INTERNATIONAL STANDARD

ISO/IEC 9593-4

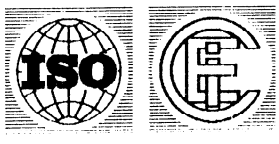
First edition
1991-12-15

Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings —

Part 4:
C

Technologies de l'information — Infographie — Interfaces langage entre un programme d'application et son support graphique —

Partie 4: C



Reference number
ISO/IEC 9593-4:1991(E)

Contents

Foreword

Foreword

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9593-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9593 consists of the following parts, under the general title *Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings* :

— *Part 1 : FORTRAN 77*
— *Part 3 : ADA*
— *Part 4 : C*

Annexes A, B, C, D, and E of this part of ISO/IEC 9593 are for information only.

Introduction

The Programmer's Hierarchical Interactive Graphics System (PHIGS), the functional description of which is given in ISO/IEC 9592-1, 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 9593 is to define a standard binding for the C computer programming language.

Information technology – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings –

Part 4:

C

1 Scope

The Programmer's Hierarchical Interactive Graphics System (PHIGS), ISO/IEC 9592-1, specifies a language independent nucleus of a graphics system. For integration into a programming language, PHIGS is embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 9593 specifies such a language dependent layer for the C language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9593. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9593 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 9592-1 : 1989, *Information processing systems – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 1:Functional description.*

ISO/IEC 9899 : 1990, *Programming Languages – C.*

ISO/IEC TR 9973 : 1988, *Information processing – Procedures for Registration of Graphical Items.*

3 The C language binding of PHIGS

3.1 Conformance

This binding incorporates the rules of conformance defined in the PHIGS Standard (ISO/IEC 9592) for PHIGS implementations, with those additional requirements specifically defined for C language implementations of PHIGS.

The following criteria are established for determining conformance of an implementation to this binding:

In order to conform, an implementation of the C binding of PHIGS shall implement ISO/IEC 9592-1. It shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 9593.

Thus, for example, the syntax of the function names shall be precisely as specified in this part of ISO/IEC 9593 and the parameters shall be of the data types stated in this part of ISO/IEC 9593.

3.2 Functions versus macros

An implementation may substitute macros for functions. However, the macros must be designed so that side-effects work properly.

3.3 Character strings

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

3.4 Function identifiers

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

In order to produce more readable C code, this part of ISO/IEC 9593 binds the PHIGS function names to identifiers which can be longer than six characters but are less than 31 characters. Implementations which must run in environments which honour less than 31 characters in external identifiers must include a set of `#defines` in the header file which equate the long identifiers defined in this part of ISO/IEC 9593 to a set of shorter identifiers which conform to the requirements of the implementation. An example of these shorter names can be found in annex C.

3.5 Registration

ISO/IEC 9592-1 reserves certain value ranges for registration[1] as graphical items. The registered graphical items will be bound to the C programming language (and other programming languages). The binding of registered items will be consistent with the bindings presented in this part of ISO/IEC 9593.

3.6 Identifiers for graphics items

ISO/IEC 9592-1 provides a mechanism to extend its functionality by the use of the graphical items GENERALIZED DRAWING PRIMITIVE, GENERALIZED DRAWING PRIMITIVE 3, GENERALIZED STRUCTURE ELEMENT and ESCAPE. These items are referenced via identifiers formed from the registration number of the item. This binding specifies the format of the identifiers but it does not specify the registration of the identifiers. The identifiers are used as arguments to the functions `pgdp`, `pgdp3`, `pgse`, and `pescape` and returned to the application by the functions `pinq_elem_content` and `pinq_cur_elem_content`.

[1] For the purpose of this International Standard and according to the rules for the designation and operation of registration authorities in the ISO/IEC JTC1 procedures, the ISO and IEC Councils have designated the National Institute of Standards and Technology (National Computer Systems Laboratory), A-266 Technology Building, Gaithersburg, MD 20899, USA, to act as registration authority.

An implementation may also represent these items as separate functions, but this is not required.

The format of the identifiers for graphical items is

"PFFF_Tn"

where:

FFF is the abbreviation of the PHIGS function name.

T is the type of the item, "R" for registered and "U" for unregistered.

n is the registration number (for unregistered items, the absolute value of the item number shall be used).

The formats for registered GENERALIZED DRAWING PRIMITIVE, GENERALIZED DRAWING PRIMITIVE 3, GENERALIZED STRUCTURE ELEMENT and ESCAPE graphical items are:

```
#define PGDP_Rn                 (n)
#define PGDP3_Rn                (n)
#define PGSE_Rn                 (n)
#define PESCAPE_Rn              (n)
```

For example, the identifier for registered GENERALIZED DRAWING PRIMITIVE item 12 would be:

```
#define PGDP_R12                (12)
```

The formats for unregistered GENERALIZED DRAWING PRIMITIVE, GENERALIZED DRAWING PRIMITIVE 3, GENERALIZED STRUCTURE ELEMENT and ESCAPE graphical items are:

```
#define PGDP_Un                 (-n)
#define PGDP3_Un                (-n)
#define PGSE_Un                 (-n)
#define PESCAPE_Un              (-n)
```

For example, the identifier for unregistered GENERALIZED DRAWING PRIMITIVE item -1 would be:

```
#define PGDP_U1                 (-1)
```

3.7 Return values

All PHIGS C functions return `void`.

3.8 Header files

C provides a mechanism to allow external files to be included in a compilation. The file `phigs.h` shall be included in any application program that intends to use PHIGS via the C binding.

Clause 5 of this binding describes the data types that shall be defined in the file `phigs.h`; clause 6 defines the macros that shall be in the file `phigs.h`. Additional implementation-dependent items may be placed in this file if needed.

The type `size_t` is used in the binding. It is defined in the standard header `<stddef.h>`. Therefore, the file `phigs.h` shall include `<stddef.h>` in order to define `size_t`.

The file `phigs.h` shall also contain external prototype declarations for all PHIGS C functions because they return `void`. For example, the declaration for the function `popen_phigs` is:

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

3.9 Memory management

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

These functions fall into two classes. One class of functions returns a simple, homogeneous list of elements. For example, the function INQUIRE STRUCTURE IDENTIFIERS returns a list of the structure identifiers in use. The other class returns complex, heterogeneous data structures. For example, the function INQUIRE LOCATOR DEVICE STATE returns the device state which includes a locator data record; the data record can contain arbitrarily complex implementation-defined data structures. The binding of these two classes of functions is described in detail below.

Additional binding-specific errors which relate to memory management are described in 3.11.3.

3.9.1 Inquiry functions which return simple lists

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

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

a) `num_elem_appl_list`: An integer input parameter which is the size of the application's list. The value of `num_elem_appl_list` indicates the number of list elements which will fit into the application's list. A value of 0 is valid and allows the application to determine the size of the implementation's list (which is returned via `num_elem_impl_list`) without having the implementation return any of the elements of its list.

b) `start_ind`: An integer input parameter which is the starting index into the implementation's list. (Index 0 is the first element of both the implementation's and application's list.) `start_ind` indicates the element in the implementation's list that is copied into index 0 of the application's list. Elements are copied sequentially from the implementation's list into the application's list until the application's list is full or there are no more elements in the implementation's list.

   If `start_ind` is out of range, error number 2200 (`PE_START_IND_INVAL`) is returned as the value of the error indicator parameter.

c) `num_elem_impl_list`: An output parameter which is a pointer to an integer. The implementation stores into this parameter the number of elements that are in the implementation's list.

Each function which returns a simple list has an output parameter *list*, which is a pointer to a structure with fields for the number of elements in the list and a pointer to the elements in the list. The type of *list* depends upon the function in which it is used. The implementation places the elements in the memory pointed to by *list->elems* and assigns the number of elements in *list->elems* to *list->num_elems*. If the implementation's list is of zero length, no data is placed in *list->elems* and *list->num_elems* is set to zero.

3.9.2 Inquiry functions which return complex data structures

The data returned by the element content functions and the functions which return input device data records can be complex in structure; they cannot be represented by a simple list of elements. It would be an onerous task for the application to allocate and to prepare data structures for these routines. In order to facilitate the task of using these inquiry functions, the binding defines a new resource, called a *Store*, to manage the memory for these functions.

A *Store* is used by the implementation to manage the memory needed by the functions which return complex data structures. The *Store* resource is opaque to the application. The application does not know the structure of the *Store* or how it is implemented. The *Store* is defined as a `void *`. The binding defines two new functions which create

(CREATE STORE, bound as `pcreate_store`) and destroy (DELETE STORE, bound as `pdel_store`) a *Store*.

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

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

For some functions, a *Store* is used to manage the memory for two or more distinct complex data structures. For example, in the function INQUIRE PICK DEVICE STATE, the *Store* manages the memory for the pick filter, the initial pick path, and the pick data record, all of which are returned to the application.

A *Store* continues to hold the information returned from the function until the *Store* is destroyed by the `pdel_store` function, or until the *Store* is used as an argument to a subsequent function which returns complex data. At that time, the old information is replaced with the new. Thus multiple calls to functions overwrite the contents of a *Store*. A *Store* only contains the results of the last function. An application may create more than one *Store*.

This binding defines two new errors that can occur when using or creating a *Store*; these errors are described in 3.11.3. For most functions using a *Store*, these and other errors are returned via the "error indicator" parameter. However, the functions RETRIEVE PATHS TO ANCESTORS, RETRIEVE PATHS TO DESCENDANTS and ESCAPE do not have an error indicator parameter. For these functions, the error reporting mechanism is used when an error is encountered. For these functions, the implementation shall, in addition to reporting the error, set the pointer to the returned data to NULL when an error occurs.

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

CREATE STORE (PHOP, *, *, *)

Parameters:

OUT	error indicator	I
OUT	store	STORE

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

002 Ignoring function, function requires state (PHOP, *, *, *).

2203 Ignoring function, error allocating Store.

Errors: none

DELETE STORE (PHOP, *, *, *)

Parameters:

OUT	error indicator	I
IN/OUT	store	STORE

Effect: Deletes the Store and all internal resources associated with it. The parameter *store* is set to NULL to signify that it is no longer valid. If an error is detected, the error indicator is set to the following error number:

002 Ignoring function, function requires state (PHOP, *, *, *).

Errors: none

3.9.3 Meaning of the size of an element

The functions INQUIRE CURRENT ELEMENT TYPE AND SIZE and INQUIRE ELEMENT TYPE AND SIZE return the size of an element. The size of an element is the size of a buffer, in bytes, that the application would have to allocate in order to contain the element. If the application would not have to allocate any memory, then 0 is returned.

3.10 Inquiries returning structure elements

PHIGS provides the ability for the application to inquire the contents of a structure element (through INQUIRE CURRENT ELEMENT CONTENT and INQUIRE ELEMENT CONTENT). PHIGS also allows the application to read in an archive file that can contain GDPs (both GENERALIZED DRAWING PRIMITIVES and GENERALIZED DRAWING PRIMITIVE 3's) and GSEs (GENERALIZED STRUCTURE ELEMENTs) which are not supported by the implementation. Thus it is possible for the application to inquire the contents of a structure element that contains a GDP or GSE data record that the implementation does not support.

In order to allow the inquiry of unsupported data records, the binding has introduced a field, called `unsupp`, to the GDP and GSE data record structures. The type of this field is `Pdata` which is a structure containing a field for the size of a block of data and another field for a pointer to the data. The `unsupp` field is used if the implementation does not support a GDP or a GSE.

3.11 Error handling

3.11.1 Application defined error handlers

An application can define the error handling function for the PHIGS implementation via the utility function SET ERROR HANDLER, which is bound as `pset_err_hand`. The definition for SET ERROR HANDLER is:

SET ERROR HANDLER (*, *, *, *)

Parameters:

IN	new error handling function	Function
OUT	old error handling function	Function

Effect: Sets the PHIGS error handling function to *new error handling function* and returns the previous function in *old error handling function*.

Errors: none

Application defined error handling functions accept the same arguments as the standard error handler. It may invoke the standard error logging function `perr_log`.

ISO/IEC 9592-1 defines the initial error handling function to be `perr_hand`; that is, the value of the parameter `old_err_hand` points to `perr_hand` when SET ERROR HANDLER is invoked for the first time.

When the application changes the error handling function, the implementation will invoke the new function when an error is detected. If the application calls the default error handling function `perr_hand`, `perr_hand` will always call the default error logging function `perr_log`; `perr_hand` does not call the error handling function specified in SET ERROR HANDLER.

3.11.2 Error codes

This binding defines, in 6.2, a set of constants for the PHIGS error numbers. Each error constant begins with the characters PE.

3.11.3 C specific PHIGS errors

This binding defines the following additional errors, beyond the ones described in ISO/IEC 9592-1.

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

3.12 Storage of two-dimensional data

3.12.1 Storage of matrices

ISO/IEC 9592-1 represents transformations as 3×3 and 4×4 matrices. This part of ISO/IEC 9593 binds a PHIGS 3×3 matrix to the type `Pmatrix` and a PHIGS 4×4 matrix to the type `Pmatrix3`.

The elements of a PHIGS 3×3 matrix are:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

and these elements are stored such that:

```
m[0][0] = a;    m[0][1] = b;    m[0][2] = c;
m[1][0] = d;    m[1][1] = e;    m[1][2] = f;
m[2][0] = g;    m[2][1] = h;    m[2][2] = i;
```

where `m` is of type `Pmatrix`.

The elements of a PHIGS 4×4 matrix are:

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

and these elements are stored such that:

```
m[0][0] = a;    m[0][1] = b;    m[0][2] = c;    m[0][3] = d;
m[1][0] = e;    m[1][1] = f;    m[1][2] = g;    m[1][3] = h;
m[2][0] = i;    m[2][1] = j;    m[2][2] = k;    m[2][3] = l;
m[3][0] = m;    m[3][1] = n;    m[3][2] = o;    m[3][3] = p;
```

where m is of type Pmatrix3.

For example, in order to store the projection transformation defined by:

$$x \rightarrow x'/w'$$
$$y \rightarrow y'/w'$$
$$z \rightarrow z'/w'$$

in PHIGS 4×4 matrix, the values shall be stored such that:

```
x' = p[0][0]*x + p[0][1]*y + p[0][2]*z + p[0][3];
y' = p[1][0]*x + p[1][1]*y + p[1][2]*z + p[1][3];
z' = p[2][0]*x + p[2][1]*y + p[2][2]*z + p[2][3];
w' = p[3][0]*x + p[3][1]*y + p[3][2]*z + p[3][3];
```

where p is of type Pmatrix3.

3.12.2 Storage of colour arrays

The entries for the Ppat_rep data type shall be stored such that the colour index at the (i,j)-th entry of a DX×DY array of colour indices is given by:

```
colr_ind(i,j) = colr_rect.colr_array[i + DX*j];
```

where colr_rect is of type Ppat_rep and

```
colr_rect.dims.size_x = DX;
colr_rect.dims.size_y = DY;
                    i = 0,...,DX-1;
                    j = 0,...,DY-1;
```

4 Tables

4.1 Abbreviation policy for construction of identifiers

The following policy is used to construct the identifiers for data types, structure fields, functions, and macros:

a) All identifiers in this part of ISO/IEC 9593 are abbreviated using the same abbreviations for every component and using underscores to denote blanks.

b) The plural of an expression is constructed by adding an "s" after its abbreviation; so, for example, "vector" is abbreviated to "vec" and "vectors" is abbreviated to "vecs"; if an expression is mapped to NULL then its plural is also abbreviated to NULL.

c) Digits are preceeded by underscores, except when the 3D version of a function or type is denoted, then no underscore is used.

d) The following are exceptions to the preceeding rules:

   1) The phrases "polyline", "polymarker", "fill area", "fill area set", and "annotation text" are not abbreviated when they are used in the primitive functions POLYLINE(3), POLYMARKER(3), FILL AREA(3), FILL AREA SET(3), and ANNOTATION TEXT RELATIVE(3).

   2) The phrase "generalized drawing primitives" is abbreviated "gdp".

   3) The phrase "generalized structure elements" is abbreviated "gse".

   4) The phrase "length" is not deleted when forming the macro identifiers for error numbers.

e) Construction of identifiers:

   1) function names:
      "p" (lower case) followed by abbreviated function name in lower case;

   2) data types:
      "P" (upper case) followed by abbreviated data type name in lower case;

   3) fields of data types:
      The name of a field comes from the description of the abstract parameter in PHIGS;

   4) Function macros:
      "Pfn_" followed by abbreviation of function names in lower case;

   5) Error macros:
      "PE_" followed by some abbreviated expression in upper case;

   6) Fields of enumeration types:
      "P" (upper case) followed by a prefix followed by an abbreviation of the field names; this prefix is constant for each enumeration field; all the fields are in upper case.

4.2 Table of abbreviations

Only the words which are abbreviated are listed in table 1. This part of ISO/IEC 9593 abbreviates the words which are in data type names, function names, data type fields names, formal parameter names, and macro names in order to form identifier names. The word "NULL" denotes those words which are deleted completely when forming identifier names.

Table 1 - Abbreviations ordered alphabetically

Word or Phrase	Abbreviation
alignment	align
ancestors	ances
and	NULL
annotation	anno

Table 1 - Abbreviations ordered alphabetically

Word or Phrase	Abbreviation
annotation text	anno
application	appl
archive	ar
aspect source flag	asf
at	NULL
attributes	attrs
availability	avail
available	avail
between	NULL
category	cat
character	char
cie l*u*v*	cieluv
classification	class
clipping	clip
colour	colr
composition	compose
concatenate	concat
conditional	cond
conflict	conf
conflicting	conf
connection	conn
control	ctrl
current	cur
default	def
deferral	defer
delete	del
dependency	dep
dependent	dep
descendants	descs
device	NULL
devices	NULL
device coordinate	dc
direction	dir
display	disp
dynamic	dyn
dynamics	dyns
element	elem
elements	elems
error	err
evaluate	eval
exclusion	excl
execute	exec
expansion	expan
facilities	facs
factor	NULL
from	NULL
generalized drawing primitive	gdp
generalized drawing primitives	gdp
generalized structure element	gse
generalized structure elements	gse
handling	hand

Table 1 - Abbreviations ordered alphabetically

Word or Phrase	Abbreviation
height	ht
highlighting	highl
horizontal	hor
hue lightness saturation	hls
hue saturation value	hsv
identifier	id
identifiers	ids
in	NULL
inclusion	incl
incremental	incr
independent	indep
index	ind
indicator	ind
indices	inds
individual	indiv
initialize	init
input	in
inquire	inq
integer	int
interior	int
invisibility	invis
length	NULL
lengths	NULL
locator	loc
logging	log
logical	NULL
luminance	lum
mapping	map
maximum	max
memory	mem
metafile	NULL
minimum	min
modelling	model
modification	mod
network	net
networks	nets
nominal	nom
normal	norm
number	num
of	NULL
operating	op
orientation	ori
output	out
overflow	overf
parallelogram	paral
pattern	pat
pending	pend
perspective	perspect
pointer	ptr
polyline	line
polymarker	marker

Table 1 - Abbreviations ordered alphabetically

Word or Phrase	Abbreviation
precision	prec
predefined	pred
primary	prim
priorities	pris
priority	pri
projection	proj
prompt	pr
queue	NULL
record	NULL
rectangle	rect
red green blue	rgb
reference	ref
references	refs
regeneration	regen
relative	rel
representation	rep
request	req
resolution	res
retrieve	ret
saturation	satur
scale factor	NULL
set of	NULL
simultaneous	simult
spacing	space
spatial	spa
state	st
structure	struct
structures	structs
supported	NULL
surface	surf
system	sys
to	NULL
transform	tran
transformation	tran
update	upd
used	NULL
valuator	val
value	NULL
vector	vec
vectors	vecs
vertical	vert
viewport	vp
volume	vol
which	NULL
window	win
workstation	ws
workstations	wss

4.3 Function names

4.3.1 List ordered alphabetically by bound name

Note – The PHIGS function INQUIRE HLHSR FACILITIES has been split into two functions, INQUIRE HLHSR MODE FACILITIES and INQUIRE HLHSR IDENTIFIER FACILITES.

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
padd_names_set	ADD NAMES TO SET
panno_text_rel	ANNOTATION TEXT RELATIVE
panno_text_rel3	ANNOTATION TEXT RELATIVE 3
pappl_data	APPLICATION DATA
par_all_structs	ARCHIVE ALL STRUCTURES
par_structs	ARCHIVE STRUCTURES
par_struct_nets	ARCHIVE STRUCTURE NETWORKS
pawait_event	AWAIT EVENT
pbuild_tran_matrix	BUILD TRANSFORMATION MATRIX
pbuild_tran_matrix3	BUILD TRANSFORMATION MATRIX 3
pcell_array	CELL ARRAY
pcell_array3	CELL ARRAY 3
pchange_struct_id	CHANGE STRUCTURE IDENTIFIER
pchange_struct_id_refs	CHANGE STRUCTURE IDENTIFIER AND REFERENCES
pchange_struct_refs	CHANGE STRUCTURE REFERENCES
pclose_ar_file	CLOSE ARCHIVE FILE
pclose_phigs	CLOSE PHIGS
pclose_struct	CLOSE STRUCTURE
pclose_ws	CLOSE WORKSTATION
pcompose_matrix	COMPOSE MATRIX
pcompose_matrix3	COMPOSE MATRIX 3
pcompose_tran_matrix	COMPOSE TRANSFORMATION MATRIX
pcompose_tran_matrix3	COMPOSE TRANSFORMATION MATRIX 3
pcopy_all_elems_struct	COPY ALL ELEMENTS FROM STRUCTURE
pcreate_store	CREATE STORE
pdel_all_structs	DELETE ALL STRUCTURES
pdel_all_structs_ar	DELETE ALL STRUCTURES FROM ARCHIVE
pdel_elem	DELETE ELEMENT
pdel_elems_labels	DELETE ELEMENTS BETWEEN LABELS
pdel_elem_range	DELETE ELEMENT RANGE
pdel_store	DELETE STORE
pdel_struct	DELETE STRUCTURE
pdel_structs_ar	DELETE STRUCTURES FROM ARCHIVE
pdel_struct_net	DELETE STRUCTURE NETWORK
pdel_struct_nets_ar	DELETE STRUCTURE NETWORKS FROM ARCHIVE
pelem_search	ELEMENT SEARCH
pemergency_close_phigs	EMERGENCY CLOSE PHIGS
pempty_struct	EMPTY STRUCTURE
perr_hand	ERROR HANDLING
perr_log	ERROR LOGGING
pescape	ESCAPE
peval_view_map_matrix	EVALUATE VIEW MAPPING MATRIX
peval_view_map_matrix3	EVALUATE VIEW MAPPING MATRIX 3
peval_view_ori_matrix	EVALUATE VIEW ORIENTATION MATRIX

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
peval_view_ori_matrix3	EVALUATE VIEW ORIENTATION MATRIX 3
pexec_struct	EXECUTE STRUCTURE
pfill_area	FILL AREA
pfill_area3	FILL AREA 3
pfill_area_set	FILL AREA SET
pfill_area_set3	FILL AREA SET 3
pflush_events	FLUSH DEVICE EVENTS
pgdp	GENERALIZED DRAWING PRIMITIVE
pgdp3	GENERALIZED DRAWING PRIMITIVE 3
pget_choice	GET CHOICE
pget_item_type	GET ITEM TYPE FROM METAFILE
pget_loc	GET LOCATOR
pget_loc3	GET LOCATOR 3
pget_pick	GET PICK
pget_string	GET STRING
pget_stroke	GET STROKE
pget_stroke3	GET STROKE 3
pget_val	GET VALUATOR
pgse	GENERALIZED STRUCTURE ELEMENT
pincr_spa_search	INCREMENTAL SPATIAL SEARCH
pincr_spa_search3	INCREMENTAL SPATIAL SEARCH 3
pinit_choice	INITIALIZE CHOICE
pinit_choice3	INITIALIZE CHOICE 3
pinit_loc	INITIALIZE LOCATOR
pinit_loc3	INITIALIZE LOCATOR 3
pinit_pick	INITIALIZE PICK
pinit_pick3	INITIALIZE PICK 3
pinit_string	INITIALIZE STRING
pinit_string3	INITIALIZE STRING 3
pinit_stroke	INITIALIZE STROKE
pinit_stroke3	INITIALIZE STROKE 3
pinit_val	INITIALIZE VALUATOR
pinit_val3	INITIALIZE VALUATOR 3
pinq_all_conf_structs	INQUIRE ALL CONFLICTING STRUCTURES
pinq_anno_facs	INQUIRE ANNOTATION FACILITIES
pinq_ar_files	INQUIRE ARCHIVE FILES
pinq_ar_st	INQUIRE ARCHIVE STATE VALUE
pinq_choice_st	INQUIRE CHOICE DEVICE STATE
pinq_choice_st3	INQUIRE CHOICE DEVICE STATE 3
pinq_colr_facs	INQUIRE COLOUR FACILITIES
pinq_colr_model	INQUIRE COLOUR MODEL
pinq_colr_model_facs	INQUIRE COLOUR MODEL FACILITIES
pinq_colr_rep	INQUIRE COLOUR REPRESENTATION
pinq_conf_res	INQUIRE CONFLICT RESOLUTION
pinq_conf_structs_net	INQUIRE CONFLICTING STRUCTURES IN NETWORK
pinq_cur_elem_content	INQUIRE CURRENT ELEMENT CONTENT
pinq_cur_elem_type_size	INQUIRE CURRENT ELEMENT TYPE AND SIZE
pinq_def_choice_data	INQUIRE DEFAULT CHOICE DEVICE DATA
pinq_def_choice_data3	INQUIRE DEFAULT CHOICE DEVICE DATA 3
pinq_def_disp_upd_st	INQUIRE DEFAULT DISPLAY UPDATE STATE
pinq_def_loc_data	INQUIRE DEFAULT LOCATOR DEVICE DATA

