gecho_switch_id), PARAMETER :: &
Gecho_switch_echo   = Gecho_switch_id(0), &
Gecho_switch_noecho = Gecho_switch_id(1)
```

Initial value

```
TYPE Ginit_val_id
  INTEGER :: i
END TYPE Ginit_val_id

TYPE(Ginit_val_id), PARAMETER :: &
Ginit_val_initval      = Ginit_val_id(0), &
Ginit_val_pet          = Ginit_val_id(1), &
Ginit_val_echoflag     = Ginit_val_id(2), &
Ginit_val_echoarea     = Ginit_val_id(3), &
Ginit_val_datarec      = Ginit_val_id(4), &
Ginit_val_all          = Ginit_val_id(5)
```

Input Status

```
TYPE Gin_st_id
  INTEGER :: i
END TYPE Gin_st_id

TYPE(Gin_st_id), PARAMETER :: &
Gin_st_ok   = Gin_st_id(0), &
Gin_st_none = Gin_st_id(1)
```

Measure class

```
TYPE Gmeasure_class_id
  INTEGER :: i
END TYPE Gmeasure_class_id

TYPE(Gmeasure_class_id), PARAMETER :: &
Gmeasure_class_loc       = Gmeasure_class_id(0), &
Gmeasure_class_stroke    = Gmeasure_class_id(1), &
Gmeasure_class_val       = Gmeasure_class_id(2), &
Gmeasure_class_choice    = Gmeasure_class_id(3), &
Gmeasure_class_pick      = Gmeasure_class_id(4), &
Gmeasure_class_string    = Gmeasure_class_id(5), &
Gmeasure_class_composite = Gmeasure_class_id(6)
```

Operating Mode

```
TYPE Gop_mode_id
  INTEGER :: i
END TYPE Gop_mode_id

TYPE(Gop_mode_id), PARAMETER :: &
Gop_mode_req     = Gop_mode_id(0), &
Gop_mode_sample  = Gop_mode_id(1), &
Gop_mode_event   = Gop_mode_id(2)
```

Pick status

```
TYPE Gpick_st_id
  INTEGER :: i
END TYPE Gpick_st_id

TYPE(Gpick_st_id), PARAMETER :: &
Gpick_st_ok     = Gpick_st_id(0), &
Gpick_st_nopick = Gpick_st_id(1)
```

8.6 Data types for workstation control

Change flag

```
TYPE Gchange_flag_id
  INTEGER :: i
```

```
END TYPE Gchange_flag_id

TYPE(Gchange_flag_id), PARAMETER :: &
Gchange_flag_changed    = Gchange_flag_id(0), &
Gchange_flag_notchanged = Gchange_flag_id(1)
```

Colour availability

```
TYPE Gcolr_avail_id
  INTEGER :: i
END TYPE Gcolr_avail_id

TYPE(Gcolr_avail_id), PARAMETER :: &
Gcolr_avail_colr = Gcolr_avail_id(0), &
Gcolr_avail_mono = Gcolr_avail_id(1)
```

Device Coordinates units

```
TYPE Gdc_units_id
  INTEGER :: i
END TYPE Gdc_units_id

TYPE(Gdc_units_id), PARAMETER :: &
Gdc_units_metres = Gdc_units_id(0), &
Gdc_units_other  = Gdc_units_id(1)
```

Display kind

```
TYPE Gdisp_kind_id
  INTEGER :: i
END TYPE Gdisp_kind_id

TYPE(Gdisp_kind_id), PARAMETER :: &
Gdisp_kind_vec    = Gdisp_kind_id(0), &
Gdisp_kind_raster = Gdisp_kind_id(1), &
Gdisp_kind_other  = Gdisp_kind_id(2)
```

Workstation status

```
TYPE Gws_st_id
  INTEGER :: i
END TYPE Gws_st_id

TYPE(Gws_st_id), PARAMETER :: &
Gws_st_uptodate    = Gws_st_id(0), &
Gws_st_notuptodate = Gws_st_id(1)
```

Colour Model

```
TYPE Gcolr_model_id
  INTEGER :: i
END TYPE Gcolr_model_id

TYPE(Gcolr_model_id), PARAMETER :: &
Gcolr_model_rgb    = Gcolr_model_id(0), &
Gcolr_model_cieluv = Gcolr_model_id(1), &
Gcolr_model_hsv    = Gcolr_model_id(2), &
Gcolr_model_hls    = Gcolr_model_id(3)
```

Colour coordinates

```
TYPE Gcolr_coord_id
  INTEGER :: i
END TYPE Gcolr_coord_id

TYPE(Gcolr_coord_id), PARAMETER :: &
Gcolr_coord_rgb    = Gcolr_coord_id(0), &
Gcolr_coord_cieluv = Gcolr_coord_id(1), &
Gcolr_coord_hsv    = Gcolr_coord_id(2), &
Gcolr_coord_hls    = Gcolr_coord_id(3)
```

Conversion Flag

```
TYPE Gconv_flag_id
  INTEGER :: i
END TYPE Gconv_flag_id

TYPE(Gconv_flag_id), PARAMETER :: &
Gconv_flag_done    = Gconv_flag_id(0), &
Gconv_flag_approx  = Gconv_flag_id(1), &
Gconv_flag_notconv = Gconv_flag_id(2)
```

Visual effects state

```
TYPE Gvis_eff_st_id
  INTEGER :: i
END TYPE Gvis_eff_st_id

TYPE(Gvis_eff_st_id), PARAMETER :: &
Gvis_eff_st_suppress    = Gvis_eff_st_id(0), &
Gvis_eff_st_allow       = Gvis_eff_st_id(1)
```

Output Primitive Representation

```
TYPE Grep_id
  INTEGER :: i
END TYPE Grep_id

TYPE(Grep_id), PARAMETER :: &
```

```
Grep_curve     = Grep_id(0), &
Grep_marker    = Grep_id(1), &
Grep_area      = Grep_id(2), &
Grep_pat       = Grep_id(3), &
Grep_char      = Grep_id(4), &
Grep_colr      = Grep_id(5), &
Grep_undefined = Grep_id(6)
```

Selection criteria type

```
TYPE Gsel_crit_id
  INTEGER :: i
END TYPE Gsel_crit_id

TYPE(Gsel_crit_id), PARAMETER :: &
Gsel_crit_vis    = Gsel_crit_id(0), &
Gsel_crit_highl  = Gsel_crit_id(1), &
Gsel_crit_det    = Gsel_crit_id(2)
```

8.7 Data types for inquiry functions

Set or Realized

```
TYPE Gset_realized_id
  INTEGER :: i
END TYPE Gset_realized_id

TYPE(Gset_realized_id), PARAMETER :: &
Gset_realized_set      = Gset_realized_id(0), &
Gset_realized_realized = Gset_realized_id(1)
```

8.8 Data types for operating state

GKS Operating State

```
TYPE Ggks_op_st_id
  INTEGER :: i
END TYPE Ggks_op_st_id

TYPE(Ggks_op_st_id), PARAMETER :: &
Ggks_op_st_gkcl = Ggks_op_st_id(0), &
Ggks_op_st_gkop = Ggks_op_st_id(1)
```

Picture Part Operating State

```
TYPE Gpic_part_op_st_id
  INTEGER :: i
END TYPE Gpic_part_op_st_id

TYPE(Gpic_part_op_st_id), PARAMETER :: &
Gpic_part_op_st_ppcl = Gpic_part_op_st_id(0), &
Gpic_part_op_st_ppop = Gpic_part_op_st_id(1)
```

Segment operating state

```
TYPE Gseg_op_st_id
  INTEGER :: i
END TYPE Gseg_op_st_id

TYPE(Gseg_op_st_id), PARAMETER :: &
Gseg_op_st_sgcl = Gseg_op_st_id(0), &
Gseg_op_st_sgop = Gseg_op_st_id(1)
```

Tiling Operating State

```
TYPE Gtiling_op_st_id
  INTEGER :: i
END TYPE Gtiling_op_st_id

TYPE(Gtiling_op_st_id), PARAMETER :: &
Gtiling_op_st_tlcl = Gtiling_op_st_id(0), &
Gtiling_op_st_tlop = Gtiling_op_st_id(1)
```

Stencil Operating State

```
TYPE Gstencil_op_st_id
  INTEGER :: i
END TYPE Gstencil_op_st_id

TYPE(Gstencil_op_st_id), PARAMETER :: &
Gstencil_op_st_stcl = Gstencil_op_st_id(0), &
Gstencil_op_st_stop = Gstencil_op_st_id(1)
```

8.9 Data types for audit trails

Audit operations

```
TYPE Gaudit_op_id
  INTEGER :: i
END TYPE Gaudit_op_id

TYPE(Gaudit_op_id), PARAMETER :: &
Gaudit_op_open  = Gaudit_op_id(0), &
Gaudit_op_close = Gaudit_op_id(1), &
Gaudit_op_begin = Gaudit_op_id(2), &
Gaudit_op_end   = Gaudit_op_id(3)
```

Process operations

```
TYPE Gprocess_op_id
  INTEGER :: i
END TYPE Gprocess_op_id
```

```
TYPE(Gprocess_op_id), PARAMETER :: &
Gprocess_op_skip = Gprocess_op_id(0), &
Gprocess_op_do   = Gprocess_op_id(1)
```

Playback operations

```
TYPE Gplayback_op_id
  INTEGER :: i
END TYPE Gplayback_op_id

TYPE(Gplayback_op_id), PARAMETER :: &
Gplayback_op_open  = Gplayback_op_id(0), &
Gplayback_op_close = Gplayback_op_id(1)
```

Record audit flag

```
TYPE Grec_audit_flag_id
  INTEGER :: i
END TYPE Grec_audit_flag_id

TYPE(Grec_audit_flag_id), PARAMETER :: &
Grec_audit_flag_off = Grec_audit_flag_id(0), &
Grec_audit_flag_on  = Grec_audit_flag_id(1)
```

8.10 Data types for selection criteria

Selection Criteria

```
TYPE Gsel_op_id
  INTEGER :: i
END TYPE Gsel_op_id

TYPE(Gsel_op_id), PARAMETER :: &
Gsel_crit_contains  = Gsel_op_id(0), &
Gsel_crit_isin      = Gsel_op_id(1), &
Gsel_crit_equals    = Gsel_op_id(2), &
Gsel_crit_selall    = Gsel_op_id(3), &
Gsel_crit_rejectall = Gsel_op_id(4), &
Gsel_crit_and       = Gsel_op_id(5), &
Gsel_crit_or        = Gsel_op_id(6), &
Gsel_crit_not       = Gsel_op_id(7)
```

8.11 Data types for paths, tilings, stencils

Abut specifier

```
TYPE Gabut_spec_id
  INTEGER :: i
END TYPE Gabut_spec_id
```

```
TYPE(Gabut_spec_id), PARAMETER :: &
Gabut_spec_right       = Gabut_spec_id(0), &
Gabut_spec_left        = Gabut_spec_id(1), &
Gabut_spec_top         = Gabut_spec_id(2), &
Gabut_spec_bottom      = Gabut_spec_id(3), &
Gabut_spec_centrevert  = Gabut_spec_id(4), &
Gabut_spec_centrehoriz = Gabut_spec_id(5), &
Gabut_spec_capline     = Gabut_spec_id(6), &
Gabut_spec_baseline    = Gabut_spec_id(7)
```

Boundary

```
TYPE Gbndry_id
  INTEGER :: i
END TYPE Gbndry_id

TYPE(Gbndry_id), PARAMETER :: &
Gbndry_closedpathseq = Gbndry_id(0), &
Gbndry_area          = Gbndry_id(1)
```

Cap

```
TYPE Gcap_id
  INTEGER :: i
END TYPE Gcap_id

TYPE(Gcap_id), PARAMETER :: &
Gcap_butted  = Gcap_id(0), &
Gcap_rounded = Gcap_id(1), &
Gcap_square  = Gcap_id(2)
```

Contour attribute name

```
TYPE Gcont_attr_name_id
  INTEGER :: i
END TYPE Gcont_attr_name_id

TYPE(Gcont_attr_name_id), PARAMETER :: &
Gcont_attr_name_style = Gcont_attr_name_id(0), &
Gcont_attr_name_width = Gcont_attr_name_id(1), &
Gcont_attr_name_cap   = Gcont_attr_name_id(2), &
Gcont_attr_name_join  = Gcont_attr_name_id(3)
```

Inside rule

```
TYPE Ginside_rule_id
  INTEGER :: i
END TYPE Ginside_rule_id

TYPE(Ginside_rule_id), PARAMETER :: &
Ginside_rule_evenodd = Ginside_rule_id(0), &
Ginside_rule_winding = Ginside_rule_id(1)
```

Instance type

```
TYPE Ginstance_id
  INTEGER :: i
END TYPE Ginstance_id

TYPE(Ginstance_id), PARAMETER :: &
Ginstance_put   = Ginstance_id(0), &
Ginstance_align = Ginstance_id(1), &
Ginstance_map   = Ginstance_id(2)
```

Join

```
TYPE Gjoin_id
  INTEGER :: i
END TYPE Gjoin_id

TYPE(Gjoin_id), PARAMETER :: &
Gjoin_round  = Gjoin_id(0), &
Gjoin_bevel  = Gjoin_id(1), &
Gjoin_mitred = Gjoin_id(2)
```

LC control points