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
pinq_def_loc_data3	INQUIRE DEFAULT LOCATOR DEVICE DATA 3
pinq_def_pick_data	INQUIRE DEFAULT PICK DEVICE DATA
pinq_def_pick_data3	INQUIRE DEFAULT PICK DEVICE DATA 3
pinq_def_string_data	INQUIRE DEFAULT STRING DEVICE DATA
pinq_def_string_data3	INQUIRE DEFAULT STRING DEVICE DATA 3
pinq_def_stroke_data	INQUIRE DEFAULT STROKE DEVICE DATA
pinq_def_stroke_data3	INQUIRE DEFAULT STROKE DEVICE DATA 3
pinq_def_val_data	INQUIRE DEFAULT VALUATOR DEVICE DATA
pinq_def_val_data3	INQUIRE DEFAULT VALUATOR DEVICE DATA 3
pinq_disp_space_size	INQUIRE DISPLAY SPACE SIZE
pinq_disp_space_size3	INQUIRE DISPLAY SPACE SIZE 3
pinq_disp_upd_st	INQUIRE DISPLAY UPDATE STATE
pinq_dyns_structs	INQUIRE DYNAMICS OF STRUCTURES
pinq_dyns_ws_attrs	INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES
pinq_edge_facs	INQUIRE EDGE FACILITIES
pinq_edge_rep	INQUIRE EDGE REPRESENTATION
pinq_edit_mode	INQUIRE EDIT MODE
pinq_elem_content	INQUIRE ELEMENT CONTENT
pinq_elem_ptr	INQUIRE ELEMENT POINTER
pinq_elem_type_size	INQUIRE ELEMENT TYPE AND SIZE
pinq_err_hand_mode	INQUIRE ERROR HANDLING MODE
pinq_gdp	INQUIRE GENERALIZED DRAWING PRIMITIVE
pinq_gdp3	INQUIRE GENERALIZED DRAWING PRIMITIVE 3
pinq_gse_facs	INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES
pinq_highl_filter	INQUIRE HIGHLIGHTING FILTER
pinq_hlhsr_id_facs	INQUIRE HLHSR IDENTIFIER FACILITIES
pinq_hlhsr_mode	INQUIRE HLHSR MODE
pinq_hlhsr_mode_facs	INQUIRE HLHSR MODE FACILITIES
pinq_int_facs	INQUIRE INTERIOR FACILITIES
pinq_int_rep	INQUIRE INTERIOR REPRESENTATION
pinq_invis_filter	INQUIRE INVISIBILITY FILTER
pinq_in_overf	INQUIRE INPUT QUEUE OVERFLOW
pinq_line_facs	INQUIRE POLYLINE FACILITIES
pinq_line_rep	INQUIRE POLYLINE REPRESENTATION
pinq_list_avail_gdp	INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES
pinq_list_avail_gdp3	INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3
pinq_list_avail_gse	INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS
pinq_list_avail_ws_types	INQUIRE LIST OF AVAILABLE WORKSTATION TYPES
pinq_list_colr_inds	INQUIRE LIST OF COLOUR INDICES
pinq_list_edge_inds	INQUIRE LIST OF EDGE INDICES
pinq_list_int_inds	INQUIRE LIST OF INTERIOR INDICES
pinq_list_line_inds	INQUIRE LIST OF POLYLINE INDICES
pinq_list_marker_inds	INQUIRE LIST OF POLYMARKER INDICES
pinq_list_pat_inds	INQUIRE LIST OF PATTERN INDICES
pinq_list_text_inds	INQUIRE LIST OF TEXT INDICES
pinq_list_view_inds	INQUIRE LIST OF VIEW INDICES
pinq_loc_st	INQUIRE LOCATOR DEVICE STATE
pinq_loc_st3	INQUIRE LOCATOR DEVICE STATE 3
pinq_marker_facs	INQUIRE POLYMARKER FACILITIES
pinq_marker_rep	INQUIRE POLYMARKER REPRESENTATION
pinq_model_clip_facs	INQUIRE MODELLING CLIPPING FACILITIES

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
pinq_more_simult_events	INQUIRE MORE SIMULTANEOUS EVENTS
pinq_num_avail_in	INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES
pinq_num_disp_pris	INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED
pinq_open_struct	INQUIRE OPEN STRUCTURE
pinq_open_wss	INQUIRE SET OF OPEN WORKSTATIONS
pinq_paths_ances	INQUIRE PATHS TO ANCESTORS
pinq_paths_descs	INQUIRE PATHS TO DESCENDANTS
pinq_pat_facs	INQUIRE PATTERN FACILITIES
pinq_pat_rep	INQUIRE PATTERN REPRESENTATION
pinq_phigs_facs	INQUIRE PHIGS FACILITIES
pinq_pick_st	INQUIRE PICK DEVICE STATE
pinq_pick_st3	INQUIRE PICK DEVICE STATE 3
pinq_posted_structs	INQUIRE POSTED STRUCTURES
pinq_pred_colr_rep	INQUIRE PREDEFINED COLOUR REPRESENTATION
pinq_pred_edge_rep	INQUIRE PREDEFINED EDGE REPRESENTATION
pinq_pred_int_rep	INQUIRE PREDEFINED INTERIOR REPRESENTATION
pinq_pred_line_rep	INQUIRE PREDEFINED POLYLINE REPRESENTATION
pinq_pred_marker_rep	INQUIRE PREDEFINED POLYMARKER REPRESENTATION
pinq_pred_pat_rep	INQUIRE PREDEFINED PATTERN REPRESENTATION
pinq_pred_text_rep	INQUIRE PREDEFINED TEXT REPRESENTATION
pinq_pred_view_rep	INQUIRE PREDEFINED VIEW REPRESENTATION
pinq_string_st	INQUIRE STRING DEVICE STATE
pinq_string_st3	INQUIRE STRING DEVICE STATE 3
pinq_stroke_st	INQUIRE STROKE DEVICE STATE
pinq_stroke_st3	INQUIRE STROKE DEVICE STATE 3
pinq_struct_ids	INQUIRE STRUCTURE IDENTIFIERS
pinq_struct_st	INQUIRE STRUCTURE STATE VALUE
pinq_struct_status	INQUIRE STRUCTURE STATUS
pinq_sys_st	INQUIRE SYSTEM STATE VALUE
pinq_text_extent	INQUIRE TEXT EXTENT
pinq_text_facs	INQUIRE TEXT FACILITIES
pinq_text_rep	INQUIRE TEXT REPRESENTATION
pinq_val_st	INQUIRE VALUATOR DEVICE STATE
pinq_val_st3	INQUIRE VALUATOR DEVICE STATE 3
pinq_view_facs	INQUIRE VIEW FACILITIES
pinq_view_rep	INQUIRE VIEW REPRESENTATION
pinq_wss_posted	INQUIRE SET OF WORKSTATIONS TO WHICH POSTED
pinq_ws_cat	INQUIRE WORKSTATION CATEGORY
pinq_ws_class	INQUIRE WORKSTATION CLASSIFICATION
pinq_ws_conn_type	INQUIRE WORKSTATION CONNECTION AND TYPE
pinq_ws_st	INQUIRE WORKSTATION STATE VALUE
pinq_ws_st_table	INQUIRE WORKSTATION STATE TABLE LENGTHS
pinq_ws_tran	INQUIRE WORKSTATION TRANSFORMATION
pinq_ws_tran3	INQUIRE WORKSTATION TRANSFORMATION 3
pinterpret_item	INTERPRET ITEM
plabel	LABEL
pmessage	MESSAGE
poffset_elem_ptr	OFFSET ELEMENT POINTER
popen_ar_file	OPEN ARCHIVE FILE
popen_phigs	OPEN PHIGS
popen_struct	OPEN STRUCTURE

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
popen_ws	OPEN WORKSTATION
ppolyline	POLYLINE
ppolyline3	POLYLINE 3
ppolymarker	POLYMARKER
ppolymarker3	POLYMARKER 3
ppost_struct	POST STRUCTURE
pread_item	READ ITEM FROM METAFILE
predraw_all_structs	REDRAW ALL STRUCTURES
premove_names_set	REMOVE NAMES FROM SET
preq_choice	REQUEST CHOICE
preq_loc	REQUEST LOCATOR
preq_loc3	REQUEST LOCATOR 3
preq_pick	REQUEST PICK
preq_string	REQUEST STRING
preq_stroke	REQUEST STROKE
preq_stroke3	REQUEST STROKE 3
preq_val	REQUEST VALUATOR
prestore_model_clip_vol	RESTORE MODELLING CLIPPING VOLUME
pret_all_structs	RETRIEVE ALL STRUCTURES
pret_paths_ances	RETRIEVE PATHS TO ANCESTORS
pret_paths_descs	RETRIEVE PATHS TO DESCENDANTS
pret_structs	RETRIEVE STRUCTURES
pret_struct_ids	RETRIEVE STRUCTURE IDENTIFIERS
pret_struct_nets	RETRIEVE STRUCTURE NETWORKS
protate	ROTATE
protate_x	ROTATE X
protate_y	ROTATE Y
protate_z	ROTATE Z
psample_choice	SAMPLE CHOICE
psample_loc	SAMPLE LOCATOR
psample_loc3	SAMPLE LOCATOR 3
psample_pick	SAMPLE PICK
psample_string	SAMPLE STRING
psample_stroke	SAMPLE STROKE
psample_stroke3	SAMPLE STROKE 3
psample_val	SAMPLE VALUATOR
pscale	SCALE
pscale3	SCALE 3
pset_anno_align	SET ANNOTATION TEXT ALIGNMENT
pset_anno_char_ht	SET ANNOTATION TEXT CHARACTER HEIGHT
pset_anno_char_up_vec	SET ANNOTATION TEXT CHARACTER UP VECTOR
pset_anno_path	SET ANNOTATION TEXT PATH
pset_anno_style	SET ANNOTATION STYLE
pset_char_expan	SET CHARACTER EXPANSION FACTOR
pset_char_ht	SET CHARACTER HEIGHT
pset_char_space	SET CHARACTER SPACING
pset_char_up_vec	SET CHARACTER UP VECTOR
pset_choice_mode	SET CHOICE MODE
pset_colr_model	SET COLOUR MODEL
pset_colr_rep	SET COLOUR REPRESENTATION
pset_conf_res	SET CONFLICT RESOLUTION

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
pset_disp_upd_st	SET DISPLAY UPDATE STATE
pset_edgetype	SET EDGETYPE
pset_edgewidth	SET EDGEWIDTH SCALE FACTOR
pset_edge_colr_ind	SET EDGE COLOUR INDEX
pset_edge_flag	SET EDGE FLAG
pset_edge_ind	SET EDGE INDEX
pset_edge_rep	SET EDGE REPRESENTATION
pset_edit_mode	SET EDIT MODE
pset_elem_ptr	SET ELEMENT POINTER
pset_elem_ptr_label	SET ELEMENT POINTER AT LABEL
pset_err_hand	SET ERROR HANDLING
pset_err_hand_mode	SET ERROR HANDLING MODE
pset_global_tran	SET GLOBAL TRANSFORMATION
pset_global_tran3	SET GLOBAL TRANSFORMATION 3
pset_highl_filter	SET HIGHLIGHTING FILTER
pset_hlhsr_id	SET HLHSR IDENTIFIER
pset_hlhsr_mode	SET HLHSR MODE
pset_indiv_asf	SET INDIVIDUAL ASF
pset_int_colr_ind	SET INTERIOR COLOUR INDEX
pset_int_ind	SET INTERIOR INDEX
pset_int_rep	SET INTERIOR REPRESENTATION
pset_int_style	SET INTERIOR STYLE
pset_int_style_ind	SET INTERIOR STYLE INDEX
pset_invis_filter	SET INVISIBILITY FILTER
pset_linetype	SET LINETYPE
pset_linewidth	SET LINEWIDTH SCALE FACTOR
pset_line_colr_ind	SET POLYLINE COLOUR INDEX
pset_line_ind	SET POLYLINE INDEX
pset_line_rep	SET POLYLINE REPRESENTATION
pset_local_tran	SET LOCAL TRANSFORMATION
pset_local_tran3	SET LOCAL TRANSFORMATION 3
pset_loc_mode	SET LOCATOR MODE
pset_marker_colr_ind	SET POLYMARKER COLOUR INDEX
pset_marker_ind	SET POLYMARKER INDEX
pset_marker_rep	SET POLYMARKER REPRESENTATION
pset_marker_size	SET MARKER SIZE SCALE FACTOR
pset_marker_type	SET MARKER TYPE
pset_model_clip_ind	SET MODELLING CLIPPING INDICATOR
pset_model_clip_vol	SET MODELLING CLIPPING VOLUME
pset_model_clip_vol3	SET MODELLING CLIPPING VOLUME 3
pset_pat_ref_point	SET PATTERN REFERENCE POINT
pset_pat_ref_point_vecs	SET PATTERN REFERENCE POINT AND VECTORS
pset_pat_rep	SET PATTERN REPRESENTATION
pset_pat_size	SET PATTERN SIZE
pset_pick_filter	SET PICK FILTER
pset_pick_id	SET PICK IDENTIFIER
pset_pick_mode	SET PICK MODE
pset_string_mode	SET STRING MODE
pset_stroke_mode	SET STROKE MODE
pset_text_align	SET TEXT ALIGNMENT
pset_text_colr_ind	SET TEXT COLOUR INDEX

Table 2 - Function names ordered by bound name

C Name	PHIGS Name
pset_text_font	SET TEXT FONT
pset_text_ind	SET TEXT INDEX
pset_text_path	SET TEXT PATH
pset_text_prec	SET TEXT PRECISION
pset_text_rep	SET TEXT REPRESENTATION
pset_val_mode	SET VALUATOR MODE
pset_view_ind	SET VIEW INDEX
pset_view_rep	SET VIEW REPRESENTATION
pset_view_rep3	SET VIEW REPRESENTATION 3
pset_view_tran_in_pri	SET VIEW TRANSFORMATION INPUT PRIORITY
pset_ws_vp	SET WORKSTATION VIEWPORT
pset_ws_vp3	SET WORKSTATION VIEWPORT 3
pset_ws_win	SET WORKSTATION WINDOW
pset_ws_win3	SET WORKSTATION WINDOW 3
ptext	TEXT
ptext3	TEXT 3
ptranslate	TRANSLATE
ptranslate3	TRANSLATE 3
ptran_point	TRANSFORM POINT
ptran_point3	TRANSFORM POINT 3
punpost_all_structs	UNPOST ALL STRUCTURES
punpost_struct	UNPOST STRUCTURE
pupd_ws	UPDATE WORKSTATION
pwrite_item	WRITE ITEM TO METAFILE

4.3.2 List ordered alphabetically by PHIGS function name

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
ADD NAMES TO SET	padd_names_set
ANNOTATION TEXT RELATIVE	panno_text_rel
ANNOTATION TEXT RELATIVE 3	panno_text_rel3
APPLICATION DATA	pappl_data
ARCHIVE ALL STRUCTURES	par_all_structs
ARCHIVE STRUCTURE NETWORKS	par_struct_nets
ARCHIVE STRUCTURES	par_structs
AWAIT EVENT	pawait_event
BUILD TRANSFORMATION MATRIX	pbuild_tran_matrix
BUILD TRANSFORMATION MATRIX 3	pbuild_tran_matrix3
CELL ARRAY	pcell_array
CELL ARRAY 3	pcell_array3
CHANGE STRUCTURE IDENTIFIER	pchange_struct_id
CHANGE STRUCTURE IDENTIFIER AND REFERENCES	pchange_struct_id_refs
CHANGE STRUCTURE REFERENCES	pchange_struct_refs
CLOSE ARCHIVE FILE	pclose_ar_file
CLOSE PHIGS	pclose_phigs
CLOSE STRUCTURE	pclose_struct
CLOSE WORKSTATION	pclose_ws

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
COMPOSE MATRIX	pcompose_matrix
COMPOSE MATRIX 3	pcompose_matrix3
COMPOSE TRANSFORMATION MATRIX	pcompose_tran_matrix
COMPOSE TRANSFORMATION MATRIX 3	pcompose_tran_matrix3
COPY ALL ELEMENTS FROM STRUCTURE	pcopy_all_elems_struct
CREATE STORE	pcreate_store
DELETE ALL STRUCTURES	pdel_all_structs
DELETE ALL STRUCTURES FROM ARCHIVE	pdel_all_structs_ar
DELETE ELEMENT	pdel_elem
DELETE ELEMENT RANGE	pdel_elem_range
DELETE ELEMENTS BETWEEN LABELS	pdel_elems_labels
DELETE STORE	pdel_store
DELETE STRUCTURE	pdel_struct
DELETE STRUCTURE NETWORK	pdel_struct_net
DELETE STRUCTURE NETWORKS FROM ARCHIVE	pdel_struct_nets_ar
DELETE STRUCTURES FROM ARCHIVE	pdel_structs_ar
ELEMENT SEARCH	pelem_search
EMERGENCY CLOSE PHIGS	pemergency_close_phigs
EMPTY STRUCTURE	pempty_struct
ERROR HANDLING	perr_hand
ERROR LOGGING	perr_log
ESCAPE	pescape
EVALUATE VIEW MAPPING MATRIX	peval_view_map_matrix
EVALUATE VIEW MAPPING MATRIX 3	peval_view_map_matrix3
EVALUATE VIEW ORIENTATION MATRIX	peval_view_ori_matrix
EVALUATE VIEW ORIENTATION MATRIX 3	peval_view_ori_matrix3
EXECUTE STRUCTURE	pexec_struct
FILL AREA	pfill_area
FILL AREA 3	pfill_area3
FILL AREA SET	pfill_area_set
FILL AREA SET 3	pfill_area_set3
FLUSH DEVICE EVENTS	pflush_events
GENERALIZED DRAWING PRIMITIVE	pgdp
GENERALIZED DRAWING PRIMITIVE 3	pgdp3
GENERALIZED STRUCTURE ELEMENT	pgse
GET CHOICE	pget_choice
GET ITEM TYPE FROM METAFILE	pget_item_type
GET LOCATOR	pget_loc
GET LOCATOR 3	pget_loc3
GET PICK	pget_pick
GET STRING	pget_string
GET STROKE	pget_stroke
GET STROKE 3	pget_stroke3
GET VALUATOR	pget_val
INCREMENTAL SPATIAL SEARCH	pincr_spa_search
INCREMENTAL SPATIAL SEARCH 3	pincr_spa_search3
INITIALIZE CHOICE	pinit_choice
INITIALIZE CHOICE 3	pinit_choice3
INITIALIZE LOCATOR	pinit_loc
INITIALIZE LOCATOR 3	pinit_loc3
INITIALIZE PICK	pinit_pick

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
INITIALIZE PICK 3	pinit_pick3
INITIALIZE STRING	pinit_string
INITIALIZE STRING 3	pinit_string3
INITIALIZE STROKE	pinit_stroke
INITIALIZE STROKE 3	pinit_stroke3
INITIALIZE VALUATOR	pinit_val
INITIALIZE VALUATOR 3	pinit_val3
INQUIRE ALL CONFLICTING STRUCTURES	pinq_all_conf_structs
INQUIRE ANNOTATION FACILITIES	pinq_anno_facs
INQUIRE ARCHIVE FILES	pinq_ar_files
INQUIRE ARCHIVE STATE VALUE	pinq_ar_st
INQUIRE CHOICE DEVICE STATE	pinq_choice_st
INQUIRE CHOICE DEVICE STATE 3	pinq_choice_st3
INQUIRE COLOUR FACILITIES	pinq_colr_facs
INQUIRE COLOUR MODEL	pinq_colr_model
INQUIRE COLOUR MODEL FACILITIES	pinq_colr_model_facs
INQUIRE COLOUR REPRESENTATION	pinq_colr_rep
INQUIRE CONFLICT RESOLUTION	pinq_conf_res
INQUIRE CONFLICTING STRUCTURES IN NETWORK	pinq_conf_structs_net
INQUIRE CURRENT ELEMENT CONTENT	pinq_cur_elem_content
INQUIRE CURRENT ELEMENT TYPE AND SIZE	pinq_cur_elem_type_size
INQUIRE DEFAULT CHOICE DEVICE DATA	pinq_def_choice_data
INQUIRE DEFAULT CHOICE DEVICE DATA 3	pinq_def_choice_data3
INQUIRE DEFAULT DISPLAY UPDATE STATE	pinq_def_disp_upd_st
INQUIRE DEFAULT LOCATOR DEVICE DATA	pinq_def_loc_data
INQUIRE DEFAULT LOCATOR DEVICE DATA 3	pinq_def_loc_data3
INQUIRE DEFAULT PICK DEVICE DATA	pinq_def_pick_data
INQUIRE DEFAULT PICK DEVICE DATA 3	pinq_def_pick_data3
INQUIRE DEFAULT STRING DEVICE DATA	pinq_def_string_data
INQUIRE DEFAULT STRING DEVICE DATA 3	pinq_def_string_data3
INQUIRE DEFAULT STROKE DEVICE DATA	pinq_def_stroke_data
INQUIRE DEFAULT STROKE DEVICE DATA 3	pinq_def_stroke_data3
INQUIRE DEFAULT VALUATOR DEVICE DATA	pinq_def_val_data
INQUIRE DEFAULT VALUATOR DEVICE DATA 3	pinq_def_val_data3
INQUIRE DISPLAY SPACE SIZE	pinq_disp_space_size
INQUIRE DISPLAY SPACE SIZE 3	pinq_disp_space_size3
INQUIRE DISPLAY UPDATE STATE	pinq_disp_upd_st
INQUIRE DYNAMICS OF STRUCTURES	pinq_dyns_structs
INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES	pinq_dyns_ws_attrs
INQUIRE EDGE FACILITIES	pinq_edge_facs
INQUIRE EDGE REPRESENTATION	pinq_edge_rep
INQUIRE EDIT MODE	pinq_edit_mode
INQUIRE ELEMENT CONTENT	pinq_elem_content
INQUIRE ELEMENT POINTER	pinq_elem_ptr
INQUIRE ELEMENT TYPE AND SIZE	pinq_elem_type_size
INQUIRE ERROR HANDLING MODE	pinq_err_hand_mode
INQUIRE GENERALIZED DRAWING PRIMITIVE	pinq_gdp
INQUIRE GENERALIZED DRAWING PRIMITIVE 3	pinq_gdp3
INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES	pinq_gse_facs
INQUIRE HIGHLIGHTING FILTER	pinq_highl_filter
INQUIRE HLHSR IDENTIFIER FACILITIES	pinq_hlhsr_id_facs

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
INQUIRE HLHSR MODE	pinq_hlhsr_mode
INQUIRE HLHSR MODE FACILITIES	pinq_hlhsr_mode_facs
INQUIRE INPUT QUEUE OVERFLOW	pinq_in_overf
INQUIRE INTERIOR FACILITIES	pinq_int_facs
INQUIRE INTERIOR REPRESENTATION	pinq_int_rep
INQUIRE INVISIBILITY FILTER	pinq_invis_filter
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	pinq_list_avail_gdp
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3	pinq_list_avail_gdp3
INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS	pinq_list_avail_gse
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	pinq_list_avail_ws_types
INQUIRE LIST OF COLOUR INDICES	pinq_list_colr_inds
INQUIRE LIST OF EDGE INDICES	pinq_list_edge_inds
INQUIRE LIST OF INTERIOR INDICES	pinq_list_int_inds
INQUIRE LIST OF PATTERN INDICES	pinq_list_pat_inds
INQUIRE LIST OF POLYLINE INDICES	pinq_list_line_inds
INQUIRE LIST OF POLYMARKER INDICES	pinq_list_marker_inds
INQUIRE LIST OF TEXT INDICES	pinq_list_text_inds
INQUIRE LIST OF VIEW INDICES	pinq_list_view_inds
INQUIRE LOCATOR DEVICE STATE	pinq_loc_st
INQUIRE LOCATOR DEVICE STATE 3	pinq_loc_st3
INQUIRE MODELLING CLIPPING FACILITIES	pinq_model_clip_facs
INQUIRE MORE SIMULTANEOUS EVENTS	pinq_more_simult_events
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	pinq_num_avail_in
INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED	pinq_num_disp_pris
INQUIRE OPEN STRUCTURE	pinq_open_struct
INQUIRE PATHS TO ANCESTORS	pinq_paths_ances
INQUIRE PATHS TO DESCENDANTS	pinq_paths_descs
INQUIRE PATTERN FACILITIES	pinq_pat_facs
INQUIRE PATTERN REPRESENTATION	pinq_pat_rep
INQUIRE PHIGS FACILITIES	pinq_phigs_facs
INQUIRE PICK DEVICE STATE	pinq_pick_st
INQUIRE PICK DEVICE STATE 3	pinq_pick_st3
INQUIRE POLYLINE FACILITIES	pinq_line_facs
INQUIRE POLYLINE REPRESENTATION	pinq_line_rep
INQUIRE POLYMARKER FACILITIES	pinq_marker_facs
INQUIRE POLYMARKER REPRESENTATION	pinq_marker_rep
INQUIRE POSTED STRUCTURES	pinq_posted_structs
INQUIRE PREDEFINED COLOUR REPRESENTATION	pinq_pred_colr_rep
INQUIRE PREDEFINED EDGE REPRESENTATION	pinq_pred_edge_rep
INQUIRE PREDEFINED INTERIOR REPRESENTATION	pinq_pred_int_rep
INQUIRE PREDEFINED PATTERN REPRESENTATION	pinq_pred_pat_rep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	pinq_pred_line_rep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	pinq_pred_marker_rep
INQUIRE PREDEFINED TEXT REPRESENTATION	pinq_pred_text_rep
INQUIRE PREDEFINED VIEW REPRESENTATION	pinq_pred_view_rep
INQUIRE SET OF OPEN WORKSTATIONS	pinq_open_wss
INQUIRE SET OF WORKSTATIONS TO WHICH POSTED	pinq_wss_posted
INQUIRE STRING DEVICE STATE	pinq_string_st
INQUIRE STRING DEVICE STATE 3	pinq_string_st3
INQUIRE STROKE DEVICE STATE	pinq_stroke_st
INQUIRE STROKE DEVICE STATE 3	pinq_stroke_st3

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
INQUIRE STRUCTURE IDENTIFIERS	pinq_struct_ids
INQUIRE STRUCTURE STATE VALUE	pinq_struct_st
INQUIRE STRUCTURE STATUS	pinq_struct_status
INQUIRE SYSTEM STATE VALUE	pinq_sys_st
INQUIRE TEXT EXTENT	pinq_text_extent
INQUIRE TEXT FACILITIES	pinq_text_facs
INQUIRE TEXT REPRESENTATION	pinq_text_rep
INQUIRE VALUATOR DEVICE STATE	pinq_val_st
INQUIRE VALUATOR DEVICE STATE 3	pinq_val_st3
INQUIRE VIEW FACILITIES	pinq_view_facs
INQUIRE VIEW REPRESENTATION	pinq_view_rep
INQUIRE WORKSTATION CATEGORY	pinq_ws_cat
INQUIRE WORKSTATION CLASSIFICATION	pinq_ws_class
INQUIRE WORKSTATION CONNECTION AND TYPE	pinq_ws_conn_type
INQUIRE WORKSTATION STATE TABLE LENGTHS	pinq_ws_st_table
INQUIRE WORKSTATION STATE VALUE	pinq_ws_st
INQUIRE WORKSTATION TRANSFORMATION	pinq_ws_tran
INQUIRE WORKSTATION TRANSFORMATION 3	pinq_ws_tran3
INTERPRET ITEM	pinterpret_item
LABEL	plabel
MESSAGE	pmessage
OFFSET ELEMENT POINTER	poffset_elem_ptr
OPEN ARCHIVE FILE	popen_ar_file
OPEN PHIGS	popen_phigs
OPEN STRUCTURE	popen_struct
OPEN WORKSTATION	popen_ws
POLYLINE	ppolyline
POLYLINE 3	ppolyline3
POLYMARKER	ppolymarker
POLYMARKER 3	ppolymarker3
POST STRUCTURE	ppost_struct
READ ITEM FROM METAFILE	pread_item
REDRAW ALL STRUCTURES	predraw_all_structs
REMOVE NAMES FROM SET	premove_names_set
REQUEST CHOICE	preq_choice
REQUEST LOCATOR	preq_loc
REQUEST LOCATOR 3	preq_loc3
REQUEST PICK	preq_pick
REQUEST STRING	preq_string
REQUEST STROKE	preq_stroke
REQUEST STROKE 3	preq_stroke3
REQUEST VALUATOR	preq_val
RESTORE MODELLING CLIPPING VOLUME	prestore_model_clip_vol
RETRIEVE ALL STRUCTURES	pret_all_structs
RETRIEVE PATHS TO ANCESTORS	pret_paths_ances
RETRIEVE PATHS TO DESCENDANTS	pret_paths_descs
RETRIEVE STRUCTURE IDENTIFIERS	pret_struct_ids
RETRIEVE STRUCTURE NETWORKS	pret_struct_nets
RETRIEVE STRUCTURES	pret_structs
ROTATE	protate
ROTATE X	protate_x

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
ROTATE Y	protate_y
ROTATE Z	protate_z
SAMPLE CHOICE	psample_choice
SAMPLE LOCATOR	psample_loc
SAMPLE LOCATOR 3	psample_loc3
SAMPLE PICK	psample_pick
SAMPLE STRING	psample_string
SAMPLE STROKE	psample_stroke
SAMPLE STROKE 3	psample_stroke3
SAMPLE VALUATOR	psample_val
SCALE	pscale
SCALE 3	pscale3
SET ANNOTATION STYLE	pset_anno_style
SET ANNOTATION TEXT ALIGNMENT	pset_anno_align
SET ANNOTATION TEXT CHARACTER HEIGHT	pset_anno_char_ht
SET ANNOTATION TEXT CHARACTER UP VECTOR	pset_anno_char_up_vec
SET ANNOTATION TEXT PATH	pset_anno_path
SET CHARACTER EXPANSION FACTOR	pset_char_expan
SET CHARACTER HEIGHT	pset_char_ht
SET CHARACTER SPACING	pset_char_space
SET CHARACTER UP VECTOR	pset_char_up_vec
SET CHOICE MODE	pset_choice_mode
SET COLOUR MODEL	pset_colr_model
SET COLOUR REPRESENTATION	pset_colr_rep
SET CONFLICT RESOLUTION	pset_conf_res
SET DISPLAY UPDATE STATE	pset_disp_upd_st
SET EDGE COLOUR INDEX	pset_edge_colr_ind
SET EDGE FLAG	pset_edge_flag
SET EDGE INDEX	pset_edge_ind
SET EDGE REPRESENTATION	pset_edge_rep
SET EDGETYPE	pset_edgetype
SET EDGEWIDTH SCALE FACTOR	pset_edgewidth
SET EDIT MODE	pset_edit_mode
SET ELEMENT POINTER	pset_elem_ptr
SET ELEMENT POINTER AT LABEL	pset_elem_ptr_label
SET ERROR HANDLING	pset_err_hand
SET ERROR HANDLING MODE	pset_err_hand_mode
SET GLOBAL TRANSFORMATION	pset_global_tran
SET GLOBAL TRANSFORMATION 3	pset_global_tran3
SET HIGHLIGHTING FILTER	pset_highl_filter
SET HLHSR IDENTIFIER	pset_hlhsr_id
SET HLHSR MODE	pset_hlhsr_mode
SET INDIVIDUAL ASF	pset_indiv_asf
SET INTERIOR COLOUR INDEX	pset_int_colr_ind
SET INTERIOR INDEX	pset_int_ind
SET INTERIOR REPRESENTATION	pset_int_rep
SET INTERIOR STYLE	pset_int_style
SET INTERIOR STYLE INDEX	pset_int_style_ind
SET INVISIBILITY FILTER	pset_invis_filter
SET LINETYPE	pset_linetype
SET LINEWIDTH SCALE FACTOR	pset_linewidth

Table 3 - Function names ordered by PHIGS name

PHIGS Name	C Name
SET LOCAL TRANSFORMATION	pset_local_tran
SET LOCAL TRANSFORMATION 3	pset_local_tran3
SET LOCATOR MODE	pset_loc_mode
SET MARKER SIZE SCALE FACTOR	pset_marker_size
SET MARKER TYPE	pset_marker_type
SET MODELLING CLIPPING INDICATOR	pset_model_clip_ind
SET MODELLING CLIPPING VOLUME	pset_model_clip_vol
SET MODELLING CLIPPING VOLUME 3	pset_model_clip_vol3
SET PATTERN REFERENCE POINT	pset_pat_ref_point
SET PATTERN REFERENCE POINT AND VECTORS	pset_pat_ref_point_vecs
SET PATTERN REPRESENTATION	pset_pat_rep
SET PATTERN SIZE	pset_pat_size
SET PICK FILTER	pset_pick_filter
SET PICK IDENTIFIER	pset_pick_id
SET PICK MODE	pset_pick_mode
SET POLYLINE COLOUR INDEX	pset_line_colr_ind
SET POLYLINE INDEX	pset_line_ind
SET POLYLINE REPRESENTATION	pset_line_rep
SET POLYMARKER COLOUR INDEX	pset_marker_colr_ind
SET POLYMARKER INDEX	pset_marker_ind
SET POLYMARKER REPRESENTATION	pset_marker_rep
SET STRING MODE	pset_string_mode
SET STROKE MODE	pset_stroke_mode
SET TEXT ALIGNMENT	pset_text_align
SET TEXT COLOUR INDEX	pset_text_colr_ind
SET TEXT FONT	pset_text_font
SET TEXT INDEX	pset_text_ind
SET TEXT PATH	pset_text_path
SET TEXT PRECISION	pset_text_prec
SET TEXT REPRESENTATION	pset_text_rep
SET VALUATOR MODE	pset_val_mode
SET VIEW INDEX	pset_view_ind
SET VIEW REPRESENTATION	pset_view_rep
SET VIEW REPRESENTATION 3	pset_view_rep3
SET VIEW TRANSFORMATION INPUT PRIORITY	pset_view_tran_in_pri
SET WORKSTATION VIEWPORT	pset_ws_vp
SET WORKSTATION VIEWPORT 3	pset_ws_vp3
SET WORKSTATION WINDOW	pset_ws_win
SET WORKSTATION WINDOW 3	pset_ws_win3
TEXT	ptext
TEXT 3	ptext3
TRANSFORM POINT	ptran_point
TRANSFORM POINT 3	ptran_point3
TRANSLATE	ptranslate
TRANSLATE 3	ptranslate3
UNPOST ALL STRUCTURES	punpost_all_structs
UNPOST STRUCTURE	punpost_struct
UPDATE WORKSTATION	pupd_ws
WRITE ITEM TO METAFILE	pwrite_item

5 Type definitions

5.1 Mapping of PHIGS data types

ISO/IEC 9592-1 specifies a set of abstract data types. Table 4 gives the mapping from those data types to the data types defined in later this binding.

Table 4 - Mapping of PHIGS data types to C

PHIGS data type		C binding data type
I	integer	`Pint`
R	real	`Pfloat`
S	string	`char *`
P2	2D point	`Ppoint`
P3	3D point	`Ppoint3`
V2	2D vector	`Pvec`
V3	3D vector	`Pvec3`
E	enumeration type	`typedef enum`
NM	classification	`Pint`
FR	filter	`Pfilter`
PP	pick path item	`Ppick_path_elem`
ER	element reference	`Pelem_ref`
HS2	half-space	`Phalf_space`
HS3	half-space	`Phalf_space3`
FP	font/precision pair	not applicable
SE	structure element	not applicable
PS	posted structure	`Pposted_struct`
B	bounding range	2 × B is `Plimit`; 3 × B is `Plimit3`
CLR	colour specification	`Pcolr_rep`
CC	chromaticity coefficient and luminance value	`Pcieluv`
C	connection identifier	`void *`
F	file	`char *`
W	workstation type	`Pint`
MCV	modelling clipping volume	not applicable
G2	2D GDP identifier	`Pint`
G3	3D GDP identifier	`Pint`
GS	GSE identifier	`Pint`
AI	archive file identifier	`Pint`
PI	pick identifier	`Pint`
EI	escape identifier	`Pint`
FN	function name	`Pint`
WI	workstation identifier	`Pint`

5.2 Environmental type definitions

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

`Pfloat` — floating point number

This data type must be defined by the implementation as a type suitable for containing the single precision floating point values used by the PHIGS implementation. Suggest:

```
typedef float   Pfloat;
```

`Pint` — integer

This data type must be defined by the implementation as a type suitable for containing the integer values used by the PHIGS implementation. Suggest:

```
typedef int      Pint;
```

5.3 Implementation dependent type definitions

The following type definitions may be modified by the implementation. If the implementation adds new fields to the unions, this convention shall be followed for naming the new field:

"xxx_tn"

where:

xxx is the name of the field. Use "pet" for input device data record prompt echo types, "gdp" for GENERALIZED DRAWING PRIMITIVE data record, "gdp3" for GENERALIZED DRAWING PRIMITIVE 3 data records, "gse" for GENERALIZED STRUCTURE ELEMENT, or "escape_in" for ESCAPE input data records, "escape_out" for ESCAPE output data records.

t is the type of field. Use "r" for fields corresponding registered items and "u" for fields for unregistered items.

n is "n" is the item number; for unregistered items, the absolute value of the item number shall be used.

The contents of the metafile item data record (`Pitem_data`) are not subject to registration and therefore do not follow this naming convention.

Pchoice_data CHOICE DATA RECORD

```
typedef struct {
  union Pchoice_pets {
    struct Pchoice_pet_r1 {
          . . .                          /* implementation dependent     */
    } pet_r1;
    struct Pchoice_pet_r2 {
          Pint        num_prompts;       /* number of prompts            */
          Ppr_switch  *prompts;          /* array of prompts             */
          . . .                          /* implementation dependent     */
    } pet_r2;
    struct Pchoice_pet_r3 {
          Pint        num_strings;       /* number of choice strings     */
          char        **strings;         /* array of choice strings      */
          . . .                          /* implementation dependent     */
    } pet_r3;
    struct Pchoice_pet_r4 {
          Pint        num_strings;       /* number of choice strings     */
          char        **strings;         /* array of choice strings      */
          . . .                          /* implementation dependent     */
    } pet_r4;
    struct Pchoice_pet_r5 {
          Pint        struct_id;         /* struct identifier            */
          Pint        num_pick_ids;      /* number of pick identifiers   */
          Pint        *pick_ids;         /* array of pick identifiers    */
          . . .                          /* implementation dependent     */
    } pet_r5;
    . . .                                /* implementation defined PET's */
  } pets;
} Pchoice_data;
```

Pchoice_data3 CHOICE DATA RECORD 3

```
typedef struct {
  union Pchoice3_pets {
    struct Pchoice3_pet_r1 {
          . . .                       /* implementation dependent     */
    } pet_r1;
    struct Pchoice3_pet_r2 {
          Pint        num_prompts;    /* number of prompts            */
          Ppr_switch  *prompts;       /* array of prompts             */
          . . .                       /* implementation dependent     */
    } pet_r2;
    struct Pchoice3_pet_r3 {
          Pint        num_strings;    /* number of choice strings     */
          char        **strings;      /* array of choice strings      */
          . . .                       /* implementation dependent     */
    } pet_r3;
    struct Pchoice3_pet_r4 {
          Pint        num_strings;    /* number of choice strings     */
          char        **strings;      /* array of choice strings      */
          . . .                       /* implementation dependent     */
    } pet_r4;
    struct Pchoice3_pet_r5 {
          Pint        struct_id;      /* struct identifier            */
          Pint        num_pick_ids;   /* number of pick identifiers   */
          Pint        *pick_ids;      /* array of pick identifiers    */
          . . .                       /* implementation dependent     */
    } pet_r5;
    . . .                             /* implementation defined PET's */
  } pets;
} Pchoice_data3;
```

Pcolr_rep COLOUR REPRESENTATION

```
typedef union {
  Prgb     rgb;        /* Red Green Blue colour specification           */
  Pcieluv  cieluv;     /* CIE L*u*v* colour specification               */
  Phls     hls;        /* Hue Lightness Saturation colour specification */
  Phsv     hsv;        /* Hue Saturation Value colour specification     */
  Pdata    unsupp;     /* colour in an unsupported colour model         */
  . . .                /* implementation defined                        */
} Pcolr_rep;
```

`Pescape_in_data` ESCAPE INPUT DATA RECORD

```
typedef union {
  struct Pescape_in_r1 {
        . . .       /* registration dependent */
  } escape_in_r1;
  . . .             /* implementation defined */
} Pescape_in_data;
```

`Pescape_out_data` ESCAPE OUTPUT DATA RECORD

```
typedef union {
  struct Pescape_out_r1 {
        . . .       /* registration dependent */
  } escape_out_r1;
  . . .             /* implementation defined */
} Pescape_out_data;
```

`Pgdp_data` GDP DATA RECORD

```
typedef union {
  struct Pgdp_r1 {
        . . .       /* registration dependent      */
  } gdp_r1;
  Pdata  unsupp;    /* unsupported GDP data record */
  . . .             /* implementation defined      */
} Pgdp_data;
```

`Pgdp_data3` GDP DATA RECORD 3

```
typedef union {
  struct Pgdp3_r1 {
        . . .       /* registration dependent       */
  } gdp3_r1;
  Pdata  unsupp;    /* unsupported GDP3 data record */
  . . .             /* implementation defined       */
} Pgdp_data3;
```

`Pgse_data` GSE DATA RECORD

```
typedef union {
  struct Pgse_r1 {
        . . .          /* registration dependent      */
  } gse_r1;
  Pdata  unsupp;       /* unsupported GSE data record */
  . . .                /* implementation defined      */
} Pgse_data;
```

`Pitem_data` ITEM DATA RECORD

```
typedef union {
  . . .                /* Metafile Records                */
  Pdata  unsupp;       /* unsupported Metafile item data */
} Pitem_data;
```

`Ploc_data` LOCATOR DATA RECORD

```
typedef struct {
  union Ploc_pets {
    struct Ploc_pet_r1 {
          . . .                                          /* impl. dependent       */
    } pet_r1;
    struct Ploc_pet_r2 {
          . . .                                          /* impl. dependent       */
    } pet_r2;
    struct Ploc_pet_r3 {
          . . .                                          /* impl. dependent       */
    } pet_r3;
    struct Ploc_pet_r4 {
          Pline_attrs           line_attrs;              /* polyline attributes   */
          . . .                                          /* impl. dependent       */
    } pet_r4;
    struct Ploc_pet_r5 {
          Pline_fill_ctrl_flag  line_fill_ctrl_flag;     /* control flag          */
          union Ploc_attrs {
                                Pline_attrs              line_attrs;
                                                         /* polyline attributes   */
                                Pint_attrs               int_attrs;
                                                         /* interior attributes   */
                                struct Ploc_fill_set {
                                                         Pint_attrs  int_attrs;
                                                         /* interior attributes   */
                                                         Pedge_attrs  edge_attrs;
                                                         /* edge attributes       */
                                } fill_set;
          } attrs;
          . . .                                          /* impl. dependent       */
    } pet_r5;
    struct Ploc_pet_r6 {
          . . .                                          /* impl. dependent       */
    } pet_r6;
    . . .                                                /* impl. defined PET's   */
  } pets;
} Ploc_data;
```

Ploc_data3 LOCATOR DATA RECORD 3

```
typedef struct {
  union Ploc3_pets {
    struct Ploc3_pet_r1 {
          . . .                                          /* impl. dependent       */
    } pet_r1;
    struct Ploc3_pet_r2 {
          . . .                                          /* impl. dependent       */
    } pet_r2;
    struct Ploc3_pet_r3 {
          . . .                                          /* impl. dependent       */
    } pet_r3;
    struct Ploc3_pet_r4 {
          Pline_attrs           line_attrs;              /* polyline attributes   */
          . . .                                          /* impl. dependent       */
    } pet_r4;
    struct Ploc3_pet_r5 {
          Pline_fill_ctrl_flag  line_fill_ctrl_flag;     /* control flag          */
          union Ploc3_attrs {
                                Pline_attrs              line_attrs;
                                                         /* polyline attributes   */
                                Pint_attrs               int_attrs;
                                                         /* interior attributes   */
                                struct Ploc3_fill_set {
                                                         Pint_attrs  int_attrs;
                                                         /* interior attributes   */
                                                         Pedge_attrs  edge_attrs;
                                                         /* edge attributes       */
                                } fill_set;
          } attrs;
          . . .                                          /* impl. dependent       */
    } pet_r5;
    struct Ploc3_pet_r6 {
          . . .                                          /* impl. dependent       */
    } pet_r6;
    . . .                                                /* impl. defined PET's   */
  } pets;
} Ploc_data3;
```

Ppick_data PICK DATA RECORD

```
typedef struct {
  union Ppick_pets {
    struct Ppick_pet_r1 {
          . . .     /* implementation dependent      */
    } pet_r1;
    . . .           /* implementation defined PET's */
  } pets;
} Ppick_data;
```

Ppick_data3 PICK DATA RECORD 3

```
typedef struct {
  union Ppick3_pets {
    struct Ppick3_pet_r1 {
          . . .     /* implementation dependent      */
    } pet_r1;
    . . .           /* implementation defined PET's */
  } pets;
} Ppick_data3;
```

Pstring_data STRING DATA RECORD

```
typedef struct {
  Pint  in_buf_size;          /* input buffer size            */
  Pint  init_pos;             /* initial editing position     */
  union Pstring_pets {
      struct Pstring_pet_r1 {
                  . . .       /* implementation dependent     */
      } pet_r1;
      . . .                   /* implementation defined PET's */
  } pets;
} Pstring_data;
```

`Pstring_data3` STRING DATA RECORD 3

```
typedef struct {
  Pint  in_buf_size;          /* input buffer size            */
  Pint  init_pos;             /* initial editing position     */
  union Pstring3_pets {
        struct Pstring3_pet_r1 {
                    . . .     /* implementation dependent     */
        } pet_r1;
        . . .                 /* implementation defined PET's */
  } pets;
} Pstring_data3;
```

`Pstroke_data` STROKE DATA RECORD

```
typedef struct {
  Pint    in_buf_size;                                  /* input buffer size            *
  Pint    init_pos;                                     /* initial editing position     *
  Pfloat  x_interval;                                   /* x interval                   *
  Pfloat  y_interval;                                   /* y interval                   *
  Pfloat  time_interval;                                /* time interval                *
  union Pstroke_pets {
        struct Pstroke_pet_r1 {
                    . . .                               /* implementation dependent     *
        } pet_r1;
        struct Pstroke_pet_r2 {
                    . . .                               /* implementation dependent     *
        } pet_r2;
        struct Pstroke_pet_r3 {
                    Pmarker_attrs  marker_attrs;        /* marker attributes            *
                    . . .                               /* implementation dependent     *
        } pet_r3;
        struct Pstroke_pet_r4 {
                    Pline_attrs    line_attrs;          /* line attributes              *
                    . . .                               /* implementation dependent     *
        } pet_r4;
        . . .                                           /* implementation defined PET's *
  } pets;
} Pstroke_data;
```

`Pstroke_data3` STROKE DATA RECORD 3

```
typedef struct {
  Pint    in_buf_size;                                  /* input buffer size
  Pint    init_pos;                                     /* initial editing position
  Pfloat  x_interval;                                   /* x interval
  Pfloat  y_interval;                                   /* y interval
  Pfloat  z_interval;                                   /* z interval
  Pfloat  time_interval;                                /* time interval
  union Pstroke3_pets {
        struct Pstroke3_pet_r1 {
                        . . .                           /* implementation dependent
        } pet_r1;
        struct Pstroke3_pet_r2 {
                        . . .                           /* implementation dependent
        } pet_r2;
        struct Pstroke3_pet_r3 {
                        Pmarker_attrs  marker_attrs;    /* marker attributes
                        . . .                           /* implementation dependent
        } pet_r3;
        struct Pstroke3_pet_r4 {
                        Pline_attrs   line_attrs;       /* line attributes
                        . . .                           /* implementation dependent
        } pet_r4;
        . . .                                           /* implementation defined P
  } pets;
} Pstroke_data3;
```

`Pval_data` VALUATOR DATA RECORD

```
typedef struct {
  Pfloat  low_value;          /* low value of valuator range  */
  Pfloat  high_value;         /* high value of valuator range */
  union Pval_pets {
        struct Pval_pet_r1 {
                    . . .     /* implementation dependent     */
        } pet_r1;
        . . .                 /* implementation defined PET's */
  } pets;
} Pval_data;
```

`Pval_data3` VALUATOR DATA RECORD 3

```
typedef struct {
  Pfloat  low_value;            /* low value of valuator range  */
  Pfloat  high_value;           /* high value of valuator range */
  union Pval3_pets {
          struct Pval3_pet_r1 {
                        . . .   /* implementation dependent     */
          } pet_r1;
          . . .                 /* implementation defined PET's */
  } pets;
} Pval_data3;
```

5.4 Implementation independent type definitions

`Par_file` ARCHIVE FILE

```
typedef struct {
  Pint   id;          /* archive file identifier */
  char   *name;       /* archive file name       */
} Par_file;
```

`Par_file_list` ARCHIVE FILE LIST

```
typedef struct {
  Pint      num_ar_files;     /* number of archive files */
  Par_file  *ar_files;        /* list of archive files   */
} Par_file_list;
```

`Par_st` ARCHIVE STATE

```
typedef enum {
    PST_ARCL,
    PST_AROP
} Par_st;
```

Pasf ASPECT SOURCE FLAG

```
typedef enum {
    PASF_BUNDLED,
    PASF_INDIV
} Pasf;
```

Paspect ASPECT

```
typedef enum {
    PASPECT_LINETYPE,
    PASPECT_LINEWIDTH,
    PASPECT_LINE_COLR_IND,
    PASPECT_MARKER_TYPE,
    PASPECT_MARKER_SIZE,
    PASPECT_MARKER_COLR_IND,
    PASPECT_TEXT_FONT,
    PASPECT_TEXT_PREC,
    PASPECT_CHAR_EXPAN,
    PASPECT_CHAR_SPACE,
    PASPECT_TEXT_COLR_IND,
    PASPECT_INT_STYLE,
    PASPECT_INT_STYLE_IND,
    PASPECT_INT_COLR_IND,
    PASPECT_EDGE_FLAG,
    PASPECT_EDGETYPE,
    PASPECT_EDGEWIDTH,
    PASPECT_EDGE_COLR_IND
} Paspect;
```

Pattrs ATTRIBUTES USED

```
typedef enum {
    PATTR_LINE,
    PATTR_MARKER,
    PATTR_TEXT,
    PATTR_INT,
    PATTR_EDGE
} Pattrs;
```

`Pcieluv` CIE L*U*V*

```
typedef struct {
  Pfloat  cieluv_x;          /* x coefficient */
  Pfloat  cieluv_y;          /* y coefficient */
  Pfloat  cieluv_y_lum;      /* y luminance   */
} Pcieluv;
```

`Pclip_ind` CLIPPING INDICATOR

```
typedef enum {
    PIND_NO_CLIP,
    PIND_CLIP
} Pclip_ind;
```

`Pcolr_avail` COLOUR AVAILABILITY

```
typedef enum {
    PAVAIL_MONOCHR,
    PAVAIL_COLR
} Pcolr_avail;
```

`Pcolr_facs` COLOUR FACILITIES

```
typedef struct {
  Pint         num_colrs;          /* number of colours                    */
  Pcolr_avail  colr_avail;         /* colour availability                  */
  Pint         num_pred_inds;      /* number of predefined colour indices  */
  Pcieluv      prim_colrs[3];      /* primary colours                      */
} Pcolr_facs;
```

`Pcompose_type` COMPOSITION TYPE

```
typedef enum {
    PTYPE_PRECONCAT,
    PTYPE_POSTCONCAT,
    PTYPE_REPLACE
} Pcompose_type;
```

`Pconf_res` CONFLICT RESOLUTION

```
typedef enum {
    PRES_MAINTAIN,
    PRES_ABANDON,
    PRES_UPD
} Pconf_res;
```

`Pctrl_flag` CONTROL FLAG

```
typedef enum {
    PFLAG_COND,
    PFLAG_ALWAYS
} Pctrl_flag;
```

`Pdata` DATA

```
typedef struct {
  size_t  size;       /* sizeof data     */
  void    *data;      /* pointer to data */
} Pdata;
```

`Pdc_units` DEVICE COORDINATE UNITS

```
typedef enum {
    PDC_METRES,
    PDC_OTHER
} Pdc_units;
```

`Pdefer_mode` DEFERRAL MODE

```
typedef enum {
    PDEFER_ASAP,
    PDEFER_BNIG,
    PDEFER_BNIL,
    PDEFER_ASTI,
    PDEFER_WAIT
} Pdefer_mode;
```

`Pdisp_space_size` DISPLAY SPACE SIZE

```
typedef struct {

  Pdc_units    dc_units;        /* device coordinate units         */
  Pfloat_size  size_dc;         /* device size in coordinate units */
  Pint_size    size_raster;     /* device size in raster units     */

} Pdisp_space_size;
```

`Pdisp_space_size3` DISPLAY SPACE SIZE 3

```
typedef struct {

  Pdc_units     dc_units;       /* device coordinate units           */
  Pfloat_size3  size_dc;        /* device volume in coordinate units */
  Pint_size3    size_raster;    /* device volume in raster units     */

} Pdisp_space_size3;
```

`Pdisp_surf_empty` DISPLAY SURFACE EMPTY

```
typedef enum {
    PSURF_NOT_EMPTY,
    PSURF_EMPTY
} Pdisp_surf_empty;
```

`Pdyns_structs` DYNAMICS OF STRUCTURES

```
typedef struct {
  Pdyn_mod  content;      /* structure content     */
  Pdyn_mod  post;         /* post structure        */
  Pdyn_mod  unpost;       /* unpost structure      */
  Pdyn_mod  del;          /* delete structure      */
  Pdyn_mod  ref;          /* structure reference   */
} Pdyns_structs;
```