```
TYPE Glc_ctrl_point_id
  INTEGER :: i
END TYPE Glc_ctrl_point_id

TYPE(Glc_ctrl_point_id), PARAMETER :: &
Glc_ctrl_point_hlccp = Glc_ctrl_point_id(0), &
Glc_ctrl_point_lccp  = Glc_ctrl_point_id(1)
```

Path

```
TYPE Gpath_id
  INTEGER :: i
END TYPE Gpath_id

TYPE(Gpath_id), PARAMETER :: &
Gpath_line     = Gpath_id(0), &
Gpath_nurb     = Gpath_id(1), &
Gpath_conicsec = Gpath_id(2)
```

Replication technique

```
TYPE Greplic_tech_id
  INTEGER :: i
END TYPE Greplic_tech_id
```

```
TYPE(Greplic_tech_id), PARAMETER :: &
Greplic_tech_dx  = Greplic_tech_id(0), &
Greplic_tech_dy  = Greplic_tech_id(1), &
Greplic_tech_dxy = Greplic_tech_id(2), &
Greplic_tech_dyx = Greplic_tech_id(3)
```

Stencil attribute name

```
TYPE Gstencil_attr_name_id
  INTEGER :: i
END TYPE Gstencil_attr_name_id

TYPE(Gstencil_attr_name_id), PARAMETER :: &
Gstencil_attr_name_topy         = Gstencil_attr_name_id(0), &
Gstencil_attr_name_capy         = Gstencil_attr_name_id(1), &
Gstencil_attr_name_halfy        = Gstencil_attr_name_id(2), &
Gstencil_attr_name_basey        = Gstencil_attr_name_id(3), &
Gstencil_attr_name_bottomy      = Gstencil_attr_name_id(4), &
Gstencil_attr_name_centrey      = Gstencil_attr_name_id(5), &
Gstencil_attr_name_leftx        = Gstencil_attr_name_id(6), &
Gstencil_attr_name_rightx       = Gstencil_attr_name_id(7), &
Gstencil_attr_name_centrex      = Gstencil_attr_name_id(8), &
Gstencil_attr_name_centre       = Gstencil_attr_name_id(9), &
Gstencil_attr_name_origin       = Gstencil_attr_name_id(10), &
Gstencil_attr_name_centretop    = Gstencil_attr_name_id(11), &
Gstencil_attr_name_centrebottom = Gstencil_attr_name_id(12), &
Gstencil_attr_name_centreleft   = Gstencil_attr_name_id(13), &
Gstencil_attr_name_centreright  = Gstencil_attr_name_id(14), &
Gstencil_attr_name_topleft      = Gstencil_attr_name_id(15), &
Gstencil_attr_name_topright     = Gstencil_attr_name_id(16), &
Gstencil_attr_name_bottomleft   = Gstencil_attr_name_id(17), &
Gstencil_attr_name_bottomright  = Gstencil_attr_name_id(18)
```

Style

```
TYPE Gstyle_id
  INTEGER :: i
END TYPE Gstyle_id

TYPE(Gstyle_id), PARAMETER :: &
Gstyle_solid      = Gstyle_id(0), &
Gstyle_dash       = Gstyle_id(1), &
Gstyle_dot        = Gstyle_id(2), &
Gstyle_dashdot    = Gstyle_id(3), &
Gstyle_dashdotdot = Gstyle_id(4)
```

Tiling parameter

```
TYPE Gtiling_param_id
  INTEGER :: i
END TYPE Gtiling_param_id
```

```
TYPE(Gtiling_param_id), PARAMETER :: &
Gtiling_param_setline       = Gtiling_param_id(0), &
Gtiling_param_nurb          = Gtiling_param_id(1), &
Gtiling_param_conicsec      = Gtiling_param_id(2), &
Gtiling_param_setfill       = Gtiling_param_id(3), &
Gtiling_param_setellipsec   = Gtiling_param_id(4), &
Gtiling_param_setellipseg   = Gtiling_param_id(5), &
Gtiling_param_setellipdisc  = Gtiling_param_id(6), &
Gtiling_param_setclosednurb = Gtiling_param_id(7), &
Gtiling_param_design        = Gtiling_param_id(8)
```

8.12 Data types for state lists and description tables

Operating state list

```
TYPE Gop_st_list_id
  INTEGER :: i
END TYPE Gop_st_list_id

TYPE(Gop_st_list_id), PARAMETER :: &
Gop_st_list_gks     = Gop_st_list_id(0), &
Gop_st_list_picpart = Gop_st_list_id(1), &
Gop_st_list_seg     = Gop_st_list_id(2), &
Gop_st_list_stencil = Gop_st_list_id(3), &
Gop_st_list_tiling  = Gop_st_list_id(4), &
Gop_st_list_all     = Gop_st_list_id(5)
```

GKS description table

```
TYPE Ggdt_id
  INTEGER :: i
END TYPE Ggdt_id

TYPE(Ggdt_id), PARAMETER :: &
Ggdt_availwsgentypes = Ggdt_id(0), &
Ggdt_availfonts      = Ggdt_id(1), &
Ggdt_deffontindmap   = Ggdt_id(2), &
Ggdt_all             = Ggdt_id(3)
```

GKS state list

```
TYPE Ggsl_id
  INTEGER :: i
END TYPE Ggsl_id

TYPE(Ggsl_id), PARAMETER :: &
Ggsl_routedir        = Ggsl_id(0), &
Ggsl_scissormode     = Ggsl_id(1), &
Ggsl_openwsids       = Ggsl_id(2), &
Ggsl_activewsids     = Ggsl_id(3), &
Ggsl_openaudits      = Ggsl_id(4), &
```

```
Ggsl_openplaybacks   = Ggsl_id(5), &
Ggsl_inputqueue      = Ggsl_id(6), &
Ggsl_fontindmap      = Ggsl_id(7), &
Ggsl_curprimattrs    = Ggsl_id(8), &
Ggsl_curnormtrannum  = Ggsl_id(9), &
Ggsl_normtrans       = Ggsl_id(10), &
Ggsl_vpinpris        = Ggsl_id(11), &
Ggsl_openpicpartname = Ggsl_id(12), &
Ggsl_picpartnames    = Ggsl_id(13), &
Ggsl_opensegname     = Ggsl_id(14), &
Ggsl_segnames        = Ggsl_id(15), &
Ggsl_seglists        = Ggsl_id(16), &
Ggsl_openstencilname = Ggsl_id(17), &
Ggsl_stencilnames    = Ggsl_id(18), &
Ggsl_stencilstlists  = Ggsl_id(19), &
Ggsl_curcontattrs    = Ggsl_id(20), &
Ggsl_opentilingname  = Ggsl_id(21), &
Ggsl_tilingnames     = Ggsl_id(22), &
Ggsl_all             = Ggsl_id(23)
```

Workstation state list

```
TYPE Gwsl_id
  INTEGER :: i
END TYPE Gwsl_id

TYPE(Gwsl_id), PARAMETER :: &
Gwsl_curvebuntab  = Gwsl_id(0), &
Gwsl_markerbuntab = Gwsl_id(1), &
Gwsl_areabuntab   = Gwsl_id(2), &
Gwsl_pattab       = Gwsl_id(3), &
Gwsl_charbuntab   = Gwsl_id(4), &
Gwsl_colrtab      = Gwsl_id(5), &
Gwsl_curwswin     = Gwsl_id(6), &
Gwsl_curwsvp      = Gwsl_id(7), &
Gwsl_curviseffst  = Gwsl_id(8), &
Gwsl_devopmodes   = Gwsl_id(9), &
Gwsl_devinitvals  = Gwsl_id(10), &
Gwsl_selcrits     = Gwsl_id(11), &
Gwsl_segnames     = Gwsl_id(12), &
Gwsl_viewreps     = Gwsl_id(13), &
Gwsl_viewpris     = Gwsl_id(14), &
Gwsl_all          = Gwsl_id(15)
```

Workstation description table

```
TYPE Gwdt_id
  INTEGER :: i
END TYPE Gwdt_id

TYPE(Gwdt_id), PARAMETER :: &
Gwdt_wstype              = Gwdt_id(0), &
Gwdt_wsst                = Gwdt_id(1), &
Gwdt_dcunits             = Gwdt_id(2), &
```

```
Gwdt_dispsize           = Gwdt_id(3), &
Gwdt_dispkind           = Gwdt_id(4), &
Gwdt_availcurvetypes    = Gwdt_id(5), &
Gwdt_numavailcurvewidths = Gwdt_id(6), &
Gwdt_nomcurvewidth      = Gwdt_id(7), &
Gwdt_mincurvewidth      = Gwdt_id(8), &
Gwdt_maxcurvewidth      = Gwdt_id(9), &
Gwdt_predcurvebuns      = Gwdt_id(10), &
Gwdt_availmarkertypes   = Gwdt_id(11), &
Gwdt_numavailmarkersizes = Gwdt_id(12), &
Gwdt_nommarkersize      = Gwdt_id(13), &
Gwdt_minmarkersize      = Gwdt_id(14), &
Gwdt_maxmarkersize      = Gwdt_id(15), &
Gwdt_predmarkerbuns     = Gwdt_id(16), &
Gwdt_availintstyles     = Gwdt_id(17), &
Gwdt_availhatchstyles   = Gwdt_id(18), &
Gwdt_predareabuns       = Gwdt_id(19), &
Gwdt_availedgetypes     = Gwdt_id(20), &
Gwdt_numavailedgewidths  = Gwdt_id(21), &
Gwdt_nomedgewidth       = Gwdt_id(22), &
Gwdt_minedgewidth       = Gwdt_id(23), &
Gwdt_maxedgewidth       = Gwdt_id(24), &
Gwdt_predpats           = Gwdt_id(25), &
Gwdt_availfontprecs     = Gwdt_id(26), &
Gwdt_numavailcharexpans  = Gwdt_id(27), &
Gwdt_mincharexpan       = Gwdt_id(28), &
Gwdt_maxcharexpan       = Gwdt_id(29), &
Gwdt_numavailtextheight  = Gwdt_id(30), &
Gwdt_mintextheight      = Gwdt_id(31), &
Gwdt_maxtextheight      = Gwdt_id(32), &
Gwdt_predcharbuns       = Gwdt_id(33), &
Gwdt_availcolrmodels    = Gwdt_id(34), &
Gwdt_colrchars          = Gwdt_id(35), &
Gwdt_numavailcolrs      = Gwdt_id(36), &
Gwdt_colravail          = Gwdt_id(37), &
Gwdt_predcolrreps       = Gwdt_id(38)

TYPE(Gwdt_id), PARAMETER :: &
Gwdt_availgdps          = Gwdt_id(39), &
Gwdt_availindevs     = Gwdt_id(40), &
Gwdt_availinitvals   = Gwdt_id(41), &
Gwdt_availmeasureids = Gwdt_id(42), &
Gwdt_availtriggerids = Gwdt_id(43), &
Gwdt_devcompose      = Gwdt_id(44), &
Gwdt_all             = Gwdt_id(45)
```

Error State List

```
TYPE Gesl_id
  INTEGER :: i
END TYPE Gesl_id

TYPE(Gesl_id), PARAMETER :: &
Gesl_err_st             = Gesl_id(0), &
Gesl_err_file           = Gesl_id(1), &
Gesl_dev_id_overflow    = Gesl_id(2)
```

Segment state list

```
TYPE Gssl_id
  INTEGER :: i
END TYPE Gssl_id

TYPE(Gssl_id), PARAMETER :: &
Gssl_segname      = Gssl_id(0), &
Gssl_assocwsids   = Gssl_id(1), &
Gssl_cursegattrs  = Gssl_id(2), &
Gssl_all          = Gssl_id(3)
```

Stencil state list

```
TYPE Gstsl_id
  INTEGER :: i
END TYPE Gstsl_id

TYPE(Gstsl_id), PARAMETER :: &
Gstsl_stencilname     = Gstsl_id(0), &
Gstsl_curstencilattrs = Gstsl_id(1), &
Gstsl_all             = Gstsl_id(2)
```

Name availability

```
TYPE Gname_id
  INTEGER :: i
END TYPE Gname_id

TYPE(Gname_id), PARAMETER :: &
Gname_available = Gname_id(0), &
Gname_undefined = Gname_id(1)
```

Identifier availability