`Pdyns_ws_attrs` DYNAMICS OF WORKSTATION ATTRIBUTES

```
typedef struct {
  Pdyn_mod  line_bundle;       /* polyline bundle representation   */
  Pdyn_mod  marker_bundle;     /* polymarker bundle representation */
  Pdyn_mod  text_bundle;       /* text bundle representation       */
  Pdyn_mod  int_bundle;        /* interior bundle representation   */
  Pdyn_mod  edge_bundle;       /* edge bundle representation       */
  Pdyn_mod  pat_rep;           /* pattern representation           */
  Pdyn_mod  colr_rep;          /* colour representation            */
  Pdyn_mod  view_rep;          /* view representation              */
  Pdyn_mod  ws_tran;           /* workstation transformation       */
  Pdyn_mod  highl_filter;      /* highlighting filter              */
  Pdyn_mod  invis_filter;      /* invisibility filter              */
  Pdyn_mod  hlhsr_mode;        /* HLHSR mode                       */
} Pdyns_ws_attrs;
```

`Pdyn_mod` DYNAMIC MODIFICATION

```
typedef enum {
    PDYN_IRG,
    PDYN_IMM,
    PDYN_CBS
} Pdyn_mod;
```

`Pecho_switch` ECHO SWITCH

```
typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;
```

`Pedge_attrs` EDGE ATTRIBUTES

```
typedef struct {
  Pasf          flag_asf;        /* edge flag asf          */
  Pasf          type_asf;        /* edgetype asf           */
  Pasf          width_asf;       /* edgewidth asf          */
  Pasf          colr_ind_asf;    /* edge colour index asf */
  Pint          ind;             /* edge index             */
  Pedge_bundle  bundle;          /* edge bundle            */
} Pedge_attrs;
```

`Pedge_bundle` EDGE BUNDLE

```
typedef struct {
  Pedge_flag  flag;          /* edge flag              */
  Pint        type;          /* edgetype               */
  Pfloat      width;         /* edgewidth scale factor */
  Pint        colr_ind;      /* edge colour index      */
} Pedge_bundle;
```

`Pedge_facs` EDGE FACILITIES

```
typedef struct {
  Pint_list  types;             /* list of edgetypes                    */
  Pint       num_widths;        /* number of available edgewidths       */
  Pfloat     nom_width;         /* nominal edgewidth                    */
  Pfloat     min_width;         /* min edgewidth                        */
  Pfloat     max_width;         /* max edgewidth                        */
  Pint       num_pred_inds;     /* number of predefined bundle indices */
} Pedge_facs;
```

Pedge_flag EDGE FLAG

```
typedef enum {
    PEDGE_OFF,
    PEDGE_ON
} Pedge_flag;
```

Pedit_mode EDIT MODE

```
typedef enum {
    PEDIT_INSERT,
    PEDIT_REPLACE
} Pedit_mode;
```

Pelem_data ELEMENT DATA

```
typedef union {
  Pint              int_data;                          /* integer valued data     */
  Pfloat            float_data;                        /* float valued data       */
  Ppoint_list3      point_list3;                       /* list of 3d points       */
  Ppoint_list       point_list;                        /* list of 2d points       */
  Ppoint_list_list3 point_list_list3;                  /* list of 3d point lists */
  Ppoint_list_list  point_list_list;                   /* list of 2d point lists */
  struct Pelem_text3 {
                    Ppoint3         pos;               /* text position           */
                    Pvec3           dir[2];            /* direction vectors       */
                    char            *char_string;      /* character string        */
  } text3;
  struct Pelem_text {
                    Ppoint          pos;               /* text position           */
                    char            *char_string;      /* character string        */
  } text;
  struct Pelem_anno_text_rel3 {
                    Ppoint3         ref_point;         /* reference point         */
                    Pvec3           offset;            /* anno. offset            */
                    char            *char_string;      /* character string        */
  } anno_text_rel3;
  struct Pelem_anno_text_rel {
                    Ppoint          ref_point;         /* reference point         */
                    Pvec            offset;            /* anno. offset            */
                    char            *char_string;      /* character string        */
  } anno_text_rel;
  struct Pelem_cell_array3 {
                    Pparal          paral;             /* parallelogram           */
                    Ppat_rep        colr_array;        /* colour array            */
  } cell_array3;
  struct Pelem_cell_array {
                    Prect           rect;              /* rectangle               */
                    Ppat_rep        colr_array;        /* colour array            */
  } cell_array;
  struct Pelem_gdp3 {
                    Pint            id;                /* GDP3 id                 */
                    Ppoint_list3    point_list;        /* point list              */
                    Pgdp_data3      data;              /* data record             */
  } gdp3;
  struct Pelem_gdp {
                    Pint            id;                /* GDP id                  */
                    Ppoint_list     point_list;        /* point list              */
                    Pgdp_data       data;              /* data record             */
  } gdp;
  Ptext_prec        text_prec;                         /* text precision          */
  Pvec              char_up_vec;                       /* char up vector          */
  Ptext_path        text_path;                         /* text path               */
  Ptext_align       text_align;                        /* text alignment          */
  Pint_style        int_style;                         /* interior style          */
  Pedge_flag        edge_flag;                         /* edge flag               */
```

```
    Ppoint              pat_ref_point;                  /* pattern ref. point
    Pfloat_size         pat_size;                       /* pattern size
    struct Pelem_pat_ref_point_vecs {
                        Ppoint3             ref_point;  /* pattern ref. point
                        Pvec3               ref_vec[2]; /* vectors
    } pat_ref_point_vecs;
    Pint_list           names;                          /* name sets
    struct Pelem_asf {
                        Paspect             id;         /* attribute id
                        Pasf                source;     /* attribute source
    } asf;
    struct Pelem_local_tran3 {
                        Pcompose_type       compose_type;   /* composition type
                        Pmatrix3            matrix;         /* tran. matrix
    } local_tran3;
    struct Pelem_local_tran {
                        Pcompose_type       compose_type;   /* composition type
                        Pmatrix             matrix;         /* tran. matrix
    } local_tran;
    Pmatrix3            global_tran3;                   /* global transform3
    Pmatrix             global_tran;                    /* global transform
    struct Pelem_model_clip3 {
                        Pint                op;             /* operator
                        Phalf_space_list3   half_spaces;    /* half space list
    } model_clip3;
    struct Pelem_model_clip {
                        Pint                op;             /* operator
                        Phalf_space_list    half_spaces;    /* half space list
    } model_clip;
    Pclip_ind           clip_ind;                       /* clipping indicator
    Pdata               appl_data;                      /* application data
    struct Pelem_gse {
                        Pint                id;             /* GSE id
                        Pgse_data           data;           /* GSE data record
    } gse;
} Pelem_data;
```

Pelem_ref ELEMENT REFERENCE

```
typedef struct {
    Pint  struct_id;      /* structure identifier */
    Pint  elem_pos;       /* element position     */
} Pelem_ref;
```

`Pelem_ref_list` ELEMENT REFERENCE LIST

```
typedef struct {
  Pint       num_elem_refs;      /* number of element references */
  Pelem_ref  *elem_refs;         /* list of element references   */
} Pelem_ref_list;
```

`Pelem_ref_list_list` ELEMENT REFERENCE LIST LIST

```
typedef struct {
  Pint            num_elem_ref_lists;      /* number of element references lists */
  Pelem_ref_list  *elem_ref_lists;         /* list of element reference lists    */
} Pelem_ref_list_list;
```

Pelem_type ELEMENT TYPE

```
typedef enum {
    PELEM_ALL,
    PELEM_NIL,
    PELEM_POLYLINE3,
    PELEM_POLYLINE,
    PELEM_POLYMARKER3,
    PELEM_POLYMARKER,
    PELEM_TEXT3,
    PELEM_TEXT,
    PELEM_ANNO_TEXT_REL3,
    PELEM_ANNO_TEXT_REL,
    PELEM_FILL_AREA3,
    PELEM_FILL_AREA,
    PELEM_FILL_AREA_SET3,
    PELEM_FILL_AREA_SET,
    PELEM_CELL_ARRAY3,
    PELEM_CELL_ARRAY,
    PELEM_GDP3,
    PELEM_GDP,
    PELEM_LINE_IND,
    PELEM_MARKER_IND,
    PELEM_TEXT_IND,
    PELEM_INT_IND,
    PELEM_EDGE_IND,
    PELEM_LINETYPE,
    PELEM_LINEWIDTH,
    PELEM_LINE_COLR_IND,
    PELEM_MARKER_TYPE,
    PELEM_MARKER_SIZE,
    PELEM_MARKER_COLR_IND,
    PELEM_TEXT_FONT,
    PELEM_TEXT_PREC,
    PELEM_CHAR_EXPAN,
    PELEM_CHAR_SPACE,
    PELEM_TEXT_COLR_IND,
    PELEM_CHAR_HT,
    PELEM_CHAR_UP_VEC,
    PELEM_TEXT_PATH,
    PELEM_TEXT_ALIGN,
    PELEM_ANNO_CHAR_HT,
    PELEM_ANNO_CHAR_UP_VEC,
    PELEM_ANNO_PATH,
    PELEM_ANNO_ALIGN,
    PELEM_ANNO_STYLE,
    PELEM_INT_STYLE,
    PELEM_INT_STYLE_IND,
    PELEM_INT_COLR_IND,
    PELEM_EDGE_FLAG,
    PELEM_EDGETYPE,
    PELEM_EDGEWIDTH,
```

```
    PELEM_EDGE_COLR_IND,
    PELEM_PAT_SIZE,
    PELEM_PAT_REF_POINT_VECS,
    PELEM_PAT_REF_POINT,
    PELEM_ADD_NAMES_SET,
    PELEM_REMOVE_NAMES_SET,
    PELEM_INDIV_ASF,
    PELEM_HLHSR_ID,
    PELEM_LOCAL_MODEL_TRAN3,
    PELEM_LOCAL_MODEL_TRAN,
    PELEM_GLOBAL_MODEL_TRAN3,
    PELEM_GLOBAL_MODEL_TRAN,
    PELEM_MODEL_CLIP_VOL3,
    PELEM_MODEL_CLIP_VOL,
    PELEM_MODEL_CLIP_IND,
    PELEM_RESTORE_MODEL_CLIP_VOL,
    PELEM_VIEW_IND,
    PELEM_EXEC_STRUCT,
    PELEM_LABEL,
    PELEM_APPL_DATA,
    PELEM_GSE,
    PELEM_PICK_ID
} Pelem_type;
```

Pelem_type_list ELEMENT TYPE LIST

```
typedef struct {
  Pint        num_elem_types;     /* number of element types */
  Pelem_type  *elem_types;        /* list of element types   */
} Pelem_type_list;
```

Perr_mode ERROR MODE

```
typedef enum {
    PERR_OFF,
    PERR_ON
} Perr_mode;
```

`Pfilter` FILTER

```
typedef struct {
  Pint_list  incl_set;      /* inclusion set */
  Pint_list  excl_set;      /* exclusion set */
} Pfilter;
```

`Pfilter_list` FILTER LIST

```
typedef struct {
  Pint     num_filters;     /* number of filters */
  Pfilter  *filters;        /* list of filters   */
} Pfilter_list;
```

`Pfloat_size` FLOAT SIZE

```
typedef struct {
  Pfloat  size_x;      /* x size */
  Pfloat  size_y;      /* y size */
} Pfloat_size;
```

`Pfloat_size3` FLOAT SIZE 3

```
typedef struct {
  Pfloat  size_x;      /* x size */
  Pfloat  size_y;      /* y size */
  Pfloat  size_z;      /* z size */
} Pfloat_size3;
```

`Pgse_id_dep` GSE IDENTIFIER DEPENDENCY

```
typedef struct {
  Pint         id;      /* GSE identifier                                  */
  Pws_dep_ind  ind;     /* workstation independent/dependent indicator */
} Pgse_id_dep;
```

`Pgse_id_dep_list` GSE IDENTIFIER DEPENDENCY LIST

```
typedef struct {
  Pint         num_id_facs;     /* number of identifiers/dependency element */
  Pgse_id_dep  *id_facs;        /* list of GSE facilities                   */
} Pgse_id_dep_list;
```

`Phalf_space` HALF SPACE

```
typedef struct {
  Ppoint  point;      /* point  */
  Pvec    norm;       /* normal */
} Phalf_space;
```

`Phalf_space3` HALF SPACE 3

```
typedef struct {
  Ppoint3  point;      /* point  */
  Pvec3    norm;       /* normal */
} Phalf_space3;
```

`Phalf_space_list` HALF SPACE LIST

```
typedef struct {
  Pint         num_half_spaces;      /* number of half spaces */
  Phalf_space  *half_spaces;         /* list of half spaces   */
} Phalf_space_list;
```

`Phalf_space_list3` HALF SPACE LIST 3

```
typedef struct {
  Pint          num_half_spaces;     /* number of half spaces */
  Phalf_space3  *half_spaces;        /* list of half spaces   */
} Phalf_space_list3;
```

`Phls` HUE LIGHTNESS SATURATION

```
typedef struct {
  Pfloat  hue;           /* hue        */
  Pfloat  lightness;     /* lightness  */
  Pfloat  satur;         /* saturation */
} Phls;
```

`Phor_text_align` HORIZONTAL TEXT ALIGNMENT

```
typedef enum {
    PHOR_NORM,
    PHOR_LEFT,
    PHOR_CTR,
    PHOR_RIGHT
} Phor_text_align;
```

`Phsv` HUE SATURATION VALUE

```
typedef struct {
  Pfloat  hue;        /* hue        */
  Pfloat  satur;      /* saturation */
  Pfloat  value;      /* value      */
} Phsv;
```

`Pinq_type` INQUIRE TYPE

```
typedef enum {
    PINQ_SET,
    PINQ_REALIZED
} Pinq_type;
```

`Pint_attrs` INTERIOR ATTRIBUTES

```
typedef struct {
  Pasf         style_asf;         /* interior style asf        */
  Pasf         style_ind_asf;     /* interior style index asf  */
  Pasf         colr_ind_asf;      /* interior colour index asf */
  Pint         ind;               /* interior index            */
  Pint_bundle  bundle;            /* interior bundle           */
} Pint_attrs;
```

`Pint_bundle` INTERIOR BUNDLE

```
typedef struct {
  Pint_style  style;          /* interior style        */
  Pint        style_ind;      /* interior style index  */
  Pint        colr_ind;       /* interior colour index */
} Pint_bundle;
```

`Pint_facs` INTERIOR FACILITIES

```
typedef struct {
  Pint        num_int_styles;     /* number of interior styles           */
  Pint_style  int_styles[5];      /* list of available interior styles   */
  Pint_list   hatch_styles;       /* list of available hatch styles      */
  Pint        num_pred_inds;      /* number of predefined bundle indices */
} Pint_facs;
```

`Pint_list` INTEGER LIST

```
typedef struct {
  Pint  num_ints;      /* number of Pints in list */
  Pint  *ints;         /* list of integers        */
} Pint_list;
```

`Pint_size` INTEGER SIZE

```
typedef struct {
  Pint  size_x;      /* x size */
  Pint  size_y;      /* y size */
} Pint_size;
```

`Pint_size3` INTEGER SIZE 3

```
typedef struct {
  Pint  size_x;      /* x size */
  Pint  size_y;      /* y size */
  Pint  size_z;      /* z size */
} Pint_size3;
```

`Pint_style` INTERIOR STYLE

```
typedef enum {
    PSTYLE_HOLLOW,
    PSTYLE_SOLID,
    PSTYLE_PAT,
    PSTYLE_HATCH,
    PSTYLE_EMPTY
} Pint_style;
```

`Pin_class` INPUT CLASS

```
typedef enum {
    PIN_NONE,
    PIN_LOC,
    PIN_STROKE,
    PIN_VAL,
    PIN_CHOICE,
    PIN_PICK,
    PIN_STRING
} Pin_class;
```

`Pin_status` INPUT STATUS

```
typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;
```

`Plimit` LIMIT

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

`Plimit3` LIMIT 3

```
typedef struct {
  Pfloat  x_min;      /* x min */
  Pfloat  x_max;      /* x max */
  Pfloat  y_min;      /* y min */
  Pfloat  y_max;      /* y max */
  Pfloat  z_min;      /* z min */
  Pfloat  z_max;      /* z max */
} Plimit3;
```

`Pline_attrs` POLYLINE ATTRIBUTES

```
typedef struct {
  Pasf          type_asf;       /* linetype asf          */
  Pasf          width_asf;      /* linewidth asf         */
  Pasf          colr_ind_asf;   /* line colour index asf */
  Pint          ind;            /* line index            */
  Pline_bundle  bundle;         /* line bundle           */
} Pline_attrs;
```

`Pline_bundle` POLYLINE BUNDLE

```
typedef struct {
  Pint    type;          /* linetype                */
  Pfloat  width;         /* linewidth scale factor */
  Pint    colr_ind;      /* colour index            */
} Pline_bundle;
```

`Pline_facs` POLYLINE FACILITIES

```
typedef struct {
  Pint_list  types;              /* list of linetypes                    */
  Pint       num_widths;         /* number of available linewidths       */
  Pfloat     nom_width;          /* nominal linewidth                    */
  Pfloat     min_width;          /* min linewidth                        */
  Pfloat     max_width;          /* max linewidth                        */
  Pint       num_pred_inds;      /* number of predefined bundle indices */
} Pline_facs;
```

`Pline_fill_ctrl_flag` POLYLINE FILL CONTROL FLAG

```
typedef enum {
    PFLAG_LINE,
    PFLAG_FILL,
    PFLAG_FILL_SET
} Pline_fill_ctrl_flag;
```

`Pmarker_attrs` POLYMARKER ATTRIBUTES

```
typedef struct {
  Pasf            type_asf;        /* marker type asf                 */
  Pasf            size_asf;        /* marker size scale factor asf */
  Pasf            colr_ind_asf;    /* marker colour index asf      */
  Pint            ind;             /* marker index                    */
  Pmarker_bundle  bundle;          /* marker bundle                   */
} Pmarker_attrs;
```

`Pmarker_bundle` POLYMARKER BUNDLE

```
typedef struct {
  Pint    type;          /* marker type              */
  Pfloat  size;          /* marker size scale factor */
  Pint    colr_ind;      /* colour index             */
} Pmarker_bundle;
```

`Pmarker_facs` POLYMARKER FACILITIES

```
typedef struct {
  Pint_list  types;            /* list of marker types                */
  Pint       num_sizes;        /* number of available marker sizes    */
  Pfloat     nom_size;         /* nominal marker size                 */
  Pfloat     min_size;         /* min marker size                     */
  Pfloat     max_size;         /* max marker size                     */
  Pint       num_pred_inds;    /* number of predefined bundle indices */
} Pmarker_facs;
```

`Pmatrix` MATRIX

```
typedef Pfloat Pmatrix[3][3];
```

`Pmatrix3` MATRIX 3

```
typedef Pfloat Pmatrix3[4][4];
```

`Pmod_mode` MODIFICATION MODE

```
typedef enum {
    PMODE_NIVE,
    PMODE_UWOR,
    PMODE_UQUM
} Pmod_mode;
```

`Pmore_simult_events` MORE SIMULTANEOUS EVENTS

```
typedef enum {
    PSIMULT_NO_MORE,
    PSIMULT_MORE
} Pmore_simult_events;
```

Pnum_in NUMBER OF INPUT DEVICES

```
typedef struct {
  Pint  loc;        /* locators  */
  Pint  stroke;     /* strokes   */
  Pint  val;        /* valuators */
  Pint  choice;     /* choices   */
  Pint  pick;       /* picks     */
  Pint  string;     /* strings   */
} Pnum_in;
```

Popen_struct_status OPEN STRUCTURE STATUS

```
typedef enum {
    PSTRUCT_NONE,
    PSTRUCT_OPEN
} Popen_struct_status;
```

Pop_mode OPERATING MODE

```
typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
} Pop_mode;
```

Pparal PARALLELOGRAM

```
typedef struct {
  Ppoint3  p;      /* point p */
  Ppoint3  q;      /* point q */
  Ppoint3  r;      /* point r */
} Pparal;
```

`Ppath_order` PATH ORDER

```
typedef enum {
    PORDER_TOP_FIRST,
    PORDER_BOTTOM_FIRST
} Ppath_order;
```

`Ppat_rep` PATTERN REPRESENTATION

```
typedef struct {
  Pint_size  dims;          /* pattern's dimensions */
  Pint       *colr_array;   /* colour index array   */
} Ppat_rep;
```

`Ppick_path` PICK PATH

```
typedef struct {
  Pint             depth;       /* pick path depth */
  Ppick_path_elem  *path_list;  /* pick path list  */
} Ppick_path;
```

`Ppick_path_elem` PICK PATH ELEMENT

```
typedef struct {
  Pint  struct_id;      /* structure identifier */
  Pint  pick_id;        /* pick identifier      */
  Pint  elem_pos;       /* element position     */
} Ppick_path_elem;
```

Ppoint POINT

```
typedef struct {
  Pfloat  x;      /* x coordinate */
  Pfloat  y;      /* y coordinate */
} Ppoint;
```

Ppoint3 POINT 3

```
typedef struct {
  Pfloat  x;      /* x coordinate */
  Pfloat  y;      /* y coordinate */
  Pfloat  z;      /* z coordinate */
} Ppoint3;
```

Ppoint_list POINT LIST

```
typedef struct {
  Pint    num_points;     /* number of Ppoints in the list */
  Ppoint  *points;        /* list of points                */
} Ppoint_list;
```

Ppoint_list3 POINT LIST 3

```
typedef struct {
  Pint     num_points;     /* number of Ppoint3s in the list */
  Ppoint3  *points;        /* list of points                 */
} Ppoint_list3;
```

`Ppoint_list_list` POINT LIST LIST

```
typedef struct {
  Pint         num_point_lists;      /* number of point lists */
  Ppoint_list  *point_lists;         /* list of point lists   */
} Ppoint_list_list;
```

`Ppoint_list_list3` POINT LIST LIST 3

```
typedef struct {
  Pint          num_point_lists;     /* number of point lists */
  Ppoint_list3  *point_lists;        /* list of point lists   */
} Ppoint_list_list3;
```

`Pposted_struct` POSTED STRUCTURE

```
typedef struct {
  Pint    id;           /* structure id       */
  Pfloat  disp_pri;     /* display priority */
} Pposted_struct;
```

`Pposted_struct_list` POSTED STRUCTURE LIST

```
typedef struct {
  Pint            num_postings;     /* number of structure postings */
  Pposted_struct  *postings;        /* list of postings             */
} Pposted_struct_list;
```

`Pproj_type` PROJECTION TYPE

```
typedef enum {
    PTYPE_PARAL,
    PTYPE_PERSPECT
} Pproj_type;
```

Ppr_switch PROMPT SWITCH

```
typedef enum {
    PPR_OFF,
    PPR_ON
} Ppr_switch;
```

Prect RECTANGLE

```
typedef struct {
  Ppoint  p;      /* point p */
  Ppoint  q;      /* point q */
} Prect;
```

Pref_flag REFERENCE FLAG

```
typedef enum {
    PFLAG_DEL,
    PFLAG_KEEP
} Pref_flag;
```

Pregen_flag REGENERATION FLAG

```
typedef enum {
    PFLAG_POSTPONE,
    PFLAG_PERFORM
} Pregen_flag;
```

Prel_pri RELATIVE PRIORITY

```
typedef enum {
    PPRI_HIGHER,
    PPRI_LOWER
} Prel_pri;
```

Prgb RED GREEN BLUE

```
typedef struct {
  Pfloat  red;        /* red intensity   */
  Pfloat  green;      /* green intensity */
  Pfloat  blue;       /* blue intensity  */
} Prgb;
```

Psearch_dir SEARCH DIRECTION

```
typedef enum {
    PDIR_BACKWARD,
    PDIR_FORWARD
} Psearch_dir;
```

Psearch_status SEARCH STATUS

```
typedef enum {
    PSEARCH_STATUS_FAILURE,
    PSEARCH_STATUS_SUCCESS
} Psearch_status;
```

Pstore STORE

```
typedef void *Pstore;
```

Pstruct_net_source STRUCTURE NETWORK SOURCE

```
typedef enum {
    PNET_CSS,
    PNET_AR
} Pstruct_net_source;
```

`Pstruct_st` STRUCTURE STATE

```
typedef enum {
    PSTRUCT_ST_STCL,
    PSTRUCT_ST_STOP
} Pstruct_st;
```

`Pstruct_status` STRUCTURE STATUS

```
typedef enum {
    PSTRUCT_STATUS_NON_EXISTENT,
    PSTRUCT_STATUS_EMPTY,
    PSTRUCT_STATUS_NOT_EMPTY
} Pstruct_status;
```

`Psys_st` SYSTEM STATE

```
typedef enum {
    PSYS_ST_PHCL,
    PSYS_ST_PHOP
} Psys_st;
```

`Ptext_align` TEXT ALIGNMENT

```
typedef struct {
  Phor_text_align   hor;      /* horizontal component */
  Pvert_text_align  vert;     /* vertical component   */
} Ptext_align;
```

`Ptext_bundle` TEXT BUNDLE

```
typedef struct {
  Pint        font;          /* text font                    */
  Ptext_prec  prec;          /* text precision               */
  Pfloat      char_expan;    /* character expansion factor */
  Pfloat      char_space;    /* character spacing            */
  Pint        colr_ind;      /* text colour index            */
} Ptext_bundle;
```

`Ptext_facs` TEXT FACILITIES

```
typedef struct {
  Pint             num_font_precs;      /* number of fonts and precisions        */
  Ptext_font_prec  *font_precs;         /* list of fonts and precisions          */
  Pint             num_char_hts;        /* number of character heights           */
  Pfloat           min_char_ht;         /* minimum height                        */
  Pfloat           max_char_ht;         /* maximum height                        */
  Pint             num_char_expans;     /* number of character expansion factors */
  Pfloat           min_char_expan;      /* minimum expansion factor              */
  Pfloat           max_char_expan;      /* maximum expansion factor              */
  Pint             num_pred_inds;       /* number of predefined bundle indices   */
} Ptext_facs;
```

`Ptext_font_prec` TEXT FONT AND PRECISION

```
typedef struct {
  Pint        font;      /* text font      */
  Ptext_prec  prec;      /* text precision */
} Ptext_font_prec;
```

Ptext_path TEXT PATH

```
typedef enum {
    PPATH_RIGHT,
    PPATH_LEFT,
    PPATH_UP,
    PPATH_DOWN
} Ptext_path;
```

Ptext_prec TEXT PRECISION

```
typedef enum {
    PPREC_STRING,
    PPREC_CHAR,
    PPREC_STROKE
} Ptext_prec;
```

Pupd_st UPDATE STATE

```
typedef enum {
    PUPD_NOT_PEND,
    PUPD_PEND
} Pupd_st;
```

Pvec VECTOR

```
typedef struct {
  Pfloat  delta_x;      /* delta x value */
  Pfloat  delta_y;      /* delta y value */
} Pvec;
```

Pvec3 VECTOR 3

```
typedef struct {
  Pfloat  delta_x;      /* delta x value */
  Pfloat  delta_y;      /* delta y value */
  Pfloat  delta_z;      /* delta z value */
} Pvec3;
```

Pvert_text_align VERTICAL TEXT ALIGNMENT

```
typedef enum {
    PVERT_NORM,
    PVERT_TOP,
    PVERT_CAP,
    PVERT_HALF,
    PVERT_BASE,
    PVERT_BOTTOM
} Pvert_text_align;
```

Pview_map VIEW MAPPING

```
typedef struct {
  Plimit  win;          /* window limits                */
  Plimit  proj_vp;      /* projection viewport limits */
} Pview_map;
```

Pview_map3 VIEW MAPPING 3

```
typedef struct {
  Plimit      win;              /* window limits              */
  Plimit3     proj_vp;          /* projection viewport limits */
  Pproj_type  proj_type;        /* projection type            */
  Ppoint3     proj_ref_point;   /* projection reference point */
  Pfloat      view_plane;       /* view plane distance        */
  Pfloat      back_plane;       /* back plane distance        */
  Pfloat      front_plane;      /* front plane distance       */
} Pview_map3;
```

`Pview_rep` VIEW REPRESENTATION

```
typedef struct {
  Pmatrix    ori_matrix;     /* orientation matrix     */
  Pmatrix    map_matrix;     /* mapping matrix         */
  Plimit     clip_limit;     /* clipping limits        */
  Pclip_ind  xy_clip;        /* X-Y clipping indicator */
} Pview_rep;
```

`Pview_rep3` VIEW REPRESENTATION 3

```
typedef struct {
  Pmatrix3   ori_matrix;     /* orientation matrix       */
  Pmatrix3   map_matrix;     /* mapping matrix           */
  Plimit3    clip_limit;     /* clipping limits          */
  Pclip_ind  xy_clip;        /* X-Y clipping indicator   */
  Pclip_ind  back_clip;      /* back clipping indicator  */
  Pclip_ind  front_clip;     /* front clipping indicator */
} Pview_rep3;
```

`Pvisual_st` VISUAL STATE

```
typedef enum {
    PVISUAL_ST_CORRECT,
    PVISUAL_ST_DEFER,
    PVISUAL_ST_SIMULATED
} Pvisual_st;
```

`Pws_cat` WORKSTATION CATEGORY

```
typedef enum {
    PCAT_OUT,
    PCAT_IN,
    PCAT_OUTIN,
    PCAT_MO,
    PCAT_MI
} Pws_cat;
```

`Pws_class` WORKSTATION CLASS

```
typedef enum {
    PCLASS_VEC,
    PCLASS_RASTER,
    PCLASS_OTHER
} Pws_class;
```

`Pws_dep_ind` WORKSTATION DEPENDENCY INDICATOR

```
typedef enum {
    PWS_INDEP,
    PWS_DEP
} Pws_dep_ind;
```

`Pws_st` WORKSTATION STATE

```
typedef enum {
    PWS_ST_WSCL,
    PWS_ST_WSOP
} Pws_st;
```

`Pws_st_tables` LENGTH OF WORKSTATION STATE TABLES

```
typedef struct {
  Pint  line_bundles;      /* max. # of polyline table entries   */
  Pint  mark_bundles;      /* max. # of polymarker table entries */
  Pint  text_bundles;      /* max. # of text table entries       */
  Pint  int_bundles;       /* max. # of interior table entries   */
  Pint  edge_bundles;      /* max. # of edge table entries       */
  Pint  pat_reps;          /* max. # of pattern table entries    */
  Pint  colr_reps;         /* max. # of colour table entries     */
  Pint  view_reps;         /* max. # of view table entries       */
} Pws_st_tables;
```

NOTES

1 The order of the enumerations in the types `Pclip_ind`, `Pcolr_avail`, `Pecho_switch`, and `Pdisp_surf_empty` have been reversed with respect to ISO/IEC 9592-1. This was done in order to keep the numerical value of the enumerations compatible with ISO/IEC 9593-1 (PHIGS Fortran).

2 The enumerations for the status value returned by the REQUEST input functions have been bound as one enumeration in type `Pin_status`; the values NO PICK and NO CHOICE have been bound as `PIN_STATUS_NO_IN`.

3 ISO/IEC 9592-1 has two large enumerations which include all the element types. These enumerations have been bound as one enumeration in the type `Pelem_type`. The bound value `PELEM_ALL` cannot be returned as a value by the functions INQUIRE CURRENT ELEMENT TYPE AND SIZE and INQUIRE ELEMENT TYPE AND SIZE.

6 Macro definitions

6.1 Function identifiers

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

Function Name Macro	Function Number
#define Pfn_open_phigs	(0)
#define Pfn_close_phigs	(1)
#define Pfn_open_ws	(2)
#define Pfn_close_ws	(3)
#define Pfn_redraw_all_structs	(4)
#define Pfn_upd_ws	(5)
#define Pfn_set_disp_upd_st	(6)
#define Pfn_message	(7)
#define Pfn_polyline3	(8)
#define Pfn_polyline	(9)
#define Pfn_polymarker3	(10)
#define Pfn_polymarker	(11)
#define Pfn_text3	(12)
#define Pfn_text	(13)
#define Pfn_anno_text_rel3	(14)
#define Pfn_anno_text_rel	(15)
#define Pfn_fill_area3	(16)
#define Pfn_fill_area	(17)
#define Pfn_fill_area_set3	(18)
#define Pfn_fill_area_set	(19)
#define Pfn_cell_array3	(20)
#define Pfn_cell_array	(21)
#define Pfn_gdp3	(22)
#define Pfn_gdp	(23)
#define Pfn_set_line_ind	(24)
#define Pfn_set_marker_ind	(25)
#define Pfn_set_text_ind	(26)
#define Pfn_set_int_ind	(27)
#define Pfn_set_edge_ind	(28)
#define Pfn_set_linetype	(29)
#define Pfn_set_linewidth	(30)
#define Pfn_set_line_colr_ind	(31)
#define Pfn_set_marker_type	(32)
#define Pfn_set_marker_size	(33)
#define Pfn_set_marker_colr_ind	(34)
#define Pfn_set_text_font	(35)
#define Pfn_set_text_prec	(36)
#define Pfn_set_char_expan	(37)
#define Pfn_set_char_space	(38)
#define Pfn_set_text_colr_ind	(39)
#define Pfn_set_char_ht	(40)
#define Pfn_set_char_up_vec	(41)
#define Pfn_set_text_path	(42)
#define Pfn_set_text_align	(43)

```
#define  Pfn_set_anno_char_ht                (44)
#define  Pfn_set_anno_char_up_vec            (45)
#define  Pfn_set_anno_path                   (46)
#define  Pfn_set_anno_align                  (47)
#define  Pfn_set_anno_style                  (48)
#define  Pfn_set_int_style                   (49)
#define  Pfn_set_int_style_ind               (50)
#define  Pfn_set_int_colr_ind                (51)
#define  Pfn_set_edge_flag                   (52)
#define  Pfn_set_edgetype                    (53)
#define  Pfn_set_edgewidth                   (54)
#define  Pfn_set_edge_colr_ind               (55)
#define  Pfn_set_pat_size                    (56)
#define  Pfn_set_pat_ref_point_vecs          (57)
#define  Pfn_set_pat_ref_point               (58)
#define  Pfn_add_names_set                   (59)
#define  Pfn_remove_names_set                (60)
#define  Pfn_set_indiv_asf                   (61)
#define  Pfn_set_line_rep                    (62)
#define  Pfn_set_marker_rep                  (63)
#define  Pfn_set_text_rep                    (64)
#define  Pfn_set_int_rep                     (65)
#define  Pfn_set_edge_rep                    (66)
#define  Pfn_set_pat_rep                     (67)
#define  Pfn_set_colr_rep                    (68)
#define  Pfn_set_highl_filter                (69)
#define  Pfn_set_invis_filter                (70)
#define  Pfn_set_colr_model                  (71)
#define  Pfn_set_hlhsr_id                    (72)
#define  Pfn_set_hlhsr_mode                  (73)
#define  Pfn_set_local_tran3                 (74)
#define  Pfn_set_local_tran                  (75)
#define  Pfn_set_global_tran3                (76)
#define  Pfn_set_global_tran                 (77)
#define  Pfn_set_model_clip_vol3             (78)
#define  Pfn_set_model_clip_vol              (79)
#define  Pfn_set_model_clip_ind              (80)
#define  Pfn_restore_model_clip_vol          (81)
#define  Pfn_set_view_ind                    (82)
#define  Pfn_set_view_rep3                   (83)
#define  Pfn_set_view_rep                    (84)
#define  Pfn_set_view_tran_in_pri            (85)
#define  Pfn_set_ws_win3                     (86)
#define  Pfn_set_ws_win                      (87)
#define  Pfn_set_ws_vp3                      (88)
#define  Pfn_set_ws_vp                       (89)
#define  Pfn_open_struct                     (90)
#define  Pfn_close_struct                    (91)
#define  Pfn_exec_struct                     (92)
#define  Pfn_label                           (93)
#define  Pfn_appl_data                       (94)
#define  Pfn_gse                             (95)
#define  Pfn_set_edit_mode                   (96)
#define  Pfn_copy_all_elems_struct           (97)
```

```
#define  Pfn_set_elem_ptr                    (98)
#define  Pfn_offset_elem_ptr                 (99)
#define  Pfn_set_elem_ptr_label              (100)
#define  Pfn_del_elem                        (101)
#define  Pfn_del_elem_range                  (102)
#define  Pfn_del_elems_labels                (103)
#define  Pfn_empty_struct                    (104)
#define  Pfn_del_struct                      (105)
#define  Pfn_del_struct_net                  (106)
#define  Pfn_del_all_structs                 (107)
#define  Pfn_change_struct_id                (108)
#define  Pfn_change_struct_refs              (109)
#define  Pfn_change_struct_id_refs           (110)
#define  Pfn_post_struct                     (111)
#define  Pfn_unpost_struct                   (112)
#define  Pfn_unpost_all_structs              (113)
#define  Pfn_open_ar_file                    (114)
#define  Pfn_close_ar_file                   (115)
#define  Pfn_ar_structs                      (116)
#define  Pfn_ar_struct_nets                  (117)
#define  Pfn_ar_all_structs                  (118)
#define  Pfn_set_conf_res                    (119)
#define  Pfn_ret_struct_ids                  (120)
#define  Pfn_ret_structs                     (121)
#define  Pfn_ret_struct_nets                 (122)
#define  Pfn_ret_all_structs                 (123)
#define  Pfn_ret_paths_ances                 (124)
#define  Pfn_ret_paths_descs                 (125)
#define  Pfn_del_structs_ar                  (126)
#define  Pfn_del_struct_nets_ar              (127)
#define  Pfn_del_all_structs_ar              (128)
#define  Pfn_set_pick_id                     (129)
#define  Pfn_set_pick_filter                 (130)
#define  Pfn_init_loc3                       (131)
#define  Pfn_init_loc                        (132)
#define  Pfn_init_stroke3                    (133)
#define  Pfn_init_stroke                     (134)
#define  Pfn_init_val3                       (135)
#define  Pfn_init_val                        (136)
#define  Pfn_init_choice3                    (137)
#define  Pfn_init_choice                     (138)
#define  Pfn_init_pick3                      (139)
#define  Pfn_init_pick                       (140)
#define  Pfn_init_string3                    (141)
#define  Pfn_init_string                     (142)
#define  Pfn_set_loc_mode                    (143)
#define  Pfn_set_stroke_mode                 (144)
#define  Pfn_set_val_mode                    (145)
#define  Pfn_set_choice_mode                 (146)
#define  Pfn_set_pick_mode                   (147)
#define  Pfn_set_string_mode                 (148)
#define  Pfn_req_loc3                        (149)
#define  Pfn_req_loc                         (150)
#define  Pfn_req_stroke3                     (151)
```

```
#define  Pfn_req_stroke                  (152)
#define  Pfn_req_val                     (153)
#define  Pfn_req_choice                  (154)
#define  Pfn_req_pick                    (155)
#define  Pfn_req_string                  (156)
#define  Pfn_sample_loc3                 (157)
#define  Pfn_sample_loc                  (158)
#define  Pfn_sample_stroke3              (159)
#define  Pfn_sample_stroke               (160)
#define  Pfn_sample_val                  (161)
#define  Pfn_sample_choice               (162)
#define  Pfn_sample_pick                 (163)
#define  Pfn_sample_string               (164)
#define  Pfn_await_event                 (165)
#define  Pfn_flush_events                (166)
#define  Pfn_get_loc3                    (167)
#define  Pfn_get_loc                     (168)
#define  Pfn_get_stroke3                 (169)
#define  Pfn_get_stroke                  (170)
#define  Pfn_get_val                     (171)
#define  Pfn_get_choice                  (172)
#define  Pfn_get_pick                    (173)
#define  Pfn_get_string                  (174)
#define  Pfn_write_item                  (175)
#define  Pfn_get_item_type               (176)
#define  Pfn_read_item                   (177)
#define  Pfn_interpret_item              (178)
#define  Pfn_set_err_hand_mode           (179)
#define  Pfn_escape                      (180)
#define  Pfn_set_err_hand                (181)
```

6.2 Error codes

```
/*                              <0          Implementation Dependent Errors */

#define  PE_NO_ERROR            (0)     /* No Error */

/* State Errors */
#define  PE_NOT_PHCL            (1)     /* Ignoring function, function requires
                                           state (PHCL,WSCL,STCL,ARCL) */
#define  PE_NOT_PHOP            (2)     /* Ignoring function, function requires
                                           state (PHOP,*,*,*) */
#define  PE_NOT_WSOP            (3)     /* Ignoring function, function requires
                                           state (PHOP,WSOP,*,*) */
#define  PE_NOT_CL              (4)     /* Ignoring function, function requires
                                           state (PHOP,WSCL,STCL,ARCL) */
#define  PE_NOT_STOP            (5)     /* Ignoring function, function requires
                                           state (PHOP,*,STOP,*) */
#define  PE_NOT_STCL            (6)     /* Ignoring function, function requires
                                           state (PHOP,*,STCL,*) */
#define  PE_NOT_AROP            (7)     /* Ignoring function, function requires
                                           state (PHOP,*,*,AROP) */
```

```
/* Workstation Errors */
#define  PE_BAD_CONN_ID          (50)    /* Ignoring function, connection iden-
                                            tifier not recognized by the imple-
                                            mentation */
#define  PE_WS_TYPE              (51)    /* Ignoring function, this information
                                            is not yet available for this works-
                                            tation type; open a workstation of
                                            this type and use the specific works-
                                            tation type */
#define  PE_BAD_WS_TYPE          (52)    /* Ignoring function, workstation type
                                            not recognized by the implementation
                                            */
#define  PE_DUP_WS_ID            (53)    /* Ignoring function, workstation iden-
                                            tifier already is in use */
#define  PE_WS_NOT_OPEN          (54)    /* Ignoring function, the specified
                                            workstation is not open */
#define  PE_NO_OPEN_WS           (55)    /* Ignoring function, workstation cannot
                                            be opened for an implementation
                                            dependent reason */
#define  PE_WS_NOT_MO            (56)    /* Ignoring function, specified worksta-
                                            tion is not of category MO */
#define  PE_WS_MI                (57)    /* Ignoring function, specified worksta-
                                            tion is of category MI */
#define  PE_WS_NOT_MI            (58)    /* Ignoring function, specified worksta-
                                            tion is not of category MI */
#define  PE_WS_NO_OUTPUT         (59)    /* Ignoring function, the specified
                                            workstation does not have output
                                            capability (i.e., the workstation
                                            category is neither OUTPUT, OUTIN,
                                            nor MO) */
#define  PE_WS_NOT_OUTIN         (60)    /* Ignoring function, specified worksta-
                                            tion is not of category OUTIN */
#define  PE_WS_NO_INPUT          (61)    /* Ignoring function, specified worksta-
                                            tion is neither of category INPUT nor
                                            of category OUTIN */
#define  PE_WS_NOT_OUT           (62)    /* Ignoring function, specified worksta-
                                            tion is neither of category OUTPUT
                                            nor of category OUTIN */
#define  PE_MAX_WS               (63)    /* Ignoring function, opening this
                                            workstation would exceed the maximum
                                            number of simultaneously open works-
                                            tations */
#define  PE_NO_GDP               (64)    /* Ignoring function, the specified
                                            workstation type is not able to gen-
                                            erate the specified generalized draw-
                                            ing primitive */

/* Output Attribute Errors */
#define  PE_BUN_IND_LT_1         (100)   /* Ignoring function, the bundle index
                                            value is less than one */
#define  PE_REP_UNDEF            (101)   /* Ignoring function, the specified
                                            representation has not been defined
                                            */
#define  PE_REP_NOT_PREDEF       (102)   /* Ignoring function, the specified
```

```
                                        representation has not been prede-
                                        fined on this workstation */
#define  PE_MAX_BUN             (103)   /* Ignoring function, setting this bun-
                                        dle table entry would exceed the max-
                                        imum number of entries allowed in the
                                        workstation bundle table */
#define  PE_BAD_LINETYPE        (104)   /* Ignoring function, the specified
                                        linetype is not available on the
                                        specified workstation */
#define  PE_BAD_MARKER_TYPE     (105)   /* Ignoring function, the specified
                                        marker type is not available on the
                                        specified workstation */
#define  PE_BAD_FONT            (106)   /* Ignoring function, the specified font
                                        is not available for the requested
                                        text precision on the specified
                                        workstation */
#define  PE_BAD_EDGETYPE        (107)   /* Ignoring function, the specified
                                        edgetype is not available on the
                                        specified workstation */
#define  PE_BAD_INT_STYLE       (108)   /* Ignoring function, the specified
                                        interior style is not available on
                                        the workstation */
#define  PE_NO_PAT              (109)   /* Ignoring function, interior style
                                        PATTERN is not supported on the
                                        workstation */
#define  PE_BAD_COLR_MODEL      (110)   /* Ignoring function, the specified
                                        colour model is not available on the
                                        workstation */
#define  PE_BAD_HLHSR_MODE      (111)   /* Ignoring function, the specified
                                        HLHSR mode is not available on the
                                        specified workstation */
#define  PE_PAT_IND_LT_1        (112)   /* Ignoring function, the pattern index
                                        value is less than one */
#define  PE_COLR_IND_LT_0       (113)   /* Ignoring function, the colour index
                                        value is less than zero */
#define  PE_VIEW_IND_LT_0       (114)   /* Ignoring function, the view index
                                        value is less than zero */
#define  PE_VIEW_IND_LT_1       (115)   /* Ignoring function, the view index
                                        value is less than one */
#define  PE_BAD_PAT_DIM         (116)   /* Ignoring function, one of the dimen-
                                        sions of pattern colour array is less
                                        than one */
#define  PE_BAD_COLR_DIM        (117)   /* Ignoring function, one of the dimen-
                                        sions of the colour index array is
                                        less than zero */
#define  PE_BAD_COLR            (118)   /* Ignoring function, one of the com-
                                        ponents of the colour specification
                                        is out of range.  The valid range is
                                        dependent upon the current colour
                                        model */

/* Transformations and Viewing Errors */
#define  PE_MAX_VIEW            (150)   /* Ignoring function, setting this view
                                        table entry would exceed the maximum
```

```
                                        number of entries allowed in the
                                        workstations view table */
#define PE_INVALID_WINDOW      (151)    /* Ignoring function, invalid window;
                                        XMIN >= XMAX, YMIN >= YMAX or ZMIN >
                                        ZMAX */
#define PE_INVALID_VIEWPORT    (152)    /* Ignoring function, invalid viewport;
                                        XMIN >= XMAX, YMIN >= YMAX or ZMIN >
                                        ZMAX */
#define PE_INVALID_CLIP        (153)    /* Ignoring function, invalid view clip-
                                        ping limits; XMIN >= XMAX, YMIN >=
                                        YMAX or ZMIN > ZMAX */
#define PE_BAD_CLIP            (154)    /* Ignoring function, the view clipping
                                        limits are not within NPC range */
#define PE_BAD_PROJ_VIEWPORT   (155)    /* Ignoring function, the projection
                                        viewport limits are not withing NPC
                                        range */
#define PE_BAD_WS_WINDOW       (156)    /* Ignoring function, the workstation
                                        window limits are not within NPC
                                        range */
#define PE_BAD_WS_VIEWPORT     (157)    /* Ignoring function, the workstation
                                        viewport is not within display space
                                        */
#define PE_BAD_PLANES          (158)    /* Ignoring function, front plane and
                                        back plane distances are equal when
                                        z-extent of the projection viewport
                                        is zero */
#define PE_BAD_VPN             (159)    /* Ignoring function, the view plane
                                        normal vector has length zero */
#define PE_BAD_VUP             (160)    /* Ignoring function, the view up vector
                                        has length zero */
#define PE_BAD_VUP_VPN         (161)    /* Ignoring function, the view up and
                                        view plane normal vectors are paral-
                                        lel thus the viewing coordinate sys-
                                        tem cannot be established */
#define PE_BAD_PRP             (162)    /* Ignoring function, the projection
                                        reference point is between the front
                                        and back planes */
#define PE_PRP_VIEW_PLANE      (163)    /* Ignoring function, the projection
                                        reference point cannot be positioned
                                        on the view plane */
#define PE_FRONT_BACK          (164)    /* Ignoring function, the back plane is
                                        in front of the front plane */

/* Structure Errors */
#define PE_IGNORE_STRUCTS      (200)    /* Warning, ignoring structures that do
                                        not exist */
#define PE_BAD_STRUCT          (201)    /* Ignoring function, the specified
                                        structure does not exist */
#define PE_BAD_ELEMENT         (202)    /* Ignoring function, the specified ele-
                                        ment does not exist */
#define PE_BAD_PATH            (203)    /* Ignoring function, specified starting
                                        path not found in CSS  */
#define PE_BAD_CEILING_IND     (204)    /* Ignoring function, specified search
                                        ceiling index out of range */
```

```
#define  PE_NO_LABEL            (205)   /* Ignoring function, the label does not
                                           exist in the open structure between
                                           the element pointer and the end of
                                           the structure */
#define  PE_NO_LABELS           (206)   /* Ignoring function, one or both of the
                                           labels does not exist in the open
                                           structure between the element pointer
                                           and the end of the structure */
#define  PE_BAD_PATH_DEPTH      (207)   /* Ignoring function, the specified
                                           path depth is less than zero (0) */
#define  PE_BAD_DISP_PRI        (208)   /* Ignoring function, the display prior-
                                           ity is out of range */

/* Input Errors */
#define  PE_NO_DEVICE           (250)   /* Ignoring function, the specified dev-
                                           ice is not available on the specified
                                           workstation */
#define  PE_NOT_REQUEST         (251)   /* Ignoring function, the function
                                           requires the input device to be in
                                           REQUEST mode */
#define  PE_NOT_SAMPLE          (252)   /* Ignoring function, the function
                                           requires the input device to be in
                                           SAMPLE mode */
#define  PE_BAD_PET             (253)   /* Warning, the specified prompt/echo
                                           type is not available on the speci-
                                           fied workstation. Prompt/echo type
                                           one will be used in its place */
#define  PE_INVALID_ECHO        (254)   /* Ignoring function, invalid echo
                                           area/volume; XMIN >= XMAX, YMIN >=
                                           YMAX or ZMIN > ZMAX */
#define  PE_BAD_ECHO            (255)   /* Ignoring function, one of the echo
                                           area/volume boundary points is out-
                                           side the range of the device */
#define  PE_QUEUE_OFLOW         (256)   /* Warning, the input queue has over-
                                           flowed */
#define  PE_NO_QUEUE_OFLOW      (257)   /* Ignoring function, input queue has
                                           not overflowed */
#define  PE_OFLOW_NO_GO         (258)   /* Ignoring function, input queue has
                                           overflowed, but associated worksta-
                                           tion has been closed */
#define  PE_BAD_CLASS           (259)   /* Ignoring function, the input device
                                           class of the current input report
                                           does not match the class being
                                           requested */
#define  PE_BAD_DATA_REC        (260)   /* Ignoring function, one of the fields
                                           within the input device data record
                                           is in error */
#define  PE_INVALID_VALUE       (261)   /* Ignoring function, initial value is
                                           invalid */
#define  PE_STROKE_BUF_SIZE     (262)   /* Ignoring function, number of points
                                           in the initial stroke is greater than
                                           the buffer size */
#define  PE_STRING_BUF_SIZE     (263)   /* Ignoring function, length of the ini-
                                           tial string is greater than the
```

```
                                        buffer size */

/* Metafile Errors */
#define  PE_ILLEGAL_ITEM_TYPE  (300)    /* Ignoring function, item type is not
                                           allowed for user items */
#define  PE_INVALID_ITEM_LENGTH         (301)/*Ignoring function, item length is
                                           invalid */
#define  PE_METAFILE_EMPTY     (302)    /* Ignoring function, no item is left in
                                           metafile input */
#define  PE_INVALID_ITEM       (303)    /* Ignoring function, metafile item is
                                           invalid */
#define  PE_BAD_ITEM_TYPE      (304)    /* Ignoring function, item type is
                                           unknown */
#define  PE_BAD_ITEM_REC       (305)    /* Ignoring function, content of item
                                           data record is invalid for the speci-
                                           fied item type */
#define  PE_MAX_ITEM_LENGTH    (306)    /* Ignoring function, maximum item data
                                           record length is invalid */
#define  PE_USER_ITEM          (307)    /* Ignoring function, user item cannot
                                           be interpreted */

/* Escape Errors */
#define  PE_ESCAPE_NOT_AVAIL   (350)    /* Warning, the specified escape is not
                                           available on one or more workstations
                                           in this implementation.  The escape
                                           will be processed by those worksta-
                                           tions on which it is available */
#define  PE_BAD_ESCAPE_DATA    (351)    /* Ignoring function, one of the fields
                                           within the escape data record is in
                                           error */

/* Archival and Retrieval Errors */
#define  PE_AR_CANT_OPEN       (400)    /* Ignoring function, the archive file
                                           cannot be opened */
#define  PE_MAX_AR             (401)    /* Ignoring function, opening this
                                           archive file would exceed the maximum
                                           number of simultaneously open archive
                                           files */
#define  PE_DUP_AR_ID          (402)    /* Ignoring function, archive file iden-
                                           tifier already in use */
#define  PE_BAD_AR             (403)    /* Ignoring function, the archive file
                                           is not a PHIGS archive file */
#define  PE_AR_NOT_OPEN        (404)    /* Ignoring function, the specified
                                           archive file is not open */
#define  PE_NAME_CONFLICT      (405)    /* Ignoring function, name conflict
                                           occurred while conflict resolution
                                           flag has value ABANDON */
#define  PE_AR_FULL            (406)    /* Warning, the archive file is full.
                                           Any structures that were archived
                                           were archived in total */
#define  PE_AR_NO_STRUCT       (407)    /* Warning, some of the specified struc-
                                           tures do not exist on the archive
                                           file */
#define  PE_AR_NO_STRUCT_EMPTY (408)    /* Warning,  some  of  the  specified
```

```
                                        structures do not exist on the
                                        archive file.  PHIGS will create
                                        empty structure in their places */

/* Miscellaneous Errors */
#define PE_BAD_ERROR_FILE       (450)   /* Ignoring function, the specified
                                        error file is invalid */

/* System Errors */
#define PE_OFLOW_PHIGS          (900)   /* Storage overflow has occurred in
                                        PHIGS */
#define PE_OFLOW_CSS            (901)   /* Storage overflow has occurred in CSS
                                        */
#define PE_IO_ERROR_READ        (902)   /* Input/Output error has occurred while
                                        reading */
#define PE_IO_ERROR_WRITE       (903)   /* Input/Output error has occurred while
                                        writing */
#define PE_IO_ERROR_TO_WS       (904)   /* Input/Output error has occurred while
                                        sending data to a workstation */
#define PE_IO_ERROR_FROM_WS     (905)   /* Input/Output error has occurred
                                        receiving data from a workstation */
#define PE_IO_ERROR_LIB         (906)   /* Input/Output error has occurred dur-
                                        ing program library management */
#define PE_IO_ERROR_WDT         (907)   /* Input/Output error has occurred while
                                        reading workstation description table
                                        */
#define PE_ARITHMETIC_ERROR     (908)   /* Arithmetic error has occurred */

/*      Binding Specific Errors */
#define PE_START_IND_INVAL      (2200)  /* Ignoring function, start index is out
                                        of range */
#define PE_LIST_LENGTH_LT_ZERO          (2201)/*Ignoring function, the length of
                                        the application's list is negative */
#define PE_ENUM_TYPE_INVAL      (2202)  /* Ignoring function, enumeration type
                                        out of range */
#define PE_ALLOC_STORE          (2203)  /* Ignoring function, error while allo-
                                        cating a Store */
#define PE_ALLOC_STORE_MEM      (2204)  /* Ignoring function, error while allo-
                                        cating memory for a Store */
```