```
TYPE Gid_id
  INTEGER :: i
END TYPE Gid_id

TYPE(Gid_id), PARAMETER :: &
Gid_available = Gid_id(0), &
Gid_undefined = Gid_id(1)
```

9 Lists of the GKS Function Names

This list is in order of appearance

GKS function name	Fortran 90 subroutine name
Open GKS	gopen_gks
Close GKS	gclose_gks
Save GKS state list	gsave_gsl
Restore GKS state list	grestore_gsl
Escape	gescape
Emergency close GKS	gemergency_close_gks
Error handling	gerr_hand
Error logging	gerr_log
Route	groute
Create output primitive	gcreate_out_prim
Open stencil	gopen_stencil
Close stencil	gclose_stencil
Rename stencil	grename_stencil
Delete stencil	gdel_stencil
Create stencil from boundary	gcreate_stencil_bndry
Create stencil from contours	gcreate_stencil_conts
Set contour attribute	gset_cont_attr
Set stencil attribute	gset_stencil_attr
Get stencil attribute	gget_stencil_attr
Instance stencil	ginst_stencil
Instance stencil along path	ginst_stencil_path
Instance stencil sequence along path	ginst_stencil_seq_path
Open tiling	gopen_tiling
Close tiling	gclose_tiling
Rename tiling	grename_tiling
Delete tiling	gdel_tiling
Create tiling component	gcreate_tiling_comp
Set primitive attribute	gset_prim_attr
Add set of names to nameset	gadd_set_names_nameset
Remove set of names from nameset	gremove_set_names_nameset
Add set of scissors to scissor set	gadd_set_scissor_scissor_set
Remove set of scissors from scissor set	gremove_set_scissor_scissor_set
Set window and viewport	gset_win_vp
Set viewport input priority	gset_vp_in_pri
Set scissor mode	gset_scissor_mode
Set normalization transformation number	gset_norm_tran_num
Delete primitives from NDC picture	gdel_prims_ndc_pic
Add set of names to NDC picture	gadd_set_names_ndc_pic
Remove set of names from NDC picture	gremove_set_names_ndc_pic
Set NDC picture primitive attribute	gset_ndc_pic_prim_attr
Remove set of scissors from NDC picture	gremove_set_scissor_ndc_pic
Add set of scissors to NDC picture	gadd_set_scissor_ndc_pic
Reorder NDC picture	greorder_ndc_pic
Copy NDC picture to NDC metafile	gcopy_ndc_pic_ndc_mf
Copy NDC metafile picture to NDC picture	gcopy_ndc_mf_pic_ndc_pic
Open picture part	gopen_pic_part
Close picture part	gclose_pic_part
Archive picture part	gar_pic_part
Retrieve picture part from archive	gret_pic_part_ar
Reopen picture part	greopen_pic_part
Append picture part	gappend_pic_part
Rename picture part	grename_pic_part
Delete picture part	gdel_pic_part

GKS function name	Fortran 90 subroutine name
Copy picture part from picture part store	gcopy_pic_part_pic_part_store
Copy NDC picture to picture part store	gcopy_ndc_pic_pic_part_store
Set logical input device mode	gset_in_dev_mode
Request input	greq_in
Sample input	gsample_in
Await input	gawait_in
Flush device events	gflush_dev_events
Set font index mapping	gset_font_ind_map
Get glyph name	gget_glyph_name
Set character code	gset_char_code
Audit	gaudit
Write user record to audit	gwrite_user_rec_audit
Playback	gplayback
Read item function name from audit	gread_item_func_name_audit
Read item from audit	gread_item_audit
Process audit item	gproc_audit_item
Inquire operating state entry	ginq_op_st_entry
Inquire GKS description table entry	ginq_gdt_entry
Inquire GKS state list entry	ginq_gsl_entry
Inquire input queue overflow	ginq_in_overf
Evaluate transformation matrix	geval_tran_matrix
Accumulate transformation matrix	gaccum_tran_matrix
Evaluate circle	geval_circle
Evaluate ellipse	geval_ell
Evaluate circular arc 3 point	geval_circ_arc_3_point
Evaluate circular arc centre	geval_circ_arc_ctr
Evaluate elliptic arc	geval_ell_arc
Evaluate hyperbolic arc	geval_hyp_arc
Evaluate parabolic arc	geval_par_arc
Evaluate wc ordinate	geval_wc_ord
Open workstation	gopen_ws
Close workstation	gclose_ws
Save workstation state list	gsave_wsl
Restore workstation state list	grestore_wsl
Get workstation status	gget_ws_status
Remove backdrop	gremove_backdrop
Set representation	gset_rep
Set view priority	gset_view_pri
Set view selection criterion	gset_view_sel_crit
Set view	gset_view
Set workstation visual effects	gset_ws_vis_effects
Set workstation window and viewport	gset_ws_win_vp
Define logical input device	gdef_in_dev
Initialize logical input device	ginit_in_dev
Set workstation selection criterion	gset_ws_sel_crit
Copy realized picture to realized metafile	gcopy_real_pic_real_mf
Copy blank realized picture to realized metafile	gcopy_blank_real_pic_real_mf
Copy realized metafile picture to backdrop	gcopy_real_mf_pic_backdrop
Message	gmessage
Inquire workstation state list entry	ginq_wsl_entry
Inquire generic workstation description table entry	ginq_gen_wdt_entry
Inquire specific workstation description table entry	ginq_spec_wdt_entry
Inquire specific workstation description table entry state	ginq_spec_wdt_entry_st
Reset specific workstation description table entry state	gres_spec_wdt_entry_st
Get text extent	gget_text_extent
Evaluate view orientation matrix	geval_view_ori_matrix
Evaluate view mapping matrix	geval_view_map_matrix
Convert colour	gconv_colr

GKS function name	Fortran 90 subroutine name
Create segment	gcreate_seg
Close segment	gclose_seg
Rename segment	grename_seg
Delete segment	gdel_seg
Delete segment from workstation	gdel_seg_ws
Associate segment with workstation	gassoc_seg_ws
Copy segment to workstation	gcopy_seg_ws
Insert segment	ginsert_seg
Set segment attribute	gset_seg_attr
Activate workstation	gactivate_ws
Deactivate workstation	gdeactivate_ws
Clear workstation	gclear_ws
Get mapped segment name	gget_map_seg_name
Get mapped workstation identifier	gget_map_ws_id

This is the list of routines in alphabetical order	
GKS function name	Fortran 90 subroutine name
Accumulate transformation matrix	gaccum_tran_matrix
Activate workstation	gactivate_ws
Add set of names to NDC picture	gadd_set_names_ndc_pic
Add set of names to nameset	gadd_set_names_nameset
Add set of scissors to NDC picture	gadd_set_scissor_ndc_pic
Add set of scissors to scissor set	gadd_set_scissor_scissor_set
Append picture part	gappend_pic_part
Archive picture part	gar_pic_part
Associate segment with workstation	gassoc_seg_ws
Await input	gawait_in
Clear workstation	gclear_ws
Close GKS	gclose_gks
Close picture part	gclose_pic_part
Close segment	gclose_seg
Close stencil	gclose_stencil
Close tiling	gclose_tiling
Close workstation	gclose_ws
Convert colour	gconv_colr
Copy NDC metafile picture to NDC picture	gcopy_ndc_mf_pic_ndc_pic
Copy NDC picture to NDC metafile	gcopy_ndc_pic_ndc_mf
Copy NDC picture to picture part store	gcopy_ndc_pic_pic_part_store
Copy blank realized picture to realized metafile	gcopy_blank_real_pic_real_mf
Copy picture part from picture part store	gcopy_pic_part_pic_part_store
Copy realized metafile picture to backdrop	gcopy_real_mf_pic_backdrop
Copy realized picture to realized metafile	gcopy_real_pic_real_mf
Copy segment to workstation	gcopy_seg_ws
Create output primitive	gcreate_out_prim
Create segment	gcreate_seg
Create stencil from boundary	gcreate_stencil_bndry
Create stencil from contours	gcreate_stencil_conts
Create tiling component	gcreate_tiling_comp
Deactivate workstation	gdeactivate_ws
Define logical input device	gdef_in_dev
Delete picture part	gdel_pic_part
Delete primitives from NDC picture	gdel_prims_ndc_pic
Delete segment	gdel_seg
Delete segment from workstation	gdel_seg_ws
Delete stencil	gdel_stencil
Delete tiling	gdel_tiling

GKS function name	Fortran 90 subroutine name
Evaluate wc ordinate	geval_wc_ord
Emergency close GKS	gemergency_close_gks
Error handling	gerr_hand
Error logging	gerr_log
Escape	gescape
Evaluate circle	geval_circle
Evaluate circular arc 3 point	geval_circ_arc_3_point
Evaluate circular arc centre	geval_circ_arc_ctr
Evaluate ellipse	geval_ell
Evaluate elliptic arc	geval_ell_arc
Evaluate hyperbolic arc	geval_hyp_arc
Evaluate parabolic arc	geval_par_arc
Evaluate transformation matrix	geval_tran_matrix
Evaluate view mapping matrix	geval_view_map_matrix
Evaluate view orientation matrix	geval_view_ori_matrix
Flush device events	gflush_dev_events
Get glyph name	gget_glyph_name
Get mapped segment name	gget_map_seg_name
Get mapped workstation identifier	gget_map_ws_id
Get stencil attribute	gget_stencil_attr
Get text extent	gget_text_extent
Get workstation status	gget_ws_status
Initialize logical input device	ginit_in_dev
Inquire GKS description table entry	ginq_gdt_entry
Inquire GKS state list entry	ginq_gsl_entry
Inquire generic workstation description table entry	ginq_gen_wdt_entry
Inquire input queue overflow	ginq_in_overf
Inquire operating state entry	ginq_op_st_entry
Inquire specific workstation description table entry	ginq_spec_wdt_entry
Inquire specific workstation description table entry state	ginq_spec_wdt_entry_st
Inquire workstation state list entry	ginq_wsl_entry
Insert segment	ginsert_seg
Instance stencil	ginst_stencil
Instance stencil along path	ginst_stencil_path
Instance stencil sequence along path	ginst_stencil_seq_path
Message	gmessage
Open GKS	gopen_gks
Open picture part	gopen_pic_part
Open stencil	gopen_stencil
Open tiling	gopen_tiling
Open workstation	gopen_ws
Playback	gplayback
Process audit item	gproc_audit_item
Read item from audit	gread_item_audit
Read item function name from audit	gread_item_func_name_audit
Remove backdrop	gremove_backdrop
Remove set of names from NDC picture	gremove_set_names_ndc_pic
Remove set of names from nameset	gremove_set_names_nameset
Remove set of scissors from NDC picture	gremove_set_scissor_ndc_pic
Remove set of scissors from scissor set	gremove_set_scissor_scissor_set
Rename picture part	grename_pic_part
Rename segment	grename_seg
Rename stencil	grename_stencil
Rename tiling	grename_tiling
Reopen picture part	greopen_pic_part
Reorder NDC picture	greorder_ndc_pic
Request input	greq_in

GKS function name	Fortran 90 subroutine name
Restore workstation state list	grestore_wsl
Reset specific workstation description table entry state	gres_spec_wdt_entry_st
Restore GKS state list	grestore_gsl
Retrieve picture part from archive	gret_pic_part_ar
Route	groute
Sample input	gsample_in
Save GKS state list	gsave_gsl
Save workstation state list	gsave_wsl
Set NDC picture primitive attribute	gset_ndc_pic_prim_attr
Set character code	gset_char_code
Set contour attribute	gset_cont_attr
Set font index mapping	gset_font_ind_map
Set logical input device mode	gset_in_dev_mode
Set normalization transformation number	gset_norm_tran_num
Set primitive attribute	gset_prim_attr
Set representation	gset_rep
Set scissor mode	gset_scissor_mode
Set segment attribute	gset_seg_attr
Set stencil attribute	gset_stencil_attr
Set view	gset_view
Set view priority	gset_view_pri
Set view selection criterion	gset_view_sel_crit
Set viewport input priority	gset_vp_in_pri
Set window and viewport	gset_win_vp
Set workstation selection criterion	gset_ws_sel_crit
Set workstation visual effects	gset_ws_vis_effects
Set workstation window and viewport	gset_ws_win_vp
Write user record to audit	gwrite_user_rec_audit
Write user record to audit	gwrite_user_rec_audit

10 GKS Errors

Some errors may occur which are not defined within the scope of the GKS standard ISO/IEC 7942-1:1994(E) .

Specifically, these new errors are defined below:

OPEN GKS

Can the error file be invalid

CLOSE GKS

SAVE GKS STATE LIST

Is there enough storage available

RESTORE GKS STATE LIST

ESCAPE

Can the escape function identification be invalid
Can the escape input data record be invalid

EMERGENCY CLOSE GKS

ERROR HANDLING

Is the error number valid
Is the function name valid
Is the error file valid

ERROR LOGGING

Is the error number valid
Is the function name valid
Is the error file valid

ROUTE

Is the direction valid

CREATE OUTPUT PRIMITIVE

Is the primitive type valid
Should language binding give errors on:

Number of points invalid
Invalid code in string
Dimensions of colour array invalid
GDP identifier is invalid
GDP data record is invalid
NURB definition invalid
Conic section is invalid

OPEN STENCIL

Is the stencil name valid
Is the inside rule valid

CLOSE STENCIL

Is a stencil open

RENAME STENCIL

Are the stencil names valid

DELETE STENCIL

Is the stencil name valid

CREATE STENCIL FROM BOUNDARY

Is the boundary specification valid
Is the stencil name valid
Is the inside rule valid

CREATE STENCIL FROM CONTOURS

Is the sequence of paths valid
Is the stencil name valid
Is the inside rule valid

SET CONTOUR ATTRIBUTE

Is the stencil attribute name valid
Is the stencil attribute value valid

SET STENCIL ATTRIBUTE

Is the stencil name valid
Is the stencil attribute name valid
Is the stencil attribute value valid

GET STENCIL ATTRIBUTE

Is the stencil name valid
Is the stencil attribute name valid

INSTANCE STENCIL

Is the stencil name valid
Is the instance name valid
Is the instance specifier valid

INSTANCE STENCIL ALONG PATH

Is the stencil name valid
Is the instance name valid
Is the instance number valid
Is the path valid

INSTANCE STENCIL SEQUENCE ALONG PATH

Is the instance sequence specifier valid
Is the path specifier valid

OPEN TILING

Is the tiling name valid

CLOSE TILING

Is a tiling open

RENAME TILING

Are the tiling names valid

DELETE TILING

Is the tiling name valid

CREATE TILING COMPONENT

Is the primitive type valid
Is the replication technique valid

Should language binding give errors on:

Number of points invalid
Invalid code in string
NURB definition invalid
Conic section is invalid

SET PRIMITIVE ATTRIBUTE

Is the attribute name valid
Is the attribute value valid

ADD SET OF NAMES TO NAMESET

Is the set of names valid

REMOVE SET OF NAMES FROM NAMESET

Is the set of names valid

SET WINDOW AND VIEWPORT

Is the transformation number less than 1
Is the window rectangle definition valid
Is the viewport rectangle definition valid

SET VIEWPORT INPUT PRIORITY

Is the transformation number negative

SET SCISSOR MODE

Is the scissor mode valid

SET NORMALIZATION TRANSFORMATION NUMBER

Is the transformation number negative

DELETE PRIMITIVES FROM NDC PICTURE

Is the selection criterion valid

ADD SET OF NAMES TO NDC PICTURE

Is the set of names valid
Is the selection criterion valid

REMOVE SET OF NAMES FROM NDC PICTURE

Is the set of names valid
Is the selection criterion valid

SET NDC PICTURE PRIMITIVE ATTRIBUTE

Is the attribute name valid
Is the attribute value valid
Is the selection criterion valid

ADD SET OF SCISSORS TO NDC PICTURE

Are the scissor specifications valid

REMOVE SET OF SCISSORS TO NDC PICTURE

Are the scissor specifications valid

REORDER NDC PICTURE

Are the selection criteria valid

COPY NDC PICTURE TO NDC METAFILE

Is the metafile specifier valid
Is the picture identifier valid
Is the selection criterion valid
Is the set of names valid

COPY NDC METAFILE PICTURE TO NDC PICTURE

Is the metafile specifier valid
Is the picture identifier valid
Is the name valid

OPEN PICTURE PART

Is the picture part name valid

CLOSE PICTURE PART

ARCHIVE PICTURE PART

Is the picture part name valid
Is the archive specifier valid
Is the archive name of picture part valid

RETRIEVE PICTURE PART FROM ARCHIVE

Is the picture part name valid
Is the archive specifier valid
Is the archive name of picture part valid

REOPEN PICTURE PART

Is the picture part name valid

APPEND PICTURE PART

Is the source picture part name valid
Is the sink picture part name valid
Is the global transformation mode valid
Is the local transformation mode valid
Is the set of names valid

RENAME PICTURE PART

Is the old picture part name valid
Is the new picture part name valid

DELETE PICTURE PART

Is the picture part name valid

COPY PICTURE PART FROM PICTURE PART STORE

Is the picture part name valid
Is the selection criterion valid
Is the global transformation mode valid
Is the local transformation mode valid
Is the set of names valid
Is the scissor select valid

COPY NDC PICTURE TO PICTURE PART STORE

Is the selection criterion valid
Is the picture part name valid
Is the set of names valid

SET LOGICAL INPUT DEVICE MODE

Is the logical input device identifier valid
Is the operating mode valid

REQUEST INPUT

Is the logical input device identifier valid

SAMPLE INPUT

Is the logical input device identifier valid

AWAIT INPUT

Is the logical input device identifier valid
Is the timeout positive

FLUSH DEVICE EVENTS

Is the logical input device identifier valid

SET FONT INDEX MAPPING

Is the font index zero
Is the font name valid

GET GLYPH NAME

Is the font index zero
Is the character code valid

SET CHARACTER CODE

Is the font index zero
Is the character code valid

AUDIT

Is the audit identifier valid
Is the audit specifier valid
Is the audit operation valid

WRITE USER RECORD TO AUDIT

Is the audit identifier valid
Is the audit user data record valid

PLAYBACK

Is the audit identifier valid
Is the audit specifier valid
Is the playback operation valid

READ ITEM FUNCTION NAME

Is the audit identifier valid

READ ITEM

Is the audit identifier valid

INTERPRET ITEM

Is the audit identifier valid
Is the interpret operation valid

INQUIRE OPERATING STATE VALUE

INQUIRE GKS DESCRIPTION TABLE

Is there enough storage space available

INQUIRE GKS STATE LIST

Is there enough storage space available

INQUIRE INPUT QUEUE OVERFLOW

EVALUATE TRANSFORMATION MATRIX

Is the coordinate switch valid

ACCUMULATE TRANSFORMATION MATRIX

Is the coordinate switch valid

EVALUATE CIRCLE

Is the radius greater than zero

EVALUATE CIRCULAR ARC 3 POINT

EVALUATE CIRCULAR ARC CENTRE

Is the senseflag valid
Is the radius greater than zero

EVALUATE ELLIPTIC ARC

EVALUATE HYPERBOLIC ARC

Is the radius greater than zero

EVALUATE PARABOLIC ARC

EVALUATE WC ORDINATE

Is the ordinate switch valid

OPEN WORKSTATION

Is the workstation identifier valid
Is the workstation type valid
Is the workstation specifier valid

CLOSE WORKSTATION

Is the workstation identifier valid

SAVE WORKSTATION STATE LIST

Is the workstation identifier valid
Are the entry names valid
Is there enough storage space available

RESTORE WORKSTATION STATE LIST

Is the workstation identifier valid
Are the entry values valid

REMOVE BACKDROP

Is the workstation identifier valid

SET COLOUR MODEL

Is the workstation identifier valid
Is the colour model valid

SET REPRESENTATION

Is the workstation identifier valid
Is the representation designation valid
Is the index valid
Is the Linetype equal to 0
Is the Linewidth scale factor less than 0
Is the NURBtype equal 0
Is the NURBwidth scale factor less than 0
Is the Marker type equal to 0
Is the Marker size scale factor less than 0
Is the Text font equal to 0
Is the Character expansion factor less than or equal to 0
Is the Edgetype equal to 0
Is the Colour index valid
Is the Pattern index valid
Are the dimensions of colour array valid
Is the Colour specification valid

SET VIEW INPUT PRIORITY

Is the workstation identifier valid
Is the view index valid
Is the reference index valid
Is the priority valid

SET VIEW SELECTION CRITERION

Is the workstation identifier valid
Is the view index valid
Is the selection criterion valid

SET VIEW

Is the workstation identifier valid
Is the view index valid
Is the scissor indicator valid
Are the scissor rectangles valid

SET NDC VISUAL EFFECTS

Is the workstation identifier valid
Is the visual effects state valid

SET WORKSTATION WINDOW AND VIEWPORT

Is the workstation identifier valid
Are the window limits valid
Are the viewport limits valid

DEFINE LOGICAL INPUT DEVICE

Is the device identifier valid
Is the measure sequence valid
Is the trigger set valid

INITIALIZE LOGICAL INPUT DEVICE

Is the device identifier valid
Is the timeout valid
Is the echo switch valid
Is the echo area rectangle definition valid

SET WORKSTATION SELECTION CRITERION

Is the workstation identifier valid
Is the workstation type valid
Is the selection criterion valid

COPY REIFIED PICTURE TO REIFIED METAFILE

Is the workstation identifier valid
Is the metafile specifier valid
Is the picture identifier valid

COPY BLANK REIFIED PICTURE TO REIFIED METAFILE

Is the workstation identifier valid
Is the metafile specifier valid
Is the picture identifier valid

COPY REIFIED METAFILE PICTURE TO BACKDROP

Is the workstation identifier valid
Is the metafile specifier valid
Is the picture identifier valid

MESSAGE

Is the workstation identifier valid
Are all the codes in the string valid

INQUIRE WORKSTATION STATE LIST

Is the workstation identifier valid
Is the type of returned values valid

INQUIRE GENERIC WORKSTATION DESCRIPTION TABLE

Is the workstation type valid

INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE ENTRY

Is the workstation identifier valid
Is the entry name valid

INQUIRE SPECIFIC WORKSTATION DESCRIPTION TABLE STATE

Is the workstation identifier valid
Is the entry name valid

RESET SPECIFIC WORKSTATION DESCRIPTION TABLE STATE

Is the workstation identifier valid
Is the entry name valid

GET TEXT EXTENT

Is the workstation identifier valid
Are all the codes in the string valid

EVALUATE VIEW ORIENTATION MATRIX

EVALUATE VIEW MAPPING MATRIX

Are the window limits valid
Are the viewport limits valid

CREATE SEGMENT

Is the segment name valid

CLOSE SEGMENT

RENAME SEGMENT

Is the old segment name valid
Is the new segment name valid

DELETE SEGMENT

Is the segment name valid

DELETE SEGMENT FROM WORKSTATION

Is the segment name valid
Is the workstation identifier valid

ASSOCIATE SEGMENT WITH WORKSTATION

Is the segment name valid
Is the workstation identifier valid

COPY SEGMENT TO WORKSTATION

Is the segment name valid
Is the workstation identifier valid

INSERT SEGMENT

Is the segment name valid

SET SEGMENT ATTRIBUTE

Is the segment name valid
Is the attribute name valid
Is the attribute value valid

ACTIVATE WORKSTATION

Is the workstation identifier valid

DEACTIVATE WORKSTATION

Is the workstation identifier valid

CLEAR WORKSTATION

Is the workstation identifier valid

GET MAPPED SEGMENT NAME

Is the segment name valid

GET MAPPED WORKSTATION IDENTIFIER

Is the mapped workstation identifier valid

11 The GKS Function Interface

11.1 General Principles

For each GKS function the corresponding Fortran 90 subroutine declaration is given. The name of the GKS function is listed, followed by its Fortran 90 name and the corresponding parameters. After that, the list of parameters is described by type and a brief identifying phrase.

The mapping of enumeration types are listed in clause 7 with the GKS abstract data types and their corresponding Fortran 90 data type being shown in clause 6.

As each abstract function in ISO/IEC 7942-1:1994(E) has been bound to a Fortran 90 subroutine then some subroutines are used to define a generic interface to cater for abstract union data types which are arguments to GKS functions.

11.2 Control Functions

Open GKS