Reserved error numbers: 2202-2299

6.3 Miscellaneous

6.3.1 Linetypes

```
#define  PLINE_SOLID                    (1)
#define  PLINE_DASH                     (2)
#define  PLINE_DOT                      (3)
#define  PLINE_DASH_DOT                 (4)
```

6.3.2 Marker types

```
#define  PMARKER_DOT                    (1)
#define  PMARKER_PLUS                   (2)
#define  PMARKER_ASTERISK               (3)
#define  PMARKER_CIRCLE                 (4)
#define  PMARKER_CROSS                  (5)
```

6.3.3 Annotation styles

```
#define PANNO_STYLE_UNCONNECTED         (1)
#define PANNO_STYLE_LEAD_LINE           (2)
```

6.3.4 Colour models

```
#define  PMODEL_RGB                     (1)
#define  PMODEL_CIELUV                  (2)
#define  PMODEL_HSV                     (3)
#define  PMODEL_HLS                     (4)
```

6.3.5 Prompt and Echo Types

```
#define  PLOC_DEF                       (1)
#define  PLOC_CROSS_HAIR                (2)
#define  PLOC_TRACK_CROSS               (3)
#define  PLOC_RUB_BAND                  (4)
#define  PLOC_RECT                      (5)
#define  PLOC_DIGIT                     (6)

#define  PSTROKE_DEF                    (1)
#define  PSTROKE_DIGIT                  (2)
#define  PSTROKE_MARKER                 (3)
#define  PSTROKE_LINE                   (4)

#define  PVAL_DEF                       (1)
```

```
#define  PVAL_GRAPH                    (2)
#define  PVAL_DIGIT                    (3)

#define  PCHOICE_DEF                   (1)
#define  PCHOICE_PR_ECHO               (2)
#define  PCHOICE_STRING_PR             (3)
#define  PCHOICE_STRING_IN             (4)
#define  PCHOICE_STRUCT                (5)

#define  PPICK_DEF                     (1)
#define  PPICK_GROUP_HIGHL             (2)
#define  PPICK_STRUCT_NETWORK          (3)

#define PSTRING_DEF                    (1)
```

6.3.6 Default parameters of OPEN PHIGS

```
#define  PDEF_MEM_SIZE                 ((size_t) (-1))
#define  PDEF_ERR_FILE                 ((char *) (0))
```

6.3.7 Element enumeration

```
#define  PFIRST_PHIGS_ELEM             (PELEM_POLYLINE3)
#define  PLAST_PHIGS_ELEM              (PELEM_PICK_ID)
```

7 C PHIGS functions

7.1 Notational conventions

The format of the binding of each PHIGS function conforms to the following template:

PHIGS Function Name

```
void   pfunction(
       Ptype0   arg0,   /* argument 0 explanation   */
       Ptype1   arg1,   /* argument 1 explanation   */
       Ptype2   arg2    /* argument 2 explanation   */
);
```

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

The C function name bound to the PHIGS function is `pfunction`. `arg0`, `arg1`, and `arg2` are the arguments to the function and correspond to the parameters of the PHIGS function definition. `Ptype0`, `Ptype1`, and `Ptype2` are the C data types of the arguments. The definitions of these types are listed in clause 5 of this binding.

To the right of each argument declaration is a C comment field which contains a brief explanation of the argument. If the comment begins with "OUT" it means the argument is used as an output parameter; the implementation returns data to the application through this argument. Arguments without "OUT" are input parameters. The function DELETE STORE has a parameter labeled "IN/OUT"; this parameter is first used as an input paramter and then it is assigned a value by the implementation. In order to conserve space in the comment field, the symbol "#" is used as an abbreviation for "number of".

All C PHIGS functions return `void`.

7.2 Control functions

OPEN PHIGS

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

CLOSE PHIGS

```
void pclose_phigs(
     void
);
```

OPEN WORKSTATION

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

CLOSE WORKSTATION

```
void pclose_ws(
    Pint   ws_id   /* workstation identifier   */
);
```

REDRAW ALL STRUCTURES

```
void predraw_all_structs(
    Pint          ws_id,      /* workstation identifier   */
    Pctrl_flag    ctrl_flag   /* control flag             */
);
```

UPDATE WORKSTATION

```
void pupd_ws(
    Pint          ws_id,       /* workstation identifier     */
    Pregen_flag   regen_flag   /* update regeneration flag   */
);
```

SET DISPLAY UPDATE STATE

```
void pset_disp_upd_st(
    Pint          ws_id,      /* workstation identifier   */
    Pdefer_mode   def_mode,   /* deferral mode            */
    Pmod_mode     mod_mode    /* modification mode        */
);
```

MESSAGE

```
void pmessage(
    Pint          ws_id,      /* workstation identifier   */
    const char    *message    /* message string           */
);
```

7.3 Output primitive functions

POLYLINE 3

```
void ppolyline3(
    const Ppoint_list3   *point_list   /* list of points   */
);
```

POLYLINE

```
void ppolyline(
    const Ppoint_list   *point_list   /* list of points   */
);
```

POLYMARKER 3

```
void ppolymarker3(
    const Ppoint_list3   *point_list   /* list of points   */
);
```

POLYMARKER

```
void ppolymarker(
    const Ppoint_list   *point_list   /* list of points   */
);
```

TEXT 3

```
void ptext3(
    const Ppoint3   *text_pos,     /* text position          */
    const Pvec3     text_dir[2],   /* text direction vectors */
    const char      *char_string   /* character string       */
);
```

TEXT

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

ANNOTATION TEXT RELATIVE 3

```
void panno_text_rel3(
    const Ppoint3   *ref_point,    /* reference point              */
    const Pvec3     *offset,       /* annotation offset            */
    const char      *char_string   /* annotation character string  */
);
```

ANNOTATION TEXT RELATIVE

```
void panno_text_rel(
    const Ppoint    *ref_point,    /* reference point              */
    const Pvec      *offset,       /* annotation offset            */
    const char      *char_string   /* annotation character string  */
);
```

FILL AREA 3

```
void pfill_area3(
    const Ppoint_list3   *point_list   /* list of points   */
);
```

FILL AREA

```
void pfill_area(
    const Ppoint_list   *point_list   /* list of points   */
);
```

FILL AREA SET 3

```
void pfill_area_set3(
    const Ppoint_list_list3   *point_list_list   /* list of point lists   */
);
```

FILL AREA SET

```
void pfill_area_set(
    const Ppoint_list_list   *point_list_list   /* list of point lists   */
);
```

CELL ARRAY 3

```
void pcell_array3(
    const Pparal     *paral,       /* cell parallelogram   */
    const Ppat_rep   *colr_array   /* colour array         */
);
```

CELL ARRAY

```
void pcell_array(
     const Prect      *rect,        /* cell rectangle   */
     const Ppat_rep   *colr_array   /* colour array     */
);
```

GENERALIZED DRAWING PRIMITIVE 3

```
void pgdp3(
     const Ppoint_list3   *point_list,   /* list of points            */
     Pint                 gdp3_id,       /* gdp function identifier   */
     const Pgdp_data3     *gdp_data      /* gdp data record           */
);
```

GENERALIZED DRAWING PRIMITIVE

```
void pgdp(
     const Ppoint_list   *point_list,   /* list of points            */
     Pint                gdp_id,        /* gdp function identifier   */
     const Pgdp_data     *gdp_data      /* gdp data record           */
);
```

7.4 Attribute specification functions

7.4.1 Bundled attribute selection

SET POLYLINE INDEX

```
void pset_line_ind(
     Pint   line_ind   /* polyline index   */
);
```

SET POLYMARKER INDEX

```
void pset_marker_ind(
     Pint   marker_ind   /* polymarker index   */
);
```

SET TEXT INDEX

```
void pset_text_ind(
     Pint   text_ind   /* text index   */
);
```

SET INTERIOR INDEX

```
void pset_int_ind(
    Pint   int_ind   /* interior index   */
);
```

SET EDGE INDEX

```
void pset_edge_ind(
    Pint   edge_ind   /* edge index   */
);
```


7.4.2 Individual attribute selection

SET LINETYPE

```
void pset_linetype(
    Pint   linetype   /* linetype   */
);
```

SET LINEWIDTH SCALE FACTOR

```
void pset_linewidth(
    Pfloat   linewidth   /* linewidth scale factor   */
);
```

SET POLYLINE COLOUR INDEX

```
void pset_line_colr_ind(
    Pint   line_colr_ind   /* polyline colour index   */
);
```

SET MARKER TYPE

```
void pset_marker_type(
    Pint   marker_type   /* marker type   */
);
```

SET MARKER SIZE SCALE FACTOR

```
void pset_marker_size(
    Pfloat   marker_size   /* marker size scale factor   */
);
```

SET POLYMARKER COLOUR INDEX

```
void pset_marker_colr_ind(
    Pint   marker_colr_ind   /* polymarker colour index   */
);
```

SET TEXT FONT

```
void pset_text_font(
    Pint   font   /* text font   */
);
```

SET TEXT PRECISION

```
void pset_text_prec(
    Ptext_prec   prec   /* text precision   */
);
```

SET CHARACTER EXPANSION FACTOR

```
void pset_char_expan(
    Pfloat   char_expan   /* character expansion factor   */
);
```

SET CHARACTER SPACING

```
void pset_char_space(
    Pfloat   char_space   /* character spacing   */
);
```

SET TEXT COLOUR INDEX

```
void pset_text_colr_ind(
    Pint   text_colr_ind   /* text colour index   */
);
```

SET CHARACTER HEIGHT

```
void pset_char_ht(
    Pfloat   char_ht   /* character height   */
);
```

SET CHARACTER UP VECTOR

```
void pset_char_up_vec(
    const Pvec   *char_up_vec   /* character up vector   */
);
```

SET TEXT PATH

```
void pset_text_path(
    Ptext_path   text_path   /* text path   */
);
```

SET TEXT ALIGNMENT

```
void pset_text_align(
    const Ptext_align   *text_align   /* text alignment   */
);
```

SET ANNOTATION TEXT CHARACTER HEIGHT

```
void pset_anno_char_ht(
    Pfloat   char_ht   /* character height   */
);
```

SET ANNOTATION TEXT CHARACTER UP VECTOR

```
void pset_anno_char_up_vec(
    const Pvec   *char_up_vec   /* character up vector   */
);
```

SET ANNOTATION TEXT PATH

```
void pset_anno_path(
    Ptext_path   text_path   /* text path   */
);
```

SET ANNOTATION TEXT ALIGNMENT

```
void pset_anno_align(
    const Ptext_align   *text_align   /* text alignment   */
);
```

SET ANNOTATION STYLE

```
void pset_anno_style(
    Pint   anno_style   /* annotation style   */
);
```

SET INTERIOR STYLE

```
void pset_int_style(
    Pint_style   int_style   /* interior style   */
);
```

SET INTERIOR STYLE INDEX

```
void pset_int_style_ind(
    Pint   int_style_ind   /* interior style index   */
);
```

SET INTERIOR COLOUR INDEX

```
void pset_int_colr_ind(
    Pint   int_colr_ind   /* interior colour index   */
);
```

SET EDGE FLAG

```
void pset_edge_flag(
    Pedge_flag   edge_flag   /* edge flag   */
);
```

SET EDGETYPE

```
void pset_edgetype(
    Pint   edgetype   /* edgetype   */
);
```

SET EDGEWIDTH SCALE FACTOR

```
void pset_edgewidth(
    Pfloat   edgewidth   /* edgewidth scale factor   */
);
```

SET EDGE COLOUR INDEX

```
void pset_edge_colr_ind(
    Pint   edge_colr_ind   /* edge colour index   */
);
```

SET PATTERN SIZE

```
void pset_pat_size(
    const Pfloat_size   *pat_size   /* pattern size   */
);
```

SET PATTERN REFERENCE POINT AND VECTORS

```
void pset_pat_ref_point_vecs(
    const Ppoint3   *pat_ref_point,   /* pattern reference point                    */
    const Pvec3     pat_ref_vec[2]    /* reference vectors: 0 = X axis; 1 = Y axis   */
);
```

SET PATTERN REFERENCE POINT

```
void pset_pat_ref_point(
    const Ppoint   *pat_ref_point   /* pattern reference point   */
);
```

ADD NAMES TO SET

```
void padd_names_set(
    const Pint_list   *names   /* name set to be added   */
);
```

REMOVE NAMES FROM SET

```
void premove_names_set(
    const Pint_list   *names   /* name set to be removed   */
);
```

7.4.3 Aspect source flag setting

SET INDIVIDUAL ASF

```
void pset_indiv_asf(
    Paspect   asf_id,      /* attribute source identifier   */
    Pasf      asf_source   /* attribute source              */
);
```

7.4.4 Workstation attribute table definition

SET POLYLINE REPRESENTATION

```
void pset_line_rep(
    Pint                 ws_id,         /* workstation identifier    */
    Pint                 line_ind,      /* polyline bundle index     */
    const Pline_bundle   *line_bundle   /* polyline representation   */
);
```

SET POLYMARKER REPRESENTATION

```
void pset_marker_rep(
    Pint                   ws_id,          /* workstation identifier      */
    Pint                   marker_ind,     /* polymarker bundle index     */
    const Pmarker_bundle   *marker_bundle  /* polymarker representation   */
);
```

SET TEXT REPRESENTATION

```
void pset_text_rep(
    Pint                 ws_id,         /* workstation identifier   */
    Pint                 text_ind,      /* text bundle index        */
    const Ptext_bundle   *text_bundle   /* text representation      */
);
```

SET INTERIOR REPRESENTATION

```
void pset_int_rep(
    Pint                 ws_id,         /* workstation identifier    */
    Pint                 int_ind,       /* interior bundle index     */
    const Pint_bundle    *int_bundle    /* interior representation   */
);
```

SET EDGE REPRESENTATION

```
void pset_edge_rep(
    Pint                ws_id,          /* workstation identifier   */
    Pint                edge_ind,       /* edge bundle index        */
    const Pedge_bundle  *edge_bundle    /* edge representation      */
);
```

SET PATTERN REPRESENTATION

```
void pset_pat_rep(
    Pint              ws_id,        /* workstation identifier   */
    Pint              pat_ind,      /* pattern bundle index     */
    const Ppat_rep    *pat_bundle   /* pattern representation   */
);
```

SET COLOUR REPRESENTATION

```
void pset_colr_rep(
    Pint              ws_id,       /* workstation identifier   */
    Pint              colr_ind,    /* colour bundle index      */
    const Pcolr_rep   *colr_rep    /* colour representation    */
);
```

7.4.5 Workstation filter definition

SET HIGHLIGHTING FILTER

```
void pset_highl_filter(
    Pint            ws_id,     /* workstation identifier   */
    const Pfilter   *filter    /* highlighting filter      */
);
```

SET INVISIBILITY FILTER

```
void pset_invis_filter(
    Pint            ws_id,     /* workstation identifier   */
    const Pfilter   *filter    /* invisibility filter      */
);
```

7.4.6 Colour model control

SET COLOUR MODEL

```
void pset_colr_model(
    Pint   ws_id,        /* workstation identifier   */
    Pint   colr_model    /* colour model             */
);
```

7.4.7 HLHSR attributes

SET HLHSR IDENTIFIER

```
void pset_hlhsr_id(
    Pint   hlhsr_id   /* HLHSR identifier   */
);
```

SET HLHSR MODE

```
void pset_hlhsr_mode(
    Pint   ws_id,        /* workstation identifier   */
    Pint   hlhsr_mode    /* HLHSR mode               */
);
```

7.5 Transformation and clipping functions

7.5.1 Modelling transformations and clipping

SET LOCAL TRANSFORMATION 3

```
void pset_local_tran3(
    Pmatrix3        local_tran,    /* local transformation matrix   */
    Pcompose_type   compose_type   /* composition type              */
);
```

SET LOCAL TRANSFORMATION

```
void pset_local_tran(
    Pmatrix         local_tran,    /* local transformation matrix   */
    Pcompose_type   compose_type   /* composition type              */
);
```

SET GLOBAL TRANSFORMATION 3

```
void pset_global_tran3(
    Pmatrix3   global_tran   /* global transformation matrix   */
);
```

SET GLOBAL TRANSFORMATION

```
void pset_global_tran(
    Pmatrix   global_tran   /* global transformation matrix   */
);
```

SET MODELLING CLIPPING VOLUME 3

```
void pset_model_clip_vol3(
    Pint                      op,            /* operator                */
    const Phalf_space_list3   *half_spaces   /* list of half spaces     */
);
```

SET MODELLING CLIPPING VOLUME

```
void pset_model_clip_vol(
    Pint                      op,            /* operator                */
    const Phalf_space_list    *half_spaces   /* list of half spaces     */
);
```

SET MODELLING CLIPPING INDICATOR

```
void pset_model_clip_ind(
    Pclip_ind   clip_ind   /* clipping indicator   */
);
```

RESTORE MODELLING CLIPPING VOLUME

```
void prestore_model_clip_vol(
        void
);
```

7.5.2 View operation

SET VIEW INDEX

```
void pset_view_ind(
     Pint   view_ind   /* view index   */
);
```

SET VIEW REPRESENTATION 3

```
void pset_view_rep3(
     Pint                ws_id,      /* workstation identifier   */
     Pint                view_ind,   /* view index               */
     const Pview_rep3    *view_rep   /* view representation      */
);
```

SET VIEW REPRESENTATION

```
void pset_view_rep(
     Pint              ws_id,      /* workstation identifier   */
     Pint              view_ind,   /* view index               */
     const Pview_rep   *view_rep   /* view representation      */
);
```

SET VIEW TRANSFORMATION INPUT PRIORITY

```
void pset_view_tran_in_pri(
     Pint       ws_id,          /* workstation identifier   */
     Pint       view_ind,       /* view index               */
     Pint       ref_view_ind,   /* reference view index     */
     Prel_pri   rel_pri         /* relative priority        */
);
```

7.5.3 Workstation transformation

SET WORKSTATION WINDOW 3

```
void pset_ws_win3(
     Pint            ws_id,           /* workstation identifier      */
     const Plimit3   *ws_win_limits   /* workstation window limits   */
);
```

SET WORKSTATION WINDOW

```
void pset_ws_win(
    Pint           ws_id,          /* workstation identifier      */
    const Plimit   *ws_win_limits  /* workstation window limits   */
);
```

SET WORKSTATION VIEWPORT 3

```
void pset_ws_vp3(
    Pint           ws_id,          /* workstation identifier      */
    const Plimit3  *ws_vp_limits   /* workstation viewport limits */
);
```

SET WORKSTATION VIEWPORT

```
void pset_ws_vp(
    Pint           ws_id,          /* workstation identifier      */
    const Plimit   *ws_vp_limits   /* workstation viewport limits */
);
```

7.5.4 Utility functions to support modelling

TRANSLATE 3

```
void ptranslate3(
    const Pvec3    *trans_vec,     /* translation vector          */
    Pint           *err_ind,       /* OUT error indicator         */
    Pmatrix3       result_tran     /* OUT transformation matrix   */
);
```

TRANSLATE

```
void ptranslate(
    const Pvec     *trans_vec,     /* translation vector          */
    Pint           *err_ind,       /* OUT error indicator         */
    Pmatrix        result_tran     /* OUT transformation matrix   */
);
```

SCALE 3

```
void pscale3(
    const Pvec3   *scale_vec,   /* scale factor vector          */
    Pint          *err_ind,     /* OUT error indicator          */
    Pmatrix3      result_tran   /* OUT transformation matrix    */
);
```

SCALE

```
void pscale(
    const Pvec    *scale_vec,   /* scale factor vector          */
    Pint          *err_ind,     /* OUT error indicator          */
    Pmatrix       result_tran   /* OUT transformation matrix    */
);
```

ROTATE X

```
void protate_x(
    Pfloat     angle,         /* rotation angle               */
    Pint       *err_ind,      /* OUT error indicator          */
    Pmatrix3   result_tran    /* OUT transformation matrix    */
);
```

ROTATE Y

```
void protate_y(
    Pfloat     angle,         /* rotation angle               */
    Pint       *err_ind,      /* OUT error indicator          */
    Pmatrix3   result_tran    /* OUT transformation matrix    */
);
```

ROTATE Z

```
void protate_z(
    Pfloat     angle,         /* rotation angle               */
    Pint       *err_ind,      /* OUT error indicator          */
    Pmatrix3   result_tran    /* OUT transformation matrix    */
);
```

ROTATE

```
void protate(
    Pfloat    angle,         /* rotation angle               */
    Pint      *err_ind,      /* OUT error indicator          */
    Pmatrix   result_tran    /* OUT transformation matrix    */
);
```

COMPOSE MATRIX 3

```
void pcompose_matrix3(
    Pmatrix3   tran_a,       /* transformation matrix a   */
    Pmatrix3   tran_b,       /* transformation matrix b   */
    Pint       *err_ind,     /* OUT error indicator       */
    Pmatrix3   result_tran   /* OUT result matrix         */
);
```

COMPOSE MATRIX

```
void pcompose_matrix(
    Pmatrix   tran_a,       /* transformation matrix a   */
    Pmatrix   tran_b,       /* transformation matrix b   */
    Pint      *err_ind,     /* OUT error indicator       */
    Pmatrix   result_tran   /* OUT result matrix         */
);
```

TRANSFORM POINT 3

```
void ptran_point3(
    const Ppoint3   *point,     /* point                    */
    Pmatrix3        tran,       /* transformation matrix    */
    Pint            *err_ind,   /* OUT error indicator      */
    Ppoint3         *result     /* OUT transformed point    */
);
```

TRANSFORM POINT

```
void ptran_point(
    const Ppoint   *point,     /* point                    */
    Pmatrix        tran,       /* transformation matrix    */
    Pint           *err_ind,   /* OUT error indicator      */
    Ppoint         *result     /* OUT transformed point    */
);
```

BUILD TRANSFORMATION MATRIX 3

```
void pbuild_tran_matrix3(
    const Ppoint3   *point,       /* fixed point                */
    const Pvec3     *shift_vec,   /* shift vector               */
    Pfloat          x_angle,      /* rotation angle X           */
    Pfloat          y_angle,      /* rotation angle Y           */
    Pfloat          z_angle,      /* rotation angle Z           */
    const Pvec3     *scale_vec,   /* scale vector               */
    Pint            *err_ind,     /* OUT error indicator        */
    Pmatrix3        result_tran   /* OUT transformation matrix  */
);
```

BUILD TRANSFORMATION MATRIX

```
void pbuild_tran_matrix(
     const Ppoint   *point,       /* fixed point                  */
     const Pvec     *shift_vec,   /* shift vector                 */
     Pfloat         angle,        /* rotation angle               */
     const Pvec     *scale_vec,   /* scale vector                 */
     Pint           *err_ind,     /* OUT error indicator          */
     Pmatrix        result_tran   /* OUT transformation matrix    */
);
```

COMPOSE TRANSFORMATION MATRIX 3

```
void pcompose_tran_matrix3(
     Pmatrix3       tran,         /* transformation matrix        */
     const Ppoint3  *point,       /* fixed point                  */
     const Pvec3    *shift_vec,   /* shift vector                 */
     Pfloat         x_angle,      /* rotation angle X             */
     Pfloat         y_angle,      /* rotation angle Y             */
     Pfloat         z_angle,      /* rotation angle Z             */
     const Pvec3    *scale_vec,   /* scale vector                 */
     Pint           *err_ind,     /* OUT error indicator          */
     Pmatrix3       result_tran   /* OUT transformation matrix    */
);
```

COMPOSE TRANSFORMATION MATRIX

```
void pcompose_tran_matrix(
     Pmatrix        tran,         /* transformation matrix        */
     const Ppoint   *point,       /* fixed point                  */
     const Pvec     *shift_vec,   /* shift vector                 */
     Pfloat         angle,        /* rotation angle               */
     const Pvec     *scale_vec,   /* scale vector                 */
     Pint           *err_ind,     /* OUT error indicator          */
     Pmatrix        result_tran   /* OUT transformation matrix    */
);
```

7.5.5 Utility functions to support viewing

EVALUATE VIEW ORIENTATION MATRIX 3

```
void peval_view_ori_matrix3(
    const Ppoint3   *view_ref_point,   /* view reference point          */
    const Pvec3     *view_norm_vec,    /* view plane normal             */
    const Pvec3     *view_up_vec,      /* view up vector                */
    Pint            *err_ind,          /* OUT error indicator           */
    Pmatrix3        result_tran        /* OUT view orientation matrix   */
);
```

EVALUATE VIEW ORIENTATION MATRIX

```
void peval_view_ori_matrix(
    const Ppoint    *view_ref_point,   /* view reference point          */
    const Pvec      *view_up_vec,      /* view up vector                */
    Pint            *err_ind,          /* OUT error indicator           */
    Pmatrix         result_tran        /* OUT view orientation matrix   */
);
```

EVALUATE VIEW MAPPING MATRIX 3

```
void peval_view_map_matrix3(
    const Pview_map3   *mapping,       /* view mapping                  */
    Pint               *err_ind,       /* OUT error indicator           */
    Pmatrix3           result_tran     /* OUT view mapping matrix       */
);
```

EVALUATE VIEW MAPPING MATRIX

```
void peval_view_map_matrix(
    const Pview_map    *mapping,       /* view mapping                  */
    Pint               *err_ind,       /* OUT error indicator           */
    Pmatrix            result_tran     /* OUT view mapping matrix       */
);
```

7.6 Structure content functions

OPEN STRUCTURE

```
void popen_struct(
    Pint   struct_id   /* structure identifier   */
);
```

CLOSE STRUCTURE

```
void pclose_struct(
     void
);
```

EXECUTE STRUCTURE

```
void pexec_struct(
    Pint   struct_id   /* structure identifier   */
);
```

LABEL

```
void plabel(
    Pint   label_id   /* label identifier   */
);
```

APPLICATION DATA

```
void pappl_data(
    const Pdata   *data   /* application data   */
);
```

GENERALIZED STRUCTURE ELEMENT

```
void pgse(
    Pint              id,         /* gse identifier    */
    const Pgse_data   *gse_data   /* gse data record   */
);
```

SET EDIT MODE

```
void pset_edit_mode(
    Pedit_mode   edit_mode   /* edit mode   */
);
```

COPY ALL ELEMENTS FROM STRUCTURE

```
void pcopy_all_elems_struct(
    Pint   struct_id   /* structure identifier   */
);
```

SET ELEMENT POINTER

```
void pset_elem_ptr(
    Pint   elem_ptr_value   /* element pointer value   */
);
```

OFFSET ELEMENT POINTER

```
void poffset_elem_ptr(
    Pint   elem_ptr_offset   /* element pointer offset   */
);
```

SET ELEMENT POINTER AT LABEL

```
void pset_elem_ptr_label(
    Pint   label_id   /* label identifier   */
);
```

DELETE ELEMENT

```
void pdel_elem(
    void
);
```

DELETE ELEMENT RANGE

```
void pdel_elem_range(
    Pint   elem_ptr1_value,   /* element pointer 1 value   */
    Pint   elem_ptr2_value    /* element pointer 2 value   */
);
```

DELETE ELEMENTS BETWEEN LABELS

```
void pdel_elems_labels(
    Pint   label1_id,   /* label 1 identifier   */
    Pint   label2_id    /* label 2 identifier   */
);
```

EMPTY STRUCTURE

```
void pempty_struct(
    Pint   struct_id   /* structure id   */
);
```

7.7 Structure manipulation functions

DELETE STRUCTURE

```
void pdel_struct(
    Pint   struct_id   /* structure identifier   */
);
```

DELETE STRUCTURE NETWORK

```
void pdel_struct_net(
    Pint        struct_id,   /* structure identifier      */
    Pref_flag   ref_flag     /* reference handling flag   */
);
```

DELETE ALL STRUCTURES

```
void pdel_all_structs(
      void
);
```

CHANGE STRUCTURE IDENTIFIER

```
void pchange_struct_id(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);
```

CHANGE STRUCTURE REFERENCES

```
void pchange_struct_refs(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);
```

CHANGE STRUCTURE IDENTIFIER AND REFERENCES

```
void pchange_struct_id_refs(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);
```

7.8 Structure display functions

POST STRUCTURE

```
void ppost_struct(
    Pint     ws_id,       /* workstation identifier   */
    Pint     struct_id,   /* structure identifier     */
    Pfloat   pri          /* priority                 */
);
```

UNPOST STRUCTURE

```
void punpost_struct(
    Pint   ws_id,      /* workstation identifier   */
    Pint   struct_id   /* structure identifier     */
);
```

UNPOST ALL STRUCTURES

```
void punpost_all_structs(
    Pint   ws_id   /* workstation identifier   */
);
```

7.9 Structure archiving functions

OPEN ARCHIVE FILE

```
void popen_ar_file(
    Pint          archive_id,     /* archive identifier   */
    const char    *archive_file   /* archive file name    */
);
```

CLOSE ARCHIVE FILE

```
void pclose_ar_file(
    Pint   archive_id   /* archive identifier   */
);
```

ARCHIVE STRUCTURES

```
void par_structs(
    Pint              archive_id,   /* archive identifier             */
    const Pint_list   *struct_ids   /* list of structure identifiers  */
);
```

ARCHIVE STRUCTURE NETWORKS

```
void par_struct_nets(
    Pint              archive_id,   /* archive identifier             */
    const Pint_list   *struct_ids   /* list of structure identifiers  */
);
```

ARCHIVE ALL STRUCTURES

```
void par_all_structs(
    Pint   archive_id   /* archive identifier   */
);
```

SET CONFLICT RESOLUTION

```
void pset_conf_res(
    Pconf_res   archival_res,   /* archival conflict resolution    */
    Pconf_res   retrieval_res   /* retrieval conflict resolution   */
);
```

RETRIEVE STRUCTURE IDENTIFIERS

```
void pret_struct_ids(
    Pint        archive_id,             /* archive identifier          */
    Pint        num_elems_appl_list,    /* # elems in appl list        */
    Pint        start_ind,              /* starting index              */
    Pint_list   *ids,                   /* OUT list of structure ids   */
    Pint        *num_elems_impl_list    /* OUT # elems in impl. list   */
);
```

RETRIEVE STRUCTURES

```
void pret_structs(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);
```

RETRIEVE STRUCTURE NETWORKS

```
void pret_struct_nets(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);
```

RETRIEVE ALL STRUCTURES

```
void pret_all_structs(
    Pint   archive_id   /* archive identifier   */
);
```

RETRIEVE PATHS TO ANCESTORS

```
void pret_paths_ances(
    Pint                  ar_id,       /* archive identifier        */
    Pint                  struct_id,   /* structure identifier      */
    Ppath_order           order,       /* path order                */
    Pint                  depth,       /* path depth                */
    Pstore                store,       /* handle to Store object    */
    Pelem_ref_list_list   **paths      /* OUT structure path list   */
);
```

If an error is detected by this function, `*paths` shall be set to NULL.

NOTE — The memory referenced by `*paths` is managed by `store`.

RETRIEVE PATHS TO DESCENDANTS

```
void pret_paths_descs(
    Pint                  ar_id,       /* archive identifier        */
    Pint                  struct_id,   /* structure identifier      */
    Ppath_order           order,       /* path order                */
    Pint                  depth,       /* path depth                */
    Pstore                store,       /* handle to Store object    */
    Pelem_ref_list_list   **paths      /* OUT structure path list   */
);
```

If an error is detected by this function, `*paths` shall be set to NULL.

NOTE — The memory referenced by `*paths` is managed by `store`.

DELETE STRUCTURES FROM ARCHIVE

```
void pdel_structs_ar(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);
```

DELETE STRUCTURE NETWORKS FROM ARCHIVE

```
void pdel_struct_nets_ar(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);
```

DELETE ALL STRUCTURES FROM ARCHIVE

```
void pdel_all_structs_ar(
    Pint   archive_id   /* archive identifier */
);
```

7.10 Input functions

7.10.1 Pick identifier and filter

SET PICK IDENTIFIER

```
void pset_pick_id(
    Pint   pick_id   /* pick identifier   */
);
```

SET PICK FILTER

```
void pset_pick_filter(
    Pint            ws_id,      /* workstation identifier   */
    Pint            pick_num,   /* pick device number       */
    const Pfilter   *filter     /* pick filter              */
);
```

7.10.2 Initialization of input devices

INITIALIZE LOCATOR 3

```
void pinit_loc3(
    Pint              ws_id,          /* workstation identifier     */
    Pint              loc_num,        /* locator device number      */
    Pint              init_view_ind,  /* initial view index         */
    const Ppoint3     *init_loc_pos,  /* initial locator position   */
    Pint              pet,            /* prompt and echo type       */
    const Plimit3     *echo_vol,      /* echo volume                */
    const Ploc_data3  *loc_data       /* data record                */
);
```

INITIALIZE LOCATOR

```
void pinit_loc(
    Pint              ws_id,          /* workstation identifier     */
    Pint              loc_num,        /* locator device number      */
    Pint              init_view_ind,  /* initial view index         */
    const Ppoint      *init_loc_pos,  /* initial locator position   */
    Pint              pet,            /* prompt and echo type       */
    const Plimit      *echo_area,     /* echo area                  */
    const Ploc_data   *loc_data       /* data record                */
);
```

INITIALIZE STROKE 3

```
void pinit_stroke3(
    Pint                  ws_id,          /* workstation identifier   */
    Pint                  stroke_num,     /* stroke device number     */
    Pint                  init_view_ind,  /* initial view index       */
    const Ppoint_list3    *init_stroke,   /* initial stroke           */
    Pint                  pet,            /* prompt and echo type     */
    const Plimit3         *echo_vol,      /* echo volume              */
    const Pstroke_data3   *stroke_data    /* data record              */
);
```

INITIALIZE STROKE

```
void pinit_stroke(
    Pint                  ws_id,          /* workstation identifier   */
    Pint                  stroke_num,     /* stroke device number     */
    Pint                  init_view_ind,  /* initial view index       */
    const Ppoint_list     *init_stroke,   /* initial stroke           */
    Pint                  pet,            /* prompt and echo type     */
    const Plimit          *echo_area,     /* echo area                */
    const Pstroke_data    *stroke_data    /* data record              */
);
```

INITIALIZE VALUATOR 3

```
void pinit_val3(
    Pint              ws_id,        /* workstation identifier   */
    Pint              val_num,      /* valuator device number   */
    Pfloat            init_value,   /* initial value            */
    Pint              pet,          /* prompt and echo type     */
    const Plimit3     *echo_vol,    /* echo volume              */
    const Pval_data3  *val_data     /* data record              */
);
```

INITIALIZE VALUATOR

```
void pinit_val(
    Pint              ws_id,        /* workstation identifier   */
    Pint              val_num,      /* valuator device number   */
    Pfloat            init_value,   /* initial value            */
    Pint              pet,          /* prompt and echo type     */
    const Plimit      *echo_area,   /* echo area                */
    const Pval_data   *val_data     /* data record              */
);
```

INITIALIZE CHOICE 3

```
void pinit_choice3(
    Pint                  ws_id,          /* workstation identifier   */
    Pint                  choice_num,     /* choice device number     */
    Pin_status            init_status,    /* initial choice status    */
    Pint                  init_choice,    /* initial choice           */
    Pint                  pet,            /* prompt and echo type     */
    const Plimit3         *echo_vol,      /* echo volume              */
    const Pchoice_data3   *choice_data    /* data record              */
);
```

INITIALIZE CHOICE

```
void pinit_choice(
    Pint                  ws_id,          /* workstation identifier   */
    Pint                  choice_num,     /* choice device number     */
    Pin_status            init_status,    /* initial choice status    */
    Pint                  init_choice,    /* initial choice           */
    Pint                  pet,            /* prompt and echo type     */
    const Plimit          *echo_area,     /* echo area                */
    const Pchoice_data    *choice_data    /* data record              */
);
```

INITIALIZE PICK 3

```
void pinit_pick3(
    Pint                ws_id,          /* workstation identifier   */
    Pint                pick_num,       /* pick device number       */
    Pin_status          init_status,    /* initial pick status      */
    const Ppick_path    *init_pick,     /* initial pick path        */
    Pint                pet,            /* prompt and echo type     */
    const Plimit3       *echo_vol,      /* echo volume              */
    const Ppick_data3   *pick_data,     /* data record              */
    Ppath_order         order           /* pick path order          */
);
```

INITIALIZE PICK

```
void pinit_pick(
    Pint                ws_id,          /* workstation identifier   */
    Pint                pick_num,       /* pick device number       */
    Pin_status          init_status,    /* initial pick status      */
    const Ppick_path    *init_pick,     /* initial pick path        */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Ppick_data    *pick_data,     /* data record              */
    Ppath_order         order           /* pick path order          */
);
```

INITIALIZE STRING 3

```
void pinit_string3(
    Pint                ws_id,          /* workstation identifier   */
    Pint                string_num,     /* string device number     */
    const char          *init_string,   /* initial string           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit3       *echo_vol,      /* echo volume              */
    const Pstring_data3 *string_data    /* data record              */
);
```

INITIALIZE STRING

```
void pinit_string(
    Pint                ws_id,          /* workstation identifier   */
    Pint                string_num,     /* string device number     */
    const char          *init_string,   /* initial string           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Pstring_data  *string_data    /* data record              */
);
```

7.10.3 Setting the mode of input devices

SET LOCATOR MODE

```
void pset_loc_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           loc_num,      /* locator device number    */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);
```

SET STROKE MODE

```
void pset_stroke_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           stroke_num,   /* stroke device number     */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);
```

SET VALUATOR MODE

```
void pset_val_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           val_num,      /* valuator device number   */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);
```

SET CHOICE MODE

```
void pset_choice_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           choice_num,   /* choice device number     */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);
```

SET PICK MODE

```
void pset_pick_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           pick_num,     /* pick device number       */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);
```

SET STRING MODE

```
void pset_string_mode(
    Pint          ws_id,        /* workstation identifier   */
    Pint          string_num,   /* string device number     */
    Pop_mode      op_mode,      /* operating mode           */
    Pecho_switch  echo_switch   /* echo switch              */
);
```

7.10.4 Request input functions

REQUEST LOCATOR 3

```
void preq_loc3(
    Pint          ws_id,        /* workstation identifier   */
    Pint          loc_num,      /* locator device number    */
    Pin_status    *in_status,   /* OUT input status         */
    Pint          *view_ind,    /* OUT view index           */
    Ppoint3       *loc_pos      /* OUT locator position     */
);
```

REQUEST LOCATOR

```
void preq_loc(
    Pint          ws_id,        /* workstation identifier   */
    Pint          loc_num,      /* locator device number    */
    Pin_status    *in_status,   /* OUT input status         */
    Pint          *view_ind,    /* OUT view index           */
    Ppoint        *loc_pos      /* OUT locator position     */
);
```

REQUEST STROKE 3

```
void preq_stroke3(
    Pint          ws_id,        /* workstation identifier   */
    Pint          stroke_num,   /* stroke device number     */
    Pin_status    *in_status,   /* OUT input status         */
    Pint          *view_ind,    /* OUT view index           */
    Ppoint_list3  *stroke       /* OUT stroke               */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE 3; the maximum size of a stroke point list supported by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA 3.

REQUEST STROKE

```
void preq_stroke(
     Pint          ws_id,        /* workstation identifier   */
     Pint          stroke_num,   /* stroke device number     */
     Pin_status    *in_status,   /* OUT input status         */
     Pint          *view_ind,    /* OUT view index           */
     Ppoint_list   *stroke       /* OUT stroke               */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE; the maximum size of a stroke point list supported by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA.

REQUEST VALUATOR

```
void preq_val(
     Pint          ws_id,        /* workstation identifier   */
     Pint          val_num,      /* valuator device number   */
     Pin_status    *in_status,   /* OUT input status         */
     Pfloat        *value        /* OUT valuator value       */
);
```

REQUEST CHOICE

```
void preq_choice(
     Pint          ws_id,        /* workstation identifier   */
     Pint          choice_num,   /* choice device number     */
     Pin_status    *in_status,   /* OUT choice status        */
     Pint          *choice       /* OUT requested choice     */
);
```

REQUEST PICK

```
void preq_pick(
     Pint          ws_id,        /* workstation identifier         */
     Pint          pick_num,     /* pick device number             */
     Pint          depth,        /* max. depth of returned path    */
     Pin_status    *in_status,   /* OUT pick status                */
     Ppick_path    *pick         /* OUT requested pick path        */
);
```

REQUEST STRING

```
void preq_string(
    Pint         ws_id,         /* workstation identifier   */
    Pint         string_num,    /* string device number     */
    Pin_status   *in_status,    /* OUT input status         */
    char         *string        /* OUT requested string     */
);
```

The application shall allocate the memory for the character string returned by this function. The maximum size of the returned character string is specified by INITIALIZE STRING or INITIALIZE STRING 3; the maximum size of a character string supported by the implementation is returned by INQUIRE DEFAULT STRING DEVICE DATA or INQUIRE DEFAULT STRING DEVICE DATA 3.

7.10.5 Sample input functions

SAMPLE LOCATOR 3

```
void psample_loc3(
    Pint      ws_id,        /* workstation identifier   */
    Pint      loc_num,      /* locator device number    */
    Pint      *view_ind,    /* OUT view index           */
    Ppoint3   *loc_pos      /* OUT locator position     */
);
```

SAMPLE LOCATOR

```
void psample_loc(
    Pint     ws_id,         /* workstation identifier   */
    Pint     loc_num,       /* locator device number    */
    Pint     *view_ind,     /* OUT view index           */
    Ppoint   *loc_pos       /* OUT locator position     */
);
```

SAMPLE STROKE 3

```
void psample_stroke3(
    Pint           ws_id,         /* workstation identifier   */
    Pint           stroke_num,    /* stroke device number     */
    Pint           *view_ind,     /* OUT view index           */
    Ppoint_list3   *stroke        /* OUT stroke               */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE 3; the maximum size of a stroke point list supported by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA 3.

SAMPLE STROKE

```
void psample_stroke(
    Pint          ws_id,        /* workstation identifier   */
    Pint          stroke_num,   /* stroke device number     */
    Pint          *view_ind,    /* OUT view index           */
    Ppoint_list   *stroke       /* OUT stroke               */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE; the maximum size of a stroke point list supported by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA.

SAMPLE VALUATOR

```
void psample_val(
    Pint     ws_id,     /* workstation identifier   */
    Pint     val_num,   /* valuator device number   */
    Pfloat   *value     /* OUT valuator value       */
);
```

SAMPLE CHOICE

```
void psample_choice(
    Pint         ws_id,              /* workstation identifier    */
    Pint         choice_num,         /* choice device number      */
    Pin_status   *choice_in_status,  /* OUT choice input status   */
    Pint         *choice             /* OUT choice                */
);
```

SAMPLE PICK

```
void psample_pick(
    Pint         ws_id,              /* workstation identifier        */
    Pint         pick_num,           /* pick device number            */
    Pint         depth,              /* max. depth of returned path   */
    Pin_status   *pick_in_status,    /* OUT pick input status         */
    Ppick_path   *pick               /* OUT pick path                 */
);
```

SAMPLE STRING

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

The application shall allocate the memory for the character string returned by this function. The maximum size of the returned character string is specified by INITIALIZE STRING or INITIALIZE STRING 3; the maximum size of a character string supported by the implementation is returned by INQUIRE DEFAULT STRING DEVICE DATA or INQUIRE DEFAULT STRING DEVICE DATA 3.

7.10.6 Event input functions

AWAIT EVENT

```
void pawait_event(
    Pfloat      timeout,      /* timeout (seconds)                */
    Pint        *ws_id,       /* OUT workstation identifier       */
    Pin_class   *dev_class,   /* OUT device class                 */
    Pint        *in_num       /* OUT logical input device number  */
);
```

FLUSH DEVICE EVENTS

```
void pflush_events(
    Pint        ws_id,        /* workstation identifier        */
    Pin_class   dev_class,    /* device class                  */
    Pint        in_num        /* logical input device number   */
);
```

GET LOCATOR 3

```
void pget_loc3(
    Pint      *view_ind,   /* OUT view index          */
    Ppoint3   *loc_pos     /* OUT locator position    */
);
```

GET LOCATOR

```
void pget_loc(
    Pint      *view_ind,   /* OUT view index          */
    Ppoint    *loc_pos     /* OUT locator position    */
);
```

GET STROKE 3

```
void pget_stroke3(
    Pint           *view_ind,   /* OUT view index   */
    Ppoint_list3   *stroke      /* OUT stroke       */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE 3; the maximum size of a stroke point list supported

by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA 3.

GET STROKE

```
void pget_stroke(
    Pint          *view_ind,   /* OUT view index   */
    Ppoint_list   *stroke      /* OUT stroke       */
);
```

The application shall allocate the memory for the point list returned by this function. The maximum size of the returned stroke point list is specified by INITIALIZE STROKE; the maximum size of a stroke point list supported by the implementation is returned by INQUIRE DEFAULT STROKE DEVICE DATA.

GET VALUATOR

```
void pget_val(
    Pfloat   *value   /* OUT valuator value   */
);
```

GET CHOICE

```
void pget_choice(
    Pin_status   *in_status,   /* OUT choice status   */
    Pint         *choice       /* OUT choice          */
);
```

GET PICK

```
void pget_pick(
    Pint         depth,        /* max. depth of returned path   */
    Pin_status   *in_status,   /* OUT pick status               */
    Ppick_path   *pick         /* OUT pick path                 */
);
```

GET STRING

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

The application shall allocate the memory for the character string returned by this function. The maximum size of the returned character string is specified by INITIALIZE STRING or INITIALIZE STRING 3; the maximum size of a character string supported by the implementation is returned by INQUIRE DEFAULT STRING DEVICE DATA or INQUIRE DEFAULT STRING DEVICE DATA 3.

7.11 Metafile functions

WRITE ITEM TO METAFILE

```
void pwrite_item(
    Pint                ws_id,              /* workstation identifier    */
    Pint                item_type,          /* item type                 */
    Pint                item_data_length,   /* item data record length   */
    const Pitem_data    *item_data          /* item data record          */
);
```

GET ITEM TYPE FROM METAFILE

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

READ ITEM FROM METAFILE

```
void pread_item(
    Pint         ws_id,                 /* workstation identifier        */
    Pint         max_item_data_length,  /* max item data record length   */
    Pitem_data   *item_data             /* OUT item data record          */
);
```

INTERPRET ITEM

```
void pinterpret_item(
    Pint                type,               /* item type                 */
    Pint                item_data_length,   /* item data record length   */
    const Pitem_data    *item_data          /* item data record          */
);
```

7.12 Inquiry functions

7.12.1 Inquiry functions for operating state values

INQUIRE SYSTEM STATE VALUE

```
void pinq_sys_st(
    Psys_st   *sys_st   /* OUT the system state   */
);
```

INQUIRE WORKSTATION STATE VALUE

```
void pinq_ws_st(
    Pws_st   *ws_st   /* OUT workstation state   */
);
```

INQUIRE STRUCTURE STATE VALUE

```
void pinq_struct_st(
    Pstruct_st   *struct_st   /* OUT structure state   */
);
```

INQUIRE ARCHIVE STATE VALUE

```
void pinq_ar_st(
    Par_st   *ar_st   /* OUT archive state   */
);
```

7.12.2 Inquiry functions for PHIGS description table

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES

```
void pinq_list_avail_ws_types(
    Pint        num_elems_appl_list,   /* # elems in appl list       */
    Pint        start_ind,             /* starting index             */
    Pint        *err_ind,              /* OUT error indicator        */
    Pint_list   *types,                /* OUT list of ws types       */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list   */
);
```

INQUIRE PHIGS FACILITIES

```
void pinq_phigs_facs(
    Pint        num_elems_appl_list,    /* # elems in appl list                    ,
    Pint        start_ind,              /* starting index                          ,
    Pint        *err_ind,               /* OUT error indicator                     ,
    Pint        *max_open_ws,           /* OUT max. # simult. open ws              ,
    Pint        *max_open_ar,           /* OUT max. # simult. open archive files   ,
    Pint        *num_avail_names,       /* OUT # available names for name sets     ,
    Pint_list   *char_sets,             /* OUT list of character sets              ,
    Pint        *num_elems_impl_list,   /* OUT # elems in impl list                ,
    Pint        *iss_norm_max,          /* OUT ISS max. length of normal filter list ,
    Pint        *iss_inv_max            /* OUT ISS max. length of inverted filter list
);
```

INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES

```
void pinq_gse_facs(
    Pint                num_elems_appl_list,   /* # elems in appl list                  *,
    Pint                start_ind,             /* starting index                        *,
    Pint                *err_ind,              /* OUT error indicator                   *,
    Pgse_id_dep_list    *gse,                  /* OUT list of GSE ids and dependencies  *,
    Pint                *num_elems_impl_list   /* OUT # elems in impl list              *,
);
```

INQUIRE MODELLING CLIPPING FACILITIES

```
void pinq_model_clip_facs(
    Pint        num_elems_appl_list,   /* # elems in appl list          */
    Pint        start_ind,             /* starting index                */
    Pint        *err_ind,              /* OUT error indicator           */
    Pint        *num_planes,           /* OUT number of distinct planes */
    Pint_list   *ops,                  /* OUT list of operators         */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list      */
);
```

7.12.3 Inquiry functions for PHIGS state list

INQUIRE EDIT MODE

```
void pinq_edit_mode(
    Pint         *err_ind,    /* OUT error indicator   */
    Pedit_mode   *edit_mode   /* OUT edit mode         */
);
```

INQUIRE SET OF OPEN WORKSTATIONS

```
void pinq_open_wss(
    Pint         num_elems_appl_list,   /* # elems in appl list        */
    Pint         start_ind,             /* starting index              */
    Pint         *err_ind,              /* OUT error indicator         */
    Pint_list    *open_ws_ids,          /* OUT list of ws ids          */
    Pint         *num_elems_impl_list   /* OUT # elems in impl list    */
);
```

INQUIRE STRUCTURE IDENTIFIERS

```
void pinq_struct_ids(
    Pint         num_elems_appl_list,   /* # elems in appl list        */
    Pint         start_ind,             /* starting index              */
    Pint         *err_ind,              /* OUT error indicator         */
    Pint_list    *struct_ids,           /* OUT list of structure ids   */
    Pint         *num_elems_impl_list   /* OUT # elems in impl list    */
);
```

INQUIRE ARCHIVE FILES