```
SUBROUTINE gopen_gks (err_file)

  CHARACTER(LEN=*), INTENT(IN)  :: err_file  ! error message file
```

Close GKS

```
SUBROUTINE gclose_gks ()
```

Save GKS state list

```
SUBROUTINE gsave_gsl (wsid, ent_names, ent_vals)
  INTEGER, INTENT(IN)                          :: wsid
  TYPE(Ggsl_id), POINTER, DIMENSION (:)        :: ent_names
  TYPE(Ggsl), POINTER, DIMENSION (:)           :: ent_vals
```

Restore GKS state list

```
SUBROUTINE grestore_gsl (wsid, ent_vals)
  INTEGER, INTENT(IN)                :: wsid
  TYPE(Ggsl), POINTER, DIMENSION (:) :: ent_vals
```

Escape

```
SUBROUTINE gescape (escape_id, in_data, out_data)
  INTEGER, INTENT(IN)                :: escape_id
  CHARACTER(LEN=*),INTENT(IN)        :: in_data
  CHARACTER(LEN=*),INTENT(OUT)       :: out_data
```

Emergency close GKS

```
SUBROUTINE gemergency_close_gks ()
```

Error handling

```
SUBROUTINE gerr_hand (err_num, fname, err_file)
  INTEGER, INTENT(IN)          :: err_num
  INTEGER, INTENT(IN)          :: fname
  CHARACTER(LEN=*), INTENT(IN) :: err_file
```

Error logging

```
SUBROUTINE gerr_log (err_num, fname, err_file)
  INTEGER, INTENT(IN)          :: err_num
  INTEGER, INTENT(IN)          :: fname
  CHARACTER(LEN=*), INTENT(IN) :: err_file
```

Route

```
SUBROUTINE groute (dir)
  TYPE(Groute_dir_id), INTENT(IN) :: dir
```

11.3 Output Functions

Create output primitive

```
INTERFACE gcreate_out_prim

 SUBROUTINE gcreate_out_line (line)
  TYPE(Gset_line), INTENT(IN)       :: line


 SUBROUTINE gcreate_out_nurb (nurb)
  TYPE(Gset_nurb), INTENT(IN)       :: nurb


 SUBROUTINE gcreate_out_conic_sec (conic_sec)
  TYPE(Gset_conic_sec), INTENT(IN)   :: conic_sec


 SUBROUTINE gcreate_out_marker (marker)
  TYPE(Gmarker), INTENT(IN)         :: marker
```

```
 SUBROUTINE gcreate_out_fill (fill)
  TYPE(Gset_fill), INTENT(IN)        :: fill


 SUBROUTINE gcreate_out_ellip_sec (ellip_sec)
  TYPE(Gset_ellip_sec), INTENT(IN)   :: ellip_sec


 SUBROUTINE gcreate_out_ellip_seg (ellip_seg)
  TYPE(Gset_ellip_seg), INTENT(IN)   :: ellip_seg


 SUBROUTINE gcreate_out_ellip_disc (ellip_disc)
  TYPE(Gset_ellip_disc), INTENT(IN)  :: ellip_disc


 SUBROUTINE gcreate_out_closed_nurb (closed_nurb)
  TYPE(Gset_closed_nurb), INTENT(IN) :: closed_nurb


 SUBROUTINE gcreate_out_text (text)
  TYPE(Gtext), INTENT(IN)            :: text


 SUBROUTINE gcreate_out_cell_array (cell_array)
  TYPE(Gcell_array), INTENT(IN)      :: cell_array