```
void pinq_ar_files(
    Pstore       store,       /* handle to Store object                 */
    Pint         *err_ind,    /* OUT error indicator                    */
    Par_file_list **ar_files  /* OUT list of archive file names and ids */
);
```

NOTE — The memory referenced by `*ar_files` is managed by `store`.

INQUIRE CONFLICT RESOLUTION

```
void pinq_conf_res(
    Pint         *err_ind,       /* OUT error indicator       */
    Pconf_res    *archive_res,   /* OUT archival resolution   */
    Pconf_res    *retrieve_res   /* OUT retrieval resolution  */
);
```

INQUIRE ALL CONFLICTING STRUCTURES

```
void pinq_all_conf_structs(
    Pint        ar_id,                  /* archive identifier                    */
    Pint        num_elems_appl_list,    /* # elems in appl list                  */
    Pint        start_ind,              /* starting index                        */
    Pint        *err_ind,               /* OUT error indicator                   */
    Pint_list   *ids,                   /* OUT list of conflicting structure ids */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list              */
);
```

INQUIRE CONFLICTING STRUCTURES IN NETWORK

```
void pinq_conf_structs_net(
    Pint                ar_id,                  /* archive identifier              */
    Pint                struct_id,              /* structure identifier            */
    Pstruct_net_source  source,                 /* structure network source        */
    Pint                num_elems_appl_list,    /* # elems in appl list            */
    Pint                start_ind,              /* starting index                  */
    Pint                *err_ind,               /* OUT error indicator             */
    Pint_list           *ids,                   /* OUT conflicting struct id list  */
    Pint                *num_elems_impl_list    /* OUT # elems in impl list        */
);
```

INQUIRE MORE SIMULTANEOUS EVENTS

```
void pinq_more_simult_events(
    Pint                *err_ind,       /* OUT error indicator       */
    Pmore_simult_events *simult_events  /* OUT simultaneous events   */
);
```

7.12.4 Inquiry functions for workstation state list

INQUIRE WORKSTATION CONNECTION AND TYPE

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

NOTE — The memory referenced by `*conn_id` is managed by `store`.

INQUIRE LIST OF VIEW INDICES

```
void pinq_list_view_inds(
    Pint        ws_id,                  /* workstation identifier     */
    Pint        num_elems_appl_list,    /* # elems in appl list       */
    Pint        start_ind,              /* starting index             */
    Pint        *err_ind,               /* OUT error indicator        */
    Pint_list   *view_inds,             /* OUT list of view indices   */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list   */
);
```

INQUIRE VIEW REPRESENTATION

```
void pinq_view_rep(
    Pint         ws_id,       /* workstation identifier              */
    Pint         view_ind,    /* view index                          */
    Pint         *err_ind,    /* OUT error indicator                 */
    Pupd_st      *upd_st,     /* OUT transformation update state     */
    Pview_rep3   *cur_view,   /* OUT current view representation     */
    Pview_rep3   *req_view    /* OUT requested view representation   */
);
```

INQUIRE HLHSR MODE

```
void pinq_hlhsr_mode(
    Pint      ws_id,        /* workstation identifier     */
    Pint      *err_ind,     /* OUT error indicator        */
    Pupd_st   *upd_st,      /* OUT HLHSR update state     */
    Pint      *cur_mode,    /* OUT current HLHSR mode     */
    Pint      *req_mode     /* OUT requested HLHSR mode   */
);
```

INQUIRE POSTED STRUCTURES

```
void pinq_posted_structs(
    Pint                  ws_id,                /* workstation identifier          */
    Pint                  num_elems_appl_list,  /* # elems in appl list            */
    Pint                  start_ind,            /* starting index                  */
    Pint                  *err_ind,             /* OUT error indicator             */
    Pposted_struct_list   *struct_ids,          /* OUT list of posted structures   */
    Pint                  *num_elems_impl_list  /* OUT # elems in impl list        */
);
```

INQUIRE DISPLAY UPDATE STATE

```
void pinq_disp_upd_st(
    Pint              ws_id,              /* workstation identifier              */
    Pint              *err_ind,           /* OUT error indicator                 */
    Pdefer_mode       *def_mode,          /* OUT deferral mode                   */
    Pmod_mode         *mod_mode,          /* OUT modification mode               */
    Pdisp_surf_empty  *disp_surf_empty,   /* OUT display surface empty           */
    Pvisual_st        *vis_st             /* OUT state of visual representation  */
);
```

INQUIRE LIST OF POLYLINE INDICES

```
void pinq_list_line_inds(
    Pint        ws_id,                /* workstation identifier                 */
    Pint        num_elems_appl_list,  /* # elems in appl list                   */
    Pint        start_ind,            /* starting index                         */
    Pint        *err_ind,             /* OUT error indicator                    */
    Pint_list   *def_line_ind,        /* OUT list of defined polyline indices   */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list               */
);
```

INQUIRE POLYLINE REPRESENTATION

```
void pinq_line_rep(
    Pint           ws_id,      /* workstation identifier       */
    Pint           index,      /* polyline index               */
    Pinq_type      type,       /* type of returned value       */
    Pint           *err_ind,   /* OUT error indicator          */
    Pline_bundle   *line_rep   /* OUT polyline representation  */
);
```

INQUIRE LIST OF POLYMARKER INDICES

```
void pinq_list_marker_inds(
    Pint        ws_id,                /* workstation identifier                 */
    Pint        num_elems_appl_list,  /* # elems in appl list                   */
    Pint        start_ind,            /* starting index                         */
    Pint        *err_ind,             /* OUT error indicator                    */
    Pint_list   *def_marker_ind,      /* OUT list of defined polymarker indices */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list               */
);
```

INQUIRE POLYMARKER REPRESENTATION

```
void pinq_marker_rep(
     Pint             ws_id,        /* workstation identifier          */
     Pint             index,        /* polymarker index                */
     Pinq_type        type,         /* type of returned value          */
     Pint             *err_ind,     /* OUT error indicator             */
     Pmarker_bundle   *marker_rep   /* OUT polymarker representation   */
);
```

INQUIRE LIST OF TEXT INDICES

```
void pinq_list_text_inds(
     Pint        ws_id,                /* workstation identifier            */
     Pint        num_elems_appl_list,  /* # elems in appl list              */
     Pint        start_ind,            /* starting index                    */
     Pint        *err_ind,             /* OUT error indicator               */
     Pint_list   *def_text_ind,        /* OUT list of defined text indices  */
     Pint        *num_elems_impl_list  /* OUT # elems in impl list          */
);
```

INQUIRE TEXT REPRESENTATION

```
void pinq_text_rep(
     Pint           ws_id,      /* workstation identifier   */
     Pint           index,      /* text index               */
     Pinq_type      type,       /* type of returned value   */
     Pint           *err_ind,   /* OUT error indicator      */
     Ptext_bundle   *text_rep   /* OUT text representation  */
);
```

INQUIRE LIST OF INTERIOR INDICES

```
void pinq_list_int_inds(
     Pint        ws_id,                /* workstation identifier            */
     Pint        num_elems_appl_list,  /* # elems in appl list              */
     Pint        start_ind,            /* starting index                    */
     Pint        *err_ind,             /* OUT error indicator               */
     Pint_list   *def_int_ind,         /* OUT list of defined interior indices  */
     Pint        *num_elems_impl_list  /* OUT # elems in impl list          */
);
```

INQUIRE INTERIOR REPRESENTATION

```
void pinq_int_rep(
    Pint          ws_id,      /* workstation identifier        */
    Pint          index,      /* interior index                */
    Pinq_type     type,       /* type of returned value        */
    Pint          *err_ind,   /* OUT error indicator           */
    Pint_bundle   *int_rep    /* OUT interior representation   */
);
```

INQUIRE LIST OF EDGE INDICES

```
void pinq_list_edge_inds(
    Pint        ws_id,                /* workstation identifier            */
    Pint        num_elems_appl_list,  /* # elems in appl list              */
    Pint        start_ind,            /* starting index                    */
    Pint        *err_ind,             /* OUT error indicator               */
    Pint_list   *def_edge_ind,        /* OUT list of defined edge indices  */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list          */
);
```

INQUIRE EDGE REPRESENTATION

```
void pinq_edge_rep(
    Pint          ws_id,      /* workstation identifier    */
    Pint          index,      /* edge index                */
    Pinq_type     type,       /* type of returned value    */
    Pint          *err_ind,   /* OUT error indicator       */
    Pedge_bundle  *edge_rep   /* OUT edge representation   */
);
```

INQUIRE LIST OF PATTERN INDICES

```
void pinq_list_pat_inds(
    Pint        ws_id,                /* workstation identifier            */
    Pint        num_elems_appl_list,  /* # elems in appl list              */
    Pint        start_ind,            /* starting index                    */
    Pint        *err_ind,             /* OUT error indicator               */
    Pint_list   *def_pat_ind,         /* OUT list of defined pattern indices */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list          */
);
```

INQUIRE PATTERN REPRESENTATION

```
void pinq_pat_rep(
    Pint        ws_id,      /* workstation identifier       */
    Pint        index,      /* pattern index                */
    Pinq_type   type,       /* type of returned value       */
    Pstore      store,      /* handle to Store object       */
    Pint        *err_ind,   /* OUT error indicator          */
    Ppat_rep    **pat_rep   /* OUT pattern representation   */
);
```

NOTE — The memory referenced by `*pat_rep` is managed by `store`.

INQUIRE COLOUR MODEL

```
void pinq_colr_model(
    Pint   ws_id,       /* workstation identifier    */
    Pint   *err_ind,    /* OUT error indicator       */
    Pint   *model       /* OUT current colour model  */
);
```

INQUIRE LIST OF COLOUR INDICES

```
void pinq_list_colr_inds(
    Pint        ws_id,                  /* workstation identifier       */
    Pint        num_elems_appl_list,    /* # elems in appl list         */
    Pint        start_ind,              /* starting index               */
    Pint        *err_ind,               /* OUT error indicator          */
    Pint_list   *colr_inds,             /* OUT list of colour indices   */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list     */
);
```

INQUIRE COLOUR REPRESENTATION

```
void pinq_colr_rep(
    Pint        ws_id,      /* workstation identifier       */
    Pint        colr_ind,   /* colour index                 */
    Pinq_type   type,       /* type of returned value       */
    Pint        *err_ind,   /* OUT error indicator          */
    Pcolr_rep   *colr_rep   /* OUT colour representation    */
);
```

INQUIRE HIGHLIGHTING FILTER

```
void pinq_highl_filter(
    Pint      ws_id,          /* workstation identifier    */
    Pstore    store,          /* handle to Store object    */
    Pint      *err_ind,       /* OUT error indicator       */
    Pfilter   **highl_filter  /* OUT highlighting filter   */
);
```

NOTE — The memory referenced by `*highl_filter` is managed by `store`.

INQUIRE INVISIBILITY FILTER

```
void pinq_invis_filter(
    Pint      ws_id,          /* workstation identifier    */
    Pstore    store,          /* handle to Store object    */
    Pint      *err_ind,       /* OUT error indicator       */
    Pfilter   **invis_filter  /* OUT invisibility filter   */
);
```

NOTE — The memory referenced by `*invis_filter` is managed by `store`.

INQUIRE WORKSTATION TRANSFORMATION 3

```
void pinq_ws_tran3(
    Pint      ws_id,          /* workstation identifier              */
    Pint      *err_ind,       /* OUT error indicator                 */
    Pupd_st   *upd_st,        /* OUT update state                    */
    Plimit3   *req_win_lim,   /* OUT requested workstation window    */
    Plimit3   *cur_win_lim,   /* OUT current workstation window      */
    Plimit3   *req_vp_lim,    /* OUT requested workstation viewport  */
    Plimit3   *cur_vp_lim     /* OUT current workstation viewport    */
);
```

INQUIRE WORKSTATION TRANSFORMATION

```
void pinq_ws_tran(
    Pint      ws_id,          /* workstation identifier              */
    Pint      *err_ind,       /* OUT error indicator                 */
    Pupd_st   *upd_st,        /* OUT update state                    */
    Plimit    *req_win_lim,   /* OUT requested workstation window    */
    Plimit    *cur_win_lim,   /* OUT current workstation window      */
    Plimit    *req_vp_lim,    /* OUT requested workstation viewport  */
    Plimit    *cur_vp_lim     /* OUT current workstation viewport    */
);
```

INQUIRE LOCATOR DEVICE STATE 3

```
void pinq_loc_st3(
     Pint           ws_id,           /* workstation identifier          */
     Pint           loc_num,         /* locator device number           */
     Pinq_type      type,            /* type of returned value          */
     Pstore         store,           /* handle to Store object          */
     Pint           *err_ind,        /* OUT error indicator             */
     Pop_mode       *op_mode,        /* OUT operating mode              */
     Pecho_switch   *echo_switch,    /* OUT echo switch                 */
     Pint           *init_view_ind,  /* OUT initial view index          */
     Ppoint3        *init_loc_pos,   /* OUT initial locator position    */
     Pint           *prompt_echo,    /* OUT prompt/echo type            */
     Plimit3        *echo_vol,       /* OUT echo volume                 */
     Ploc_data3     **loc_data       /* OUT data record                 */
);
```

NOTE — The memory referenced by `*loc_data` is managed by `store`.

INQUIRE LOCATOR DEVICE STATE

```
void pinq_loc_st(
     Pint           ws_id,           /* workstation identifier          */
     Pint           loc_num,         /* locator device number           */
     Pinq_type      type,            /* type of returned value          */
     Pstore         store,           /* handle to Store object          */
     Pint           *err_ind,        /* OUT error indicator             */
     Pop_mode       *op_mode,        /* OUT operating mode              */
     Pecho_switch   *echo_switch,    /* OUT echo switch                 */
     Pint           *init_view_ind,  /* OUT initial view index          */
     Ppoint         *init_loc_pos,   /* OUT initial locator position    */
     Pint           *prompt_echo,    /* OUT prompt/echo type            */
     Plimit         *echo_area,      /* OUT echo area                   */
     Ploc_data      **loc_data       /* OUT data record                 */
);
```

NOTE — The memory referenced by `*loc_data` is managed by `store`.

INQUIRE STROKE DEVICE STATE 3

```
void pinq_stroke_st3(
     Pint            ws_id,            /* workstation identifier   */
     Pint            stroke_num,       /* stroke device number     */
     Pinq_type       type,             /* type of returned value   */
     Pstore          store,            /* handle to Store object   */
     Pint            *err_ind,         /* OUT error indicator      */
     Pop_mode        *op_mode,         /* OUT operating mode       */
     Pecho_switch    *echo_switch,     /* OUT echo switch          */
     Pint            *init_view_ind,   /* OUT initial view index   */
     Ppoint_list3    **init_stroke,    /* OUT initial stroke       */
     Pint            *prompt_echo,     /* OUT prompt/echo type     */
     Plimit3         *echo_vol,        /* OUT echo volume          */
     Pstroke_data3   **stroke_data     /* OUT data record          */
);
```

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

INQUIRE STROKE DEVICE STATE

```
void pinq_stroke_st(
     Pint            ws_id,            /* workstation identifier   */
     Pint            stroke_num,       /* stoke device number      */
     Pinq_type       type,             /* type of returned value   */
     Pstore          store,            /* handle to Store object   */
     Pint            *err_ind,         /* OUT error indicator      */
     Pop_mode        *op_mode,         /* OUT operating mode       */
     Pecho_switch    *echo_switch,     /* OUT echo switch          */
     Pint            *init_view_ind,   /* OUT initial view index   */
     Ppoint_list     **init_stroke,    /* OUT initial stroke       */
     Pint            *prompt_echo,     /* OUT prompt/echo type     */
     Plimit          *echo_area,       /* OUT echo area            */
     Pstroke_data    **stroke_data     /* OUT data record          */
);
```

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

INQUIRE VALUATOR DEVICE STATE 3

```
void pinq_val_st3(
     Pint            ws_id,            /* workstation identifier   */
     Pint            val_num,          /* valuator device number   */
     Pstore          store,            /* handle to Store object   */
     Pint            *err_ind,         /* OUT error indicator      */
     Pop_mode        *op_mode,         /* OUT operating mode       */
     Pecho_switch    *echo_switch,     /* OUT echo switch          */
     Pfloat          *init_value,      /* OUT initial value        */
     Pint            *prompt_echo,     /* OUT prompt/echo type     */
     Plimit3         *echo_vol,        /* OUT echo volume          */
     Pval_data3      **val_data        /* OUT data record          */
);
```

NOTE — The memory referenced by `*val_data` is managed by `store`.

INQUIRE VALUATOR DEVICE STATE

```
void pinq_val_st(
    Pint          ws_id,          /* workstation identifier   */
    Pint          val_num,        /* valuator device number   */
    Pstore        store,          /* handle to Store object   */
    Pint          *err_ind,       /* OUT error indicator      */
    Pop_mode      *op_mode,       /* OUT operating mode       */
    Pecho_switch  *echo_switch,   /* OUT echo switch          */
    Pfloat        *init_value,    /* OUT initial value        */
    Pint          *prompt_echo,   /* OUT prompt/echo type     */
    Plimit        *echo_area,     /* OUT echo area            */
    Pval_data     **val_data      /* OUT data record          */
);
```

NOTE — The memory referenced by `*val_data` is managed by `store`.

INQUIRE CHOICE DEVICE STATE 3

```
void pinq_choice_st3(
    Pint            ws_id,          /* workstation identifier      */
    Pint            choice_num,     /* choice device number        */
    Pstore          store,          /* handle to Store object      */
    Pint            *err_ind,       /* OUT error indicator         */
    Pop_mode        *op_mode,       /* OUT operating mode          */
    Pecho_switch    *echo_switch,   /* OUT echo switch             */
    Pin_status      *init_status,   /* OUT initial choice status   */
    Pint            *init_choice,   /* OUT initial choice          */
    Pint            *prompt_echo,   /* OUT prompt/echo type        */
    Plimit3         *echo_vol,      /* OUT echo volume             */
    Pchoice_data3   **choice_data   /* OUT data record             */
);
```

NOTE — The memory referenced by `*choice_data` is managed by `store`.

INQUIRE CHOICE DEVICE STATE

```
void pinq_choice_st(
    Pint            ws_id,          /* workstation identifier      */
    Pint            choice_num,     /* choice device number        */
    Pstore          store,          /* handle to Store object      */
    Pint            *err_ind,       /* OUT error indicator         */
    Pop_mode        *op_mode,       /* OUT operating mode          */
    Pecho_switch    *echo_switch,   /* OUT echo switch             */
    Pin_status      *init_status,   /* OUT initial choice status   */
    Pint            *init_choice,   /* OUT initial choice          */
    Pint            *prompt_echo,   /* OUT prompt/echo type        */
    Plimit          *echo_area,     /* OUT echo area               */
    Pchoice_data    **choice_data   /* OUT data record             */
);
```

NOTE — The memory referenced by `*choice_data` is managed by `store`.

INQUIRE PICK DEVICE STATE 3

```
void pinq_pick_st3(
     Pint           ws_id,          /* workstation identifier    */
     Pint           pick_num,       /* pick device number        */
     Pinq_type      type,           /* type of returned value    */
     Pstore         store,          /* handle to Store object    */
     Pint           *err_ind,       /* OUT error indicator       */
     Pop_mode       *op_mode,       /* OUT operating mode        */
     Pecho_switch   *echo_switch,   /* OUT echo switch           */
     Pfilter        **pick_filter,  /* OUT pick filter           */
     Pin_status     *init_status,   /* OUT initial pick status   */
     Ppick_path     **init_pick,    /* OUT initial pick path     */
     Pint           *prompt_echo,   /* OUT prompt/echo type      */
     Plimit3        *echo_vol,      /* OUT echo volume           */
     Ppick_data3    **pick_data,    /* OUT data record           */
     Ppath_order    *order          /* OUT pick path order       */
);
```

NOTE — The memory referenced by `*pick_filter`, `*init_path` and `*pick_data` is managed by `store`.

INQUIRE PICK DEVICE STATE

```
void pinq_pick_st(
     Pint           ws_id,          /* workstation identifier    */
     Pint           pick_num,       /* pick device number        */
     Pinq_type      type,           /* type of returned value    */
     Pstore         store,          /* handle to Store object    */
     Pint           *err_ind,       /* OUT error indicator       */
     Pop_mode       *op_mode,       /* OUT operating mode        */
     Pecho_switch   *echo_switch,   /* OUT echo switch           */
     Pfilter        **pick_filter,  /* OUT pick filter           */
     Pin_status     *init_status,   /* OUT initial pick status   */
     Ppick_path     **init_pick,    /* OUT initial pick path     */
     Pint           *prompt_echo,   /* OUT prompt/echo type      */
     Plimit         *echo_area,     /* OUT echo area             */
     Ppick_data     **pick_data,    /* OUT data record           */
     Ppath_order    *order          /* OUT pick path order       */
);
```

NOTE — The memory referenced by `*pick_filter`, `*init_path` and `*pick_data` is managed by `store`.

INQUIRE STRING DEVICE STATE 3

```
void pinq_string_st3(
    Pint            ws_id,          /* workstation identifier   */
    Pint            string_num,     /* string device number     */
    Pstore          store,          /* handle to Store object   */
    Pint            *err_ind,       /* OUT error indicator      */
    Pop_mode        *op_mode,       /* OUT operating mode       */
    Pecho_switch    *echo_switch,   /* OUT echo switch          */
    char            **init_string,  /* OUT initial string       */
    Pint            *prompt_echo,   /* OUT prompt/echo type     */
    Plimit3         *echo_vol,      /* OUT echo volume          */
    Pstring_data3   **string_data   /* OUT data record          */
);
```

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

INQUIRE STRING DEVICE STATE

```
void pinq_string_st(
    Pint            ws_id,          /* workstation identifier   */
    Pint            string_num,     /* string device number     */
    Pstore          store,          /* handle to Store object   */
    Pint            *err_ind,       /* OUT error indicator      */
    Pop_mode        *op_mode,       /* OUT operating mode       */
    Pecho_switch    *echo_switch,   /* OUT echo switch          */
    char            **init_string,  /* OUT initial string       */
    Pint            *prompt_echo,   /* OUT prompt/echo type     */
    Plimit          *echo_area,     /* OUT echo area            */
    Pstring_data    **string_data   /* OUT data record          */
);
```

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

7.12.5 Inquiry functions for workstation description table

INQUIRE WORKSTATION CATEGORY

```
void pinq_ws_cat(
    Pint        ws_type,    /* workstation type             */
    Pint        *err_ind,   /* OUT error indicator          */
    Pws_cat     *cat        /* OUT workstation category     */
);
```

INQUIRE DISPLAY SPACE SIZE 3

```
void pinq_disp_space_size3(
    Pint                ws_type,    /* workstation type      */
    Pint                *err_ind,   /* OUT error indicator   */
    Pdisp_space_size3   *size       /* OUT display size      */
);
```

INQUIRE DISPLAY SPACE SIZE

```
void pinq_disp_space_size(
    Pint                ws_type,    /* workstation type      */
    Pint                *err_ind,   /* OUT error indicator   */
    Pdisp_space_size    *size       /* OUT display size      */
);
```

The PHIGS abstract function INQUIRE HLHSR FACILITIES has been split into the following two functions.

INQUIRE HLHSR IDENTIFIER FACILITIES

```
void pinq_hlhsr_id_facs(
    Pint        ws_type,                /* workstation type          */
    Pint        num_elems_appl_list,    /* # elems in appl list      */
    Pint        start_ind,              /* starting index            */
    Pint        *err_ind,               /* OUT error indicator       */
    Pint_list   *hlhsr_ids,             /* OUT list of HLHSR ids     */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

INQUIRE HLHSR MODE FACILITIES

```
void pinq_hlhsr_mode_facs(
    Pint        ws_type,                /* workstation type          */
    Pint        num_elems_appl_list,    /* # elems in appl list      */
    Pint        start_ind,              /* starting index            */
    Pint        *err_ind,               /* OUT error indicator       */
    Pint_list   *hlhsr_modes,           /* OUT list of HLHSR modes   */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

INQUIRE VIEW FACILITIES

```
void pinq_view_facs(
    Pint   ws_type,         /* workstation type                      */
    Pint   *err_ind,        /* OUT error indicator                   */
    Pint   *num_view_ind    /* OUT number of predefined view indices */
);
```

INQUIRE PREDEFINED VIEW REPRESENTATION

```
void pinq_pred_view_rep(
    Pint          ws_type,     /* workstation type          */
    Pint          index,       /* predefined view index     */
    Pint          *err_ind,    /* OUT error indicator       */
    Pview_rep3    *view        /* OUT view representation   */
);
```

INQUIRE WORKSTATION CLASSIFICATION

```
void pinq_ws_class(
    Pint         ws_type,     /* workstation type        */
    Pint         *err_ind,    /* OUT error indicator     */
    Pws_class    *ws_class    /* OUT workstation class   */
);
```

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES

```
void pinq_dyns_ws_attrs(
    Pint              ws_type,    /* workstation type          */
    Pint              *err_ind,   /* OUT error indicator       */
    Pdyns_ws_attrs    *attr       /* OUT attributes dynamics   */
);
```

INQUIRE DEFAULT DISPLAY UPDATE STATE

```
void pinq_def_disp_upd_st(
    Pint           ws_type,     /* workstation type        */
    Pint           *err_ind,    /* OUT error indicator     */
    Pdefer_mode    *def_mode,   /* OUT deferral mode       */
    Pmod_mode      *mod_mode    /* OUT modification mode   */
);
```

INQUIRE POLYLINE FACILITIES

```
void pinq_line_facs(
    Pint          ws_type,               /* workstation type          */
    Pint          num_elems_appl_list,   /* # elems in appl list      */
    Pint          start_ind,             /* starting index            */
    Pint          *err_ind,              /* OUT error indicator       */
    Pline_facs    *fac,                  /* OUT polyline facilities   */
    Pint          *num_elems_impl_list   /* OUT # elems in impl list  */
);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION

```
void pinq_pred_line_rep(
    Pint           ws_type,    /* workstation type              */
    Pint           index,      /* predefined index              */
    Pint           *err_ind,   /* OUT error indicator           */
    Pline_bundle   *bundle     /* OUT predefined polyline rep   */
);
```

INQUIRE POLYMARKER FACILITIES

```
void pinq_marker_facs(
    Pint           ws_type,                /* workstation type          */
    Pint           num_elems_appl_list,    /* # elems in appl list      */
    Pint           start_ind,              /* starting index            */
    Pint           *err_ind,               /* OUT error indicator       */
    Pmarker