 SUBROUTINE gcreate_out_design (design)
  TYPE(Gdesign), INTENT(IN)          :: design


 SUBROUTINE gcreate_out_gdp (gdp)
  TYPE(Ggdp), INTENT(IN)             :: gdp


END INTERFACE
```

Open stencil

```
SUBROUTINE gopen_stencil ( stencil_name, inside_rule )
  INTEGER, INTENT(IN)               :: stencil_name
  TYPE(Ginside_rule_id), INTENT(IN) :: inside_rule
```

Close stencil

```
SUBROUTINE gclose_stencil ()
```

Rename stencil

```
SUBROUTINE grename_stencil ( old_stencil_name, new_stencil_name )
  INTEGER, INTENT(IN) :: old_stencil_name
  INTEGER, INTENT(IN) :: new_stencil_name
```

Delete stencil

```
SUBROUTINE gdel_stencil ( stencil_name )
   INTEGER, INTENT(IN) :: stencil_name
```

Create stencil from boundary

```
SUBROUTINE gcreate_stencil_bndry ( stencil_name, inside_rule, bndry )
  INTEGER               :: stencil_name
  TYPE(Ginside_rule_id) :: inside_rule
  TYPE(Gseq_bndry)      :: bndry
```

Create stencil from contours

```
SUBROUTINE gcreate_stencil_conts (stencil_name, inside_rule, path )
  INTEGER, INTENT(IN)                  :: stencil_name
  TYPE(Ginside_rule_id), INTENT(IN)    :: inside_rule
  TYPE(Gseq_path), INTENT(IN)          :: path
```

Set contour attribute

```
INTERFACE gset_cont_attr

 SUBROUTINE gset_cont_attr_style (style)
  TYPE(Gstyle_id), INTENT(IN) :: style


 SUBROUTINE gset_cont_attr_width (width)
  REAL, INTENT(IN)            :: width


 SUBROUTINE gset_cont_attr_cap (cap)
  TYPE(Gcap_id), INTENT(IN)   :: cap


 SUBROUTINE gset_cont_attr_join (join)
  TYPE(Gjoin), INTENT(IN)     :: join


END INTERFACE
```

Set stencil attribute

```
INTERFACE gset_stencil_attr

 SUBROUTINE gset_st_attr_topy (topy)
  TYPE(Gtopy)    :: topy


 SUBROUTINE gset_st_attr_capy (capy)
  TYPE(Gcapy)    :: capy


 SUBROUTINE gset_st_attr_halfy (halfy)
  TYPE(Ghalfy)   :: halfy


 SUBROUTINE gset_st_attr_basey (basey)
  TYPE(Gbasey)   :: basey


 SUBROUTINE gset_st_attr_bottomy (bottomy)
  TYPE(Gbottomy) :: bottomy


 SUBROUTINE gset_st_attr_centrey (centrey)
  TYPE(Gcentrey) :: centrey


 SUBROUTINE gset_st_attr_leftx (leftx)
  TYPE(Gleftx)   :: leftx


 SUBROUTINE gset_st_attr_rightx (rightx)
  TYPE(Grightx)  :: rightx


 SUBROUTINE gset_st_attr_centrex (centrex)
  TYPE(Gcentrex) :: centrex


 SUBROUTINE gset_st_attr_centre (centre)
  TYPE(Glc_point_centre)        :: centre


 SUBROUTINE gset_st_attr_origin (origin)
  TYPE(Glc_point_origin)        :: origin


 SUBROUTINE gset_st_attr_centretop (centretop)
  TYPE(Glc_point_centretop)     :: centretop


 SUBROUTINE gset_st_attr_centrebottom (centrebottom)
  TYPE(Glc_point_centrebottom)  :: centrebottom


 SUBROUTINE gset_st_attr_centreleft (centreleft)
  TYPE(Glc_point_centreleft)    :: centreleft
```

```
 SUBROUTINE gset_st_attr_centreright (centreright)
  TYPE(Glc_point_centreright)   :: centreright


 SUBROUTINE gset_st_attr_topleft (topleft)
  TYPE(Glc_point_topleft)       :: topleft


 SUBROUTINE gset_st_attr_topright (topright)
  TYPE(Glc_point_topright)      :: topright


 SUBROUTINE gset_st_attr_bottomleft (bottomleft)
  TYPE(Glc_point_bottomleft)    :: bottomleft


 SUBROUTINE gset_st_attr_bottomright (bottomright)
  TYPE(Glc_point_bottomright)   :: bottomright


END INTERFACE
```

Get stencil attribute

```
SUBROUTINE gget_stencil_attr (stencil_name, attr_name, error_ind, attr_val)
  INTEGER, INTENT(IN)                           :: stencil_name
  TYPE(Gstencil_attr_name_id), INTENT(IN)       :: attr_name
  INTEGER,INTENT(OUT)                           :: error_ind
  TYPE(Gstencil_attr_val),INTENT(OUT)           :: attr_val
```

Instance stencil

```
INTERFACE ginst_stencil

 SUBROUTINE ginst_stencil_put_spec (stencil_name, instance_name, stencil_tran, &
                                    put_spec)
  INTEGER, INTENT(IN)               :: stencil_name
  INTEGER, INTENT(IN)               :: instance_name
  TYPE(Gmatrix_23), INTENT(IN)      :: stencil_tran
  TYPE(Gput), INTENT(IN)            :: put_spec


 SUBROUTINE ginst_stencil_align_spec (stencil_name, instance_name, stencil_tran, &
                                      align_spec)
  INTEGER, INTENT(IN)               :: stencil_name
  INTEGER, INTENT(IN)               :: instance_name
  TYPE(Gmatrix_23), INTENT(IN)      :: stencil_tran
  TYPE(Galign), INTENT(IN)          :: align_spec
```

```
 SUBROUTINE ginst_stencil_map_spec (stencil_name, instance_name, stencil_tran, &
                                   map_spec)
  INTEGER, INTENT(IN)               :: stencil_name
  INTEGER, INTENT(IN)               :: instance_name
  TYPE(Gmatrix_23), INTENT(IN)      :: stencil_tran
  TYPE(Gmap), INTENT(IN)            :: map_spec


END INTERFACE
```

Instance stencil along path

```
SUBROUTINE ginst_stencil_path (stencil_name, instance_name, start_map, &
                               end_map, instance_num, path )
  INTEGER, INTENT(IN)          :: stencil_name
  INTEGER, INTENT(IN)          :: instance_name
  TYPE(Gwc_point), INTENT(IN)  :: start_map
  TYPE(Gwc_point), INTENT(IN)  :: end_map
  REAL, INTENT(IN)             :: instance_num
  TYPE(Gpath), INTENT(IN)      :: path
```

Instance stencil sequence along path

```
SUBROUTINE ginst_stencil_seq_path (instance_seq_spec, path, stencil_tran)
  TYPE(Ginstance_seq_spec), INTENT(IN) :: instance_seq_spec
  TYPE(Gpath_spec), INTENT(IN)         :: path
  TYPE(Gmatrix_23), INTENT(IN)         :: stencil_tran
```

Open tiling

```
SUBROUTINE gopen_tiling ( tiling_name )
  INTEGER, INTENT(IN) :: tiling_name
```

Close tiling

```
SUBROUTINE gclose_tiling ()
```

Rename tiling

```
SUBROUTINE grename_tiling ( old_tiling_name, new_tiling_name  )
  INTEGER, INTENT(IN) :: old_tiling_name
  INTEGER, INTENT(IN) :: new_tiling_name
```

Delete tiling

```
SUBROUTINE gdel_tiling ( tiling_name )
  INTEGER tiling_name
```

Create tiling component

```
INTERFACE gcreate_tiling_comp

 SUBROUTINE gctc_set_line (set_line, origin, rep_tech)
  TYPE(Gset_line), INTENT(IN)                :: set_line
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_nurb (set_nurb, origin, rep_tech)
  TYPE(Gset_nurb), INTENT(IN)                :: set_nurb
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_conic_sec (set_conic_sec, origin, rep_tech)
  TYPE(Gset_conic_sec), INTENT(IN)           :: set_conic_sec
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_fill (set_fill, origin, rep_tech)
  TYPE(Gset_fill), INTENT(IN)                :: set_fill
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_ellip_sec (set_ellip_sec, origin, rep_tech)
  TYPE(Gset_ellip_sec), INTENT(IN)           :: set_ellip_sec
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_ellip_seg (set_ellip_seg, origin, rep_tech)
  TYPE(Gset_ellip_seg), INTENT(IN)           :: set_ellip_seg
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_ellip_disc (set_ellip_disc, origin, rep_tech)
  TYPE(Gset_ellip_disc), INTENT(IN)          :: set_ellip_disc
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_set_closed_nurb (set_closed_nurb, origin, rep_tech)
  TYPE(Gset_closed_nurb), INTENT(IN)         :: set_closed_nurb
  TYPE(Gwc_point), INTENT(IN)                :: origin
  TYPE(Greplic_tech_id), INTENT(IN)          :: rep_tech

 SUBROUTINE gctc_design (design, origin, rep_tech)
  TYPE(Gdesign), INTENT(IN)                  :: design

END INTERFACE
```

11.4 Output attributes

Set primitive attribute

```
INTERFACE gset_prim_attr

 SUBROUTINE gset_pa_pick_id (pick_id)
  TYPE(Gpick_id), INTENT(IN)                     :: pick_id


 SUBROUTINE gset_pa_name_set (name_set)
  TYPE(Gname_set), INTENT(IN)                    :: name_set


 SUBROUTINE gset_pa_scissor_set (scissor_set)
  TYPE(Gscissor_set), INTENT(IN)                 :: scissor_set


 SUBROUTINE gset_pa_global_tran (global_tran)
  TYPE(Gmatrix_23_global), INTENT(IN)            :: global_tran


 SUBROUTINE gset_pa_loc_tran (loc_tran)
  TYPE(Gmatrix_23_local), INTENT(IN)             :: loc_tran


 SUBROUTINE gset_pa_pat_size (pat_size)
  TYPE(Gpat_size), INTENT(IN)                    :: pat_size


 SUBROUTINE gset_pa_pat_ref_pt (pat_ref_pt)
  TYPE(Gwc_point), INTENT(IN)                    :: pat_ref_pt


 SUBROUTINE gset_pa_text_height (text_height)
  TYPE(Gtext_height), INTENT(IN)                 :: text_height


 SUBROUTINE gset_pa_text_up_vec (text_up_vec)
  TYPE(Gtext_up_vec), INTENT(IN)                 :: text_up_vec


 SUBROUTINE gset_pa_text_skew (text_skew)
  TYPE(Gtext_skew), INTENT(IN)                   :: text_skew


 SUBROUTINE gset_pa_text_path (text_path)
  TYPE(Gtext_path_id), INTENT(IN)                :: text_path


 SUBROUTINE gset_pa_text_align (text_align)
  TYPE(Gtext_align), INTENT(IN)                  :: text_align
```

```
SUBROUTINE gset_pa_curve_ind (curve_ind)
 TYPE(Gcurve_ind), INTENT(IN)                  :: curve_ind


SUBROUTINE gset_pa_curve_asfs (curve_asfs)
 TYPE(Gcurve_asfs), INTENT(IN)                 :: curve_asfs


SUBROUTINE gset_pa_curve_type (curve_type)
 TYPE(Gcurve_type), INTENT(IN)                 :: curve_type


SUBROUTINE gset_pa_curve_width (curve_width)
 TYPE(Gcurve_width), INTENT(IN)                :: curve_width


SUBROUTINE gset_pa_curve_colr_spec (curve_colr_spec)
 TYPE(Gcurve_colr_spec), INTENT(IN)            :: curve_colr_spec


SUBROUTINE gset_pa_marker_ind (marker_ind)
 TYPE(Gmarker_ind), INTENT(IN)                 :: marker_ind


SUBROUTINE gset_pa_marker_asfs (marker_asfs)
 TYPE(Gmarker_asfs), INTENT(IN)                :: marker_asfs


SUBROUTINE gset_pa_marker_type (marker_type)
 TYPE(Gmarker_type), INTENT(IN)                :: marker_type


SUBROUTINE gset_pa_marker_size (marker_size)
 TYPE(Gmarker_size), INTENT(IN)                :: marker_size


SUBROUTINE gset_pa_marker_colr_spec (marker_colr_spec)
 TYPE(Gcolr_spec), INTENT(IN)                  :: marker_colr_spec


SUBROUTINE gset_pa_area_ind (area_ind)
 TYPE(Garea_ind), INTENT(IN)                   :: area_ind


SUBROUTINE gset_pa_area_asfs (area_asfs)
 TYPE(Garea_asfs), INTENT(IN)                  :: area_asfs


SUBROUTINE gset_pa_int_style (int_style)
 TYPE(Gint_style_id), INTENT(IN)               :: int_style


SUBROUTINE gset_pa_int_style_ind (int_style_ind)
 TYPE(Gint_style_ind), INTENT(IN)              :: int_style_ind
```

```
 SUBROUTINE gset_pa_int_colr_spec (int_colr_spec)
  TYPE(Gint_colr_spec), INTENT(IN)              :: int_colr_spec


 SUBROUTINE gset_pa_edge_flag (edge_flag)
  TYPE(Gedge_flag_id), INTENT(IN)               :: edge_flag


 SUBROUTINE gset_pa_edge_type (edge_type)
  TYPE(Gedge_type), INTENT(IN)                  :: edge_type


 SUBROUTINE gset_pa_edge_width (edge_width)
  TYPE(Gedge_width), INTENT(IN)                 :: edge_width


 SUBROUTINE gset_pa_edge_colr_spec (edge_colr_spec)
  TYPE(Gedge_colr_spec), INTENT(IN)             :: edge_colr_spec


 SUBROUTINE gset_pa_char_ind (char_ind)
  TYPE(Gchar_ind), INTENT(IN)                   :: char_ind


 SUBROUTINE gset_pa_char_asfs (char_asfs)
  TYPE(Gchar_asfs), INTENT(IN)                  :: char_asfs


 SUBROUTINE gset_pa_char_font_prec (char_font_prec)
  TYPE(Gfont_prec), INTENT(IN)                  :: char_font_prec


 SUBROUTINE gset_pa_char_expan (char_expan)
  TYPE(Gchar_expan), INTENT(IN)                 :: char_expan


 SUBROUTINE gset_pa_char_space (char_space)
  TYPE(Gchar_space), INTENT(IN)                 :: char_space


 SUBROUTINE gset_pa_char_colr_spec (char_colr_spec)
  TYPE(Gchar_colr_spec), INTENT(IN)             :: char_colr_spec


END INTERFACE
```

Add set of names to nameset

```
SUBROUTINE gadd_set_names_nameset (nameset)
  TYPE (Gname_set), INTENT(IN) :: nameset
```

Remove set of names from nameset

```
SUBROUTINE gremove_set_names_nameset (nameset)
  TYPE (Gname_set), INTENT(IN) :: nameset
```

Add set of scissors to scissor set

```
SUBROUTINE gadd_set_scissor_scissor_set (scissor_set)
  TYPE (Gscissor_set), INTENT(IN) :: scissor_set
```

Remove set of scissors from scissor set

```
SUBROUTINE gremove_set_scissor_scissor_set (scissor_id_set)
  TYPE (Gscissor_id_set), INTENT(IN) :: scissor_id_set
```

11.5 Normalization Transformation Functions

Set window and viewport

```
SUBROUTINE gset_win_vp (normtran, win_limits, vp_limits)
  INTEGER, INTENT(IN)        :: normtran
  TYPE (Gwc_win), INTENT(IN) :: win_limits
  TYPE (Gndc_vp), INTENT(IN) :: vp_limits
```

Set viewport input priority

```
SUBROUTINE gset_vp_in_pri (normtran, ref_tran_num, rel_pri)
  INTEGER, INTENT(IN)       :: normtran
  INTEGER, INTENT(IN)       :: ref_tran_num
  TYPE(Gpri_id), INTENT(IN) :: rel_pri
```

Set scissor mode

```
SUBROUTINE gset_scissor_mode (mode)
  TYPE(Gscissor_mode_id), INTENT(IN) :: mode
```

Set normalization transformation number

```
SUBROUTINE gset_norm_tran_num (normtran)
  INTEGER, INTENT(IN) :: normtran
```

11.6 NDC Picture Functions

Delete primitives from NDC picture

```
SUBROUTINE gdel_prims_ndc_pic (selcrit)
  TYPE(Gsel_crit), INTENT(IN) :: selcrit
```

Add set of names to NDC picture

```
SUBROUTINE gadd_set_names_ndc_pic (name_set, selcrit)
  TYPE(Gname_set), INTENT(IN)  :: name_set
  TYPE (Gsel_crit), INTENT(IN) :: selcrit
```

Remove set of names from NDC picture

```
SUBROUTINE gremove_set_names_ndc_pic (name_set, selcrit)
  TYPE(Gname_set), INTENT(IN) :: name_set
  TYPE(Gsel_crit), INTENT(IN) :: selcrit
```

Set NDC picture primitive attribute

```
INTERFACE gset_ndc_pic_prim_attr

 SUBROUTINE gset_nppa_pick_id (selcrit, pick_id)
  TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
  TYPE(Gpick_id), INTENT(IN)                    :: pick_id

 SUBROUTINE gset_nppa_name_set (selcrit, name_set)
  TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
  TYPE(Gname_set), INTENT(IN)                   :: name_set

 SUBROUTINE gset_nppa_scissor_set (selcrit, scissor_set)
  TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
  TYPE(Gscissor_set), INTENT(IN)                :: scissor_set

 SUBROUTINE gset_nppa_global_tran (selcrit, global_tran)
  TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
  TYPE(Gmatrix_23_global), INTENT(IN)           :: global_tran

 SUBROUTINE gset_nppa_loc_tran (selcrit, loc_tran)
  TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
  TYPE(Gmatrix_23_local), INTENT(IN)            :: loc_tran
```

```
SUBROUTINE gset_nppa_pat_size (selcrit, pat_size)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gpat_size), INTENT(IN)                   :: pat_size


SUBROUTINE gset_nppa_pat_ref_pt (selcrit, pat_ref_pt)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gwc_point), INTENT(IN)                   :: pat_ref_pt


SUBROUTINE gset_nppa_text_height (selcrit, text_height)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gtext_height), INTENT(IN)                :: text_height


SUBROUTINE gset_nppa_text_up_vec (selcrit, text_up_vec)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gtext_up_vec), INTENT(IN)                :: text_up_vec


SUBROUTINE gset_nppa_text_skew (selcrit, text_skew)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gtext_skew), INTENT(IN)                  :: text_skew


SUBROUTINE gset_nppa_text_path (selcrit, text_path)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gtext_path_id), INTENT(IN)               :: text_path


SUBROUTINE gset_nppa_text_align (selcrit, text_align)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gtext_align), INTENT(IN)                 :: text_align


SUBROUTINE gset_nppa_curve_ind (selcrit, curve_ind)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gcurve_ind), INTENT(IN)                  :: curve_ind


SUBROUTINE gset_nppa_curve_asfs (selcrit, curve_asfs)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gcurve_asfs), INTENT(IN)                 :: curve_asfs


SUBROUTINE gset_nppa_curve_type (selcrit, curve_type)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gcurve_type), INTENT(IN)                 :: curve_type


SUBROUTINE gset_nppa_curve_width (selcrit, curve_width)
 TYPE(Gsel_crit) , INTENT(IN)                  :: selcrit
 TYPE(Gcurve_width), INTENT(IN)                :: curve_width
```

```
SUBROUTINE gset_nppa_curve_colr_spec (selcrit, curve_colr_spec)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gcurve_colr_spec), INTENT(IN)          :: curve_colr_spec


SUBROUTINE gset_nppa_marker_ind (selcrit, marker_ind)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gmarker_ind), INTENT(IN)               :: marker_ind


SUBROUTINE gset_nppa_marker_asfs (selcrit, marker_asfs)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gmarker_asfs), INTENT(IN)              :: marker_asfs


SUBROUTINE gset_nppa_marker_type (selcrit, marker_type)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gmarker_type), INTENT(IN)              :: marker_type


SUBROUTINE gset_nppa_marker_size (selcrit, marker_size)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gmarker_size), INTENT(IN)              :: marker_size


SUBROUTINE gset_nppa_marker_colr_spec (selcrit, marker_colr_spec)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gcolr_spec), INTENT(IN)                :: marker_colr_spec


SUBROUTINE gset_nppa_area_ind (selcrit, area_ind)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Garea_ind), INTENT(IN)                 :: area_ind


SUBROUTINE gset_nppa_area_asfs (selcrit, area_asfs)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Garea_asfs), INTENT(IN)                :: area_asfs


SUBROUTINE gset_nppa_int_style (selcrit, int_style)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gint_style_id), INTENT(IN)             :: int_style


SUBROUTINE gset_nppa_int_style_ind (selcrit, int_style_ind)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gint_style_ind), INTENT(IN)            :: int_style_ind


SUBROUTINE gset_nppa_int_colr_spec (selcrit, int_colr_spec)
 TYPE(Gsel_crit) , INTENT(IN)                :: selcrit
 TYPE(Gint_colr_spec), INTENT(IN)            :: int_colr_spec
```

```
 SUBROUTINE gset_nppa_edge_flag (selcrit, edge_flag)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gedge_flag_id), INTENT(IN)              :: edge_flag


 SUBROUTINE gset_nppa_edge_type (selcrit, edge_type)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gedge_type), INTENT(IN)                 :: edge_type


 SUBROUTINE gset_nppa_edge_width (selcrit, edge_width)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gedge_width), INTENT(IN)                :: edge_width


 SUBROUTINE gset_nppa_edge_colr_spec (selcrit, edge_colr_spec)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gedge_colr_spec), INTENT(IN)            :: edge_colr_spec


 SUBROUTINE gset_nppa_char_ind (selcrit, char_ind)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gchar_ind), INTENT(IN)                  :: char_ind


 SUBROUTINE gset_nppa_char_asfs (selcrit, char_asfs)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gchar_asfs), INTENT(IN)                 :: char_asfs


 SUBROUTINE gset_nppa_char_font_prec (selcrit, char_font_prec)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gfont_prec), INTENT(IN)                 :: char_font_prec


 SUBROUTINE gset_nppa_char_expan (selcrit, char_expan)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gchar_expan), INTENT(IN)                :: char_expan


 SUBROUTINE gset_nppa_char_space (selcrit, char_space)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gchar_space), INTENT(IN)                :: char_space


 SUBROUTINE gset_nppa_char_colr_spec (selcrit, char_colr_spec)
  TYPE(Gsel_crit) , INTENT(IN)                 :: selcrit
  TYPE(Gchar_colr_spec), INTENT(IN)            :: char_colr_spec


END INTERFACE
```

Remove set of scissors from NDC picture

```
SUBROUTINE gremove_set_scissor_ndc_pic ( scissor_id_set, selcrit )
  TYPE(Gscissor_id_set), INTENT(IN) :: scissor_id_set
  TYPE(Gsel_crit), INTENT(IN)       :: selcrit
```

Add set of scissors to NDC picture

```
SUBROUTINE gadd_set_scissor_ndc_pic ( scissor_set, selcrit )
  TYPE(Gscissor_set), INTENT(IN) :: scissor_set
  TYPE(Gsel_crit), INTENT(IN)    :: selcrit
```

Reorder NDC picture

```
SUBROUTINE greorder_ndc_pic ( src_sel, ref_sel, rel_pos )
  TYPE(Gsel_crit), INTENT(IN) :: src_sel
  TYPE(Gsel_crit), INTENT(IN) :: ref_sel
  TYPE(Gpos_id), INTENT(IN)   :: rel_pos
```

11.7 Metafile Functions

Copy NDC picture to NDC metafile

```
SUBROUTINE gcopy_ndc_pic_ndc_mf  ( ndc_mf_spec, picid, selcrit, nameset )
  CHARACTER(LEN=*), INTENT(IN) :: ndc_mf_spec
  INTEGER, INTENT(IN)          :: picid
  TYPE(Gsel_crit), INTENT(IN)  :: selcrit
  TYPE(Gname_set), INTENT(IN)  :: nameset
```

Copy NDC metafile picture to NDC picture

```
SUBROUTINE gcopy_ndc_mf_pic_ndc_pic (ndc_mf_spec, picid, nameset)
  CHARACTER(LEN=*), INTENT(IN) :: ndc_mf_spec
  INTEGER, INTENT(IN)          :: picid
  TYPE(Gname_set), INTENT(IN)  :: nameset
```

11.8 Picture part store functions

Open picture part

```
SUBROUTINE gopen_pic_part (partname)
  INTEGER, INTENT(IN) :: partname
```

Close picture part

```
SUBROUTINE gclose_pic_part ()
```

Archive picture part

```
SUBROUTINE gar_pic_part (part_name, ar_spec, ar_name)
  INTEGER, INTENT(IN)           :: part_name
  CHARACTER(LEN=*), INTENT(IN)  :: ar_spec
  INTEGER, INTENT(IN)           :: ar_name
```

Retrieve picture part from archive

```
SUBROUTINE gret_pic_part_ar (ar_spec, ar_name, part_name)
  CHARACTER (LEN=*), INTENT(IN) :: ar_spec
  INTEGER, INTENT(IN)           :: ar_name
  INTEGER, INTENT(IN)           :: part_name
```

Reopen picture part

```
SUBROUTINE greopen_pic_part (part_name)
  INTEGER, INTENT(IN) :: part_name
```

Append picture part

```
SUBROUTINE gappend_pic_part (src_partname, sink_partname, globaltran,   &
                             globalmode,  localtran, localmode, nameset )
  INTEGER, INTENT(IN)                   :: src_partname
  INTEGER, INTENT(IN)                   :: sink_partname
  TYPE(Gmatrix_23), INTENT(IN)          :: globaltran
  TYPE(Gtran_mode_id), INTENT(IN)       :: globalmode
  TYPE(Gmatrix_23), INTENT(IN)          :: localtran
  TYPE(Gtran_mode_id), INTENT(IN)       :: localmode
  TYPE(Gname_set), INTENT(IN)           :: nameset
```

Rename picture part

```
SUBROUTINE grename_pic_part (old_name, new_name)
  INTEGER, INTENT(IN) :: old_name
  INTEGER, INTENT(IN) :: new_name
```

Delete picture part

```
SUBROUTINE gdel_pic_part (part_name)
  INTEGER, INTENT(IN) :: part_name ! picture part name to delete
```

Copy picture part from picture part store

```
SUBROUTINE gcopy_pic_part_pic_part_store (part_name, sel_crit, global_tran, &
                                          global_mode, local_tran, local_mode,&
                                          name_set, scissor_sel)
  INTEGER, INTENT(IN)                   :: part_name
  TYPE(Gsel_crit), INTENT(IN)           :: sel_crit
  TYPE(Gmatrix_23), INTENT(IN)          :: global_tran
  TYPE(Gtran_mode_id), INTENT(IN)       :: global_mode
  TYPE(Gmatrix_23), INTENT(IN)          :: local_tran
  TYPE(Gtran_mode_id), INTENT(IN)       :: local_mode
  TYPE(Gname_set), INTENT(IN)           :: name_set
  TYPE(Gscissor_sel_id), INTENT(IN)     :: scissor_sel
```

Copy NDC picture to picture part store

```
SUBROUTINE gcopy_ndc_pic_pic_part_store (sel_crit, part_name, name_set)
  TYPE (Gsel_crit), INTENT(IN)  :: sel_crit
  INTEGER, INTENT(IN)           :: part_name
  TYPE (Gname_set), INTENT(IN)  :: name_set
```

11.9 Input Functions

Set logical input device mode

```
SUBROUTINE gset_in_dev_mode ( dev_id, mode )
  TYPE(Gdev_id), INTENT(IN)     :: dev_id
  TYPE(Gop_mode_id), INTENT(IN) :: mode
```

Request input

```
SUBROUTINE greq_in ( dev_id, in_st, in_val )
  TYPE(Gdev_id),INTENT(IN)    :: dev_id
  TYPE(Gin_st_id),INTENT(OUT) :: in_st
  TYPE(Gin_val),INTENT(OUT)   :: in_val
```

Sample input

```
SUBROUTINE gsample_in ( dev_id, in_val  )
  TYPE(Gdev_id),INTENT(IN)  :: dev_id
  TYPE(Gin_val),INTENT(OUT) :: in_val
```

Await input

```
SUBROUTINE gawait_in ( timeout, events )
  REAL, INTENT(IN)                   :: timeout
  TYPE(Gdev_id_val_set),INTENT(OUT) :: events
```

Flush device events

```
SUBROUTINE gflush_dev_events  (dev_id)
  TYPE(Gdev_id), INTENT(IN) :: dev_id
```

11.10 Font and glyph functions

Set font index mapping

```
SUBROUTINE gset_font_ind_map ( font_index, font_name )
  INTEGER, INTENT(IN)          :: font_index
  CHARACTER(LEN=*), INTENT(IN) :: font_name
```

Get glyph name

```
SUBROUTINE gget_glyph_name ( font_index, spec_char, spec_glyph  )
  INTEGER, INTENT(IN)           :: font_index
  CHARACTER(LEN=1), INTENT(IN)  :: spec_char
  CHARACTER(LEN=*), INTENT(OUT) :: spec_glyph
```

Set character code

```
SUBROUTINE gset_char_code ( font_index, spec_char, spec_glyph  )
  INTEGER, INTENT(IN)          :: font_index
  CHARACTER(LEN=1), INTENT(IN) :: spec_char
  CHARACTER(LEN=*), INTENT(IN) :: spec_glyph
```

11.11 Audit and Playback Functions

Audit

```
SUBROUTINE gaudit (audit_id, audit_op)
  INTEGER, INTENT(IN)             :: audit_id
  TYPE(Gaudit_op_id), INTENT(IN)  :: audit_op
```

Write user record to audit

```
SUBROUTINE gwrite_user_rec_audit (audit_id, rec)
  INTEGER, INTENT(IN)                :: audit_id
  TYPE(Gaudit_user_data), INTENT(IN) :: rec
```

Playback

```
SUBROUTINE gplayback  (audit_id, playback_op)
  INTEGER, INTENT(IN)             :: audit_id
  TYPE(Gplayback_ops), INTENT(IN) :: playback_op
```

Read item function name from audit

```
SUBROUTINE gread_item_func_name_audit (audit_id, func_name)
  INTEGER, INTENT(IN)      :: audit_id
  TYPE(Gfn_id),INTENT(OUT) :: func_name
```

Read item from audit

```
SUBROUTINE gread_item_audit (audit_id, func_name, func_params)
  INTEGER, INTENT(IN)            :: audit_id
  TYPE(Gfn_id),INTENT(OUT)       :: func_name
  TYPE(Gfunc_params),INTENT(OUT) :: func_params
```

Process audit item

```
SUBROUTINE gproc_audit_item (audit_id, intr_op)
  INTEGER, INTENT(IN)              :: audit_id
  TYPE(Gprocess_op_id), INTENT(IN) :: intr_op
```

11.12 Inquiry Functions

Inquire operating state entry

```
SUBROUTINE ginq_op_st_entry ( entry_name, error_ind, entry_value )
  TYPE(Gop_st_list_id), INTENT(IN) :: entry_name
  INTEGER,INTENT(OUT)              :: error_ind
  TYPE(Gop_st_list),INTENT(OUT)    :: entry_value
```

Inquire GKS description table entry

```
SUBROUTINE ginq_gdt_entry (ent_name, err_ind, ent_val)
  TYPE(Ggdt_id), INTENT(IN) :: ent_name
  INTEGER,INTENT(OUT)       :: err_ind
  TYPE(Ggdt),INTENT(OUT)    :: ent_val
```

Inquire GKS state list entry

```
SUBROUTINE ginq_gsl_entry (ent_name, err_ind, ent_val)
  TYPE(Ggsl_id), INTENT(IN) :: ent_name
  INTEGER,INTENT(OUT)       :: err_ind
  TYPE(Ggsl),INTENT(OUT)    :: ent_val
```

Inquire input queue overflow

```
SUBROUTINE ginq_in_overf (err_ind, &
                          dev_id   )
  INTEGER,INTENT(OUT)       :: err_ind
  TYPE(Gdev_id),INTENT(OUT) :: dev_id
```

11.13 Utility Functions

Evaluate transformation matrix

```
SUBROUTINE geval_tran_matrix (fixed_pt, shift_vec, rot_angle, scale_facs, &
                              tran_mat)
  TYPE(Gwc_ndc_point), INTENT(IN) :: fixed_pt
  TYPE(gwc_ndc_point), INTENT(IN) :: shift_vec
  REAL, INTENT(IN)                :: rot_angle
  TYPE(Gscale_facs), INTENT(IN)   :: scale_facs
  TYPE(Gmatrix_23),INTENT(OUT)    :: tran_mat
```

Accumulate transformation matrix

```
SUBROUTINE gaccum_tran_matrix (in_mat, fixed_pt, shift_vec, rot_angle, &
                               scale_facs, tran_mat )
  TYPE(Gmatrix_23), INTENT(IN)    :: in_mat
  TYPE(Gwc_ndc_point), INTENT(IN) :: fixed_pt
  TYPE(Gwc_ndc_point), INTENT(IN) :: shift_vec
  REAL, INTENT(IN)                :: rot_angle
  TYPE(Gscale_facs), INTENT(IN)   :: scale_facs
  TYPE(Gmatrix_23),INTENT(OUT)    :: tran_mat
```

Evaluate circle

```
SUBROUTINE geval_circle ( centre, radius, circle )
 TYPE(Gwc_point), INTENT(IN)  :: centre
 REAL, INTENT(IN)             :: radius
 TYPE(Gmatrix_33),INTENT(OUT) :: circle
```

Evaluate ellipse

```
SUBROUTINE geval_ell (ellip_spec, ellip)
  TYPE(Gellip_spec), INTENT(IN) :: ellip_spec
  TYPE(Gmatrix_33),INTENT(OUT)  :: ellip
```

Evaluate circular arc 3 point

```
SUBROUTINE geval_circ_arc_3_point ( start_pt, imm_pt, end_pt, circular_arc  )
 TYPE(Gwc_point), INTENT(IN)  :: start_pt
 TYPE(Gwc_point), INTENT(IN)  :: imm_pt
 TYPE(Gwc_point), INTENT(IN)  :: end_pt
 TYPE(Gconic_sec),INTENT(OUT) :: circular_arc
```

Evaluate circular arc centre

```
SUBROUTINE geval_circ_arc_ctr (centre, start_vec, end_vec, sense_flag, &
                               radius, circular_arc)
 TYPE(Gwc_point), INTENT(IN)       :: centre
 TYPE(Gvec), INTENT(IN)            :: start_vec
 TYPE(Gvec), INTENT(IN)            :: end_vec
 TYPE(Gsense_flag_id), INTENT(IN)  :: sense_flag
 REAL, INTENT(IN)                  :: radius
 TYPE(Gconic_sec),INTENT(OUT)      :: circular_arc
```

Evaluate elliptic arc

```
SUBROUTINE geval_ell_arc ( centre, ellip_spec, start_vec, &
                           end_vec, conic_arc )
 TYPE(Gwc_point), INTENT(IN)   :: centre
 TYPE(Gellip_spec), INTENT(IN) :: ellip_spec
 TYPE(Gvec), INTENT(IN)        :: start_vec
 TYPE(Gvec), INTENT(IN)        :: end_vec
 TYPE(Gconic_sec),INTENT(OUT)  :: conic_arc
```

Evaluate hyperbolic arc

```
SUBROUTINE geval_hyp_arc ( hyperbola_spec, start_vec, end_vec, conic_arc )
 TYPE(Ghyperbola_spec), INTENT(IN) :: hyperbola_spec
 TYPE(Gvec), INTENT(IN)            :: start_vec
 TYPE(Gvec), INTENT(IN)            :: end_vec
 TYPE(Gconic_sec),INTENT(OUT)      :: conic_arc
```

Evaluate parabolic arc

```
SUBROUTINE geval_par_arc ( parabola_spec, conic_arc )
 TYPE(Gparabola_spec), INTENT(IN) :: parabola_spec
 TYPE(Gconic_sec),INTENT(OUT)     :: conic_arc
```

Evaluate wc ordinate

```
SUBROUTINE geval_wc_ord ( wc_ordinate, ord_sel, ndc_val )
 REAL, INTENT(IN)              :: wc_ordinate
 TYPE(Gord_sel_id), INTENT(IN) :: ord_sel
 REAL,INTENT(OUT)              :: ndc_val
```

11.14 Workstation Functions

11.14.1 Control Functions

Open workstation

```
SUBROUTINE gopen_ws (wsid, wsspec, wstype)
  INTEGER, INTENT(IN)           :: wsid
  CHARACTER (LEN=*), INTENT(IN) :: wsspec
  TYPE(Gws_type), INTENT(IN)    :: wstype
```

Close workstation

```
SUBROUTINE gclose_ws (wsid)
  INTEGER, INTENT(IN) :: wsid
```

Save workstation state list

```
SUBROUTINE gsave_wsl (wsid, ent_names, ent_vals )
  INTEGER, INTENT(IN)                  :: wsid
  TYPE(Gwsl_id), POINTER, DIMENSION (:) :: ent_names
  TYPE(Gwsl), POINTER, DIMENSION (:)    :: ent_vals
```

Restore workstation state list

```
SUBROUTINE grestore_wsl (wsid, ent_vals)
  INTEGER, INTENT(IN)                 :: wsid
  TYPE(Gwsl), POINTER, DIMENSION (:)   :: ent_vals
```

Get workstation status

```
SUBROUTINE gget_ws_status ( ws_id, ws_st  )
 INTEGER, INTENT(IN)          :: ws_id
 TYPE(Gws_st_id), INTENT(OUT) :: ws_st
```

Remove backdrop

```
SUBROUTINE gremove_backdrop (wsid)
  INTEGER, INTENT(IN) :: wsid
```

Set representation

```
INTERFACE gset_rep

 SUBROUTINE gset_rep_curv_bun ( wsid, rep_ind, curv_bun)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Gcurve_bun), INTENT(IN)       :: curv_bun


 SUBROUTINE gset_rep_mark_bun ( wsid, rep_ind, mark_bun)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Gmarker_bun), INTENT(IN)      :: mark_bun


 SUBROUTINE gset_rep_area_bun ( wsid, rep_ind, area_bun)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Garea_bun), INTENT(IN)        :: area_bun


 SUBROUTINE gset_rep_char_bun ( wsid, rep_ind, char_bun)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Gchar_bun), INTENT(IN)        :: char_bun


 SUBROUTINE gset_rep_patt_array ( wsid, rep_ind, patt_array)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Gcolr_array_spec), INTENT(IN) :: patt_array


 SUBROUTINE gset_rep_colr ( wsid, rep_ind, colr)
  INTEGER, INTENT(IN)                :: wsid
  INTEGER, INTENT(IN)                :: rep_ind
  TYPE(Gcolr_coord), INTENT(IN)      :: colr


END INTERFACE
```

Set view priority

```
SUBROUTINE gset_view_pri ( wsid, view_ind, ref_view_ind, rel_pri )
  INTEGER, INTENT(IN)       :: wsid
  INTEGER, INTENT(IN)       :: view_ind
  INTEGER, INTENT(IN)       :: ref_view_ind
  TYPE(Gpri_id), INTENT(IN) :: rel_pri
```

Set view selection criterion

```
SUBROUTINE gset_view_sel_crit ( wsid, view_ind, sel_crit  )
  INTEGER, INTENT(IN)         :: wsid
  INTEGER, INTENT(IN)         :: view_ind
  TYPE(Gsel_crit), INTENT(IN) :: sel_crit
```

Set view

```
SUBROUTINE gset_view ( wsid, view_ind, view_ori_matrix, view_map_matrix, &
                       view_scissor )
  INTEGER, INTENT(IN)            :: wsid
  INTEGER, INTENT(IN)            :: view_ind
  TYPE(Gmatrix_23), INTENT(IN)    :: view_ori_matrix
  TYPE(Gmatrix_23), INTENT(IN)    :: view_map_matrix
  TYPE(Gview_scissor), INTENT(IN) :: view_scissor
```

Set workstation visual effects

```
SUBROUTINE gset_ws_vis_effects (wsid, vis_eff)
  INTEGER, INTENT(IN)              :: wsid
  TYPE(Gvis_eff_st_id), INTENT(IN) :: vis_eff
```

Set workstation window and viewport

```
SUBROUTINE gset_ws_win_vp (wsid, ldc_win, dc_vp)
  INTEGER, INTENT(IN)        :: wsid
  TYPE(Gndc_win), INTENT(IN) :: ldc_win
  TYPE(Gdc_vp), INTENT(IN)   :: dc_vp
```

Define logical input device

```
SUBROUTINE gdef_in_dev (dev_id, seq_measure_id, trigger_set, init_val)
  TYPE(Gdev_id), INTENT(IN)         :: dev_id
  TYPE(Gseq_measure_id), INTENT(IN) :: seq_measure_id
  TYPE(Gtrigger_set), INTENT(IN)    :: trigger_set
  TYPE(Ginit_val), INTENT(IN)       :: init_val
```

Initialize logical input device

```
SUBROUTINE ginit_in_dev (dev_id, init_val)
  TYPE(Gdev_id), INTENT(IN)   :: dev_id
  TYPE(Ginit_val), INTENT(IN) :: init_val
```

Set workstation selection criterion

```
SUBROUTINE gset_ws_sel_crit (wsid, sel_type, sel_crit)
  INTEGER, INTENT(IN)             :: wsid
  TYPE(Gsel_crit_id), INTENT(IN) :: sel_type
  TYPE(Gsel_crit), INTENT(IN)    :: sel_crit
```

Copy realized picture to realized metafile

```
SUBROUTINE gcopy_real_pic_real_mf (wsid, rld_mf_spec, picid)
  INTEGER, INTENT(IN)           :: wsid
  CHARACTER(LEN=*), INTENT(IN)  :: rld_mf_spec
  INTEGER, INTENT(IN)           :: picid
```

Copy blank realized picture to realized metafile

```
SUBROUTINE gcopy_blank_real_pic_real_mf (wsid, rld_mf_spec, picid)
  INTEGER, INTENT(IN)           :: wsid
  CHARACTER(LEN=*), INTENT(IN)  :: rld_mf_spec
  INTEGER, INTENT(IN)           :: picid
```

Copy realized metafile picture to backdrop

```
SUBROUTINE gcopy_real_mf_pic_backdrop (wsid, rld_mf_spec, picid)
  INTEGER , INTENT(IN)          :: wsid
  CHARACTER(LEN=*), INTENT(IN)  :: rld_mf_spec
  INTEGER, INTENT(IN)           :: picid
```

Message

```
SUBROUTINE gmessage (wsid, char_str)
  INTEGER, INTENT(IN)          :: wsid
  CHARACTER(LEN=*), INTENT(IN) :: char_str
```

11.14.2 Inquiry Functions

Inquire workstation state list entry

```
SUBROUTINE ginq_wsl_entry (wsid, ent_name, ent_type, err_ind, ent_val)
  INTEGER, INTENT(IN)                  :: wsid
  TYPE(Gwsl_id), INTENT(IN)            :: ent_name
  TYPE(Gset_realized_id), INTENT(IN)   :: ent_type
  INTEGER, INTENT(OUT)                 :: err_ind
  TYPE(Gwsl), INTENT(OUT)              :: ent_val
```

Inquire generic workstation description table entry

```
SUBROUTINE ginq_gen_wdt_entry (wstype, entry_name, error_ind, entry_val)
  INTEGER, INTENT(IN)       :: wstype
  TYPE(Gwdt_id), INTENT(IN) :: entry_name
  INTEGER, INTENT(OUT)      :: error_ind
  TYPE(Gwdt), INTENT(OUT)   :: entry_val
```

Inquire specific workstation description table entry

```
SUBROUTINE ginq_spec_wdt_entry (wsid, entry_name, error_ind, entry_val)
  INTEGER, INTENT(IN)       :: wsid
  TYPE(Gwdt_id), INTENT(IN) :: entry_name
  INTEGER, INTENT(OUT)      :: error_ind
  TYPE(Gwdt), INTENT(OUT)   :: entry_val
```

Inquire specific workstation description table entry state

```
SUBROUTINE ginq_spec_wdt_entry_st ( wsid, entry_name, error_ind, entry_state )
  INTEGER, INTENT(IN)       :: wsid
  TYPE(Gwdt_id), INTENT(IN) :: entry_name
  INTEGER, INTENT(OUT)      :: error_ind
  INTEGER, INTENT(OUT)      :: entry_state
```

Reset specific workstation description table entry state

```
SUBROUTINE gres_spec_wdt_entry_st (wsid, entry_name)
  INTEGER, INTENT(IN)       :: wsid
  TYPE(Gwdt_id), INTENT(IN) :: entry_name
```

11.14.3 Retrieval functions

Get text extent

```
SUBROUTINE gget_text_extent (wsid, text_pos, char_str, err_ind, concat_pt, &
                             xmin, xmax, ymin, ymax)
  INTEGER, INTENT(IN)           :: wsid
  TYPE(Gwc_point), INTENT(IN)   :: text_pos
  CHARACTER (LEN=*), INTENT(IN) :: char_str
  INTEGER,INTENT(OUT)           :: err_ind
  TYPE(Gwc_point),INTENT(OUT)   :: concat_pt
  TYPE(Gwc_point),INTENT(OUT)   :: xmin
  TYPE(Gwc_point),INTENT(OUT)   :: xmax
  TYPE(Gwc_point),INTENT(OUT)   :: ymin
  TYPE(Gwc_point),INTENT(OUT)   :: ymax
```

11.14.4 Viewing utility functions

Evaluate view orientation matrix

```
SUBROUTINE geval_view_ori_matrix ( view_ref_pt, view_up_vec, view_ori_mat  )
  TYPE(Gndc_point), INTENT(IN)   :: view_ref_pt
  TYPE(Gview_up_vec), INTENT(IN) :: view_up_vec
  TYPE(Gmatrix_23), INTENT(OUT)  :: view_ori_mat
```

Evaluate view mapping matrix

```
SUBROUTINE geval_view_map_matrix ( win_lim, vp_lim, view_map_mat  )
  TYPE(Gndc_win), INTENT(IN)    :: win_lim
  TYPE(Gldc_vp), INTENT(IN)     :: vp_lim
  TYPE(Gmatrix_23), INTENT(OUT) :: view_map_mat
```

Convert colour

```
SUBROUTINE gconv_colr(wsid, colr_spec, colr_model, colr_val, conv_flag)
  INTEGER, INTENT(IN)              :: wsid
  TYPE(Gcolr_spec), INTENT(IN)     :: colr_spec
  TYPE(Gcolr_model_id), INTENT(IN) :: colr_model
  TYPE(Gcolr_spec), INTENT(OUT)    :: colr_val
  TYPE(Gconv_flag_id), INTENT(OUT) :: conv_flag
```

11.15 Segment and Workstation Activation Functions

11.15.1 Segment Functions

Create segment

```
SUBROUTINE gcreate_seg (seg_name)
  INTEGER, INTENT(IN) :: seg_name
```

Close segment

```
SUBROUTINE gclose_seg ()
```

Rename segment

```
SUBROUTINE grename_seg (old_name, new_name)
  INTEGER, INTENT(IN) :: old_name
  INTEGER, INTENT(IN) :: new_name
```

Delete segment

```
SUBROUTINE gdel_seg (seg_name)
  INTEGER, INTENT(IN) :: seg_name
```

Delete segment from workstation

```
SUBROUTINE gdel_seg_ws (wsid, seg_name)
  INTEGER, INTENT(IN) :: wsid
  INTEGER, INTENT(IN) :: seg_name
```

Associate segment with workstation

```
SUBROUTINE gassoc_seg_ws (wsid, seg_name)
  INTEGER, INTENT(IN) :: wsid
  INTEGER, INTENT(IN) :: seg_name
```

Copy segment to workstation

```
SUBROUTINE gcopy_seg_ws (wsid, seg_name)
  INTEGER, INTENT(IN) :: wsid
  INTEGER, INTENT(IN) :: seg_name
```

Insert segment

```
SUBROUTINE ginsert_seg (seg_name, tran_mat)
  INTEGER, INTENT(IN)          :: seg_name
  TYPE(Gmatrix_23), INTENT(IN) :: tran_mat
```

Set segment attribute

```
SUBROUTINE gset_seg_attr (seg_name, seg_attr_val)
  INTEGER, INTENT(IN)             :: seg_name
  TYPE(Gseg_attr_val), INTENT(IN) :: seg_attr_val
```

11.15.2 Workstation Activation Functions

Activate workstation

```
SUBROUTINE gactivate_ws (wsid)
  INTEGER, INTENT(IN) :: wsid
```

Deactivate workstation

```
SUBROUTINE gdeactivate_ws (wsid)
  INTEGER, INTENT(IN) :: wsid
```

Clear workstation

```
SUBROUTINE gclear_ws (wsid)
  INTEGER, INTENT(IN) :: wsid
```

11.15.3 Utility Functions

Get mapped segment name

```
SUBROUTINE gget_map_seg_name (seg_name, map_name)
  INTEGER, INTENT(IN)  :: seg_name
  INTEGER, INTENT(OUT) :: map_name
```

Get mapped workstation identifier

```
SUBROUTINE gget_map_ws_id (wsid, map_name)
  INTEGER, INTENT(IN)          :: wsid
  TYPE(Ggen_name), INTENT(OUT) :: map_name
```

A FORTRAN 90 Examples

```
module fbindstring
contains
 function gstring(chr)
  character, pointer, dimension(:) :: gstring
  character(len=*),intent(in) :: chr
  ! returns the string consisting of the characters char
  integer                     :: lc
  lc=len(chr)
  allocate(gstring(1:lc))
  do i=1,lc
    gstring(i) = chr(i:i)
  end do
 end function gstring
end module fbindstring

program star

 use fbindtypes
 use fbindsubs
 use fbindstring

 implicit none

! Variables needed

  character(len=7)  :: err_file
  integer           :: err_ind

  integer           :: wsid
  character(len=80) :: wsspec
  type(Gws_type)    :: wstype

  type(Gwsl)        :: ent_val
  type(Gndc_win)    :: wswn = Gndc_win(-1.25,-1.25,1.25,1.25)

  type(Gint_colr_spec)   :: col_yellow
  type(Gchar_colr_spec)  :: col_white

  type(Gint_style_id) :: int_style

  type(Gtext_align)  :: text_align

  type(Gset_fill)   :: star_coords

  type(Gtext)       :: star_title


!       Open GKS and activate the workstations.

        wsid = 1

!       Need to specify the implmentation dependent versions
!       of wsspec and wstype

        err_file = 'err.log'
```

```
        call gopen_gks (err_file)
        call gopen_ws (wsid, wsspec, wstype)
        call gactivate_ws (wsid)

!       Center the window around the origin.

        call ginq_wsl_entry (wsid, Gwsl_curwsvp, Gset_realized_set, &
                             err_ind, ent_val)

        call gset_ws_win_vp (wsid, wswn, ent_val%ws_vp)

        call gset_norm_tran_num (1)

!       Fill the star with solid yellow.

        col_yellow%colr_spec = Gcolr_spec_rgb
        col_yellow%rgb = Gcolr_rgb(1.0,1.0,0.0)
        int_style=Gint_style_solid

        call gset_prim_attr (int_style)
        call gset_prim_attr (col_yellow)

!       Draw the star.

        allocate (star_coords%seqs(1))
        allocate (star_coords%seqs(1)%point(5))

        star_coords%seqs(1)%point%x = &
              (/0.951057,-0.951057,0.587785,0.0,-0.587785/)
        star_coords%seqs(1)%point%y = &
              (/0.309017,0.309017,-0.951057,1.0,-0.951057/)

        call gcreate_out_prim (star_coords)

!       Select large chracters centered under the star.

        call gset_prim_attr (Gtext_height(0.15))

        text_align%hor_align = Ghor_align_centre
        text_align%vert_align = Gvert_align_half

        call gset_prim_attr(text_align)

        col_white%colr_spec = Gcolr_spec_rgb
        col_white%rgb = Gcolr_rgb(1.0,1.0,1.0)

        call gset_prim_attr (col_white)

!       Draw the title.

        star_title%charstring = gstring('Star')
        star_title%ref_point%x = 0.0
        star_title%ref_point%y = -1.1

        call gcreate_out_prim (star_title)
```

```
!       Close the workstation and shut down GKS.

        call gdeactivate_ws (wsid)
        call gclose_ws (wsid)
        call gclose_gks ()

        end program star
```

B Index of Types and Function Names

— I —

— M —

— O —

— P —

— R —

— S —

— W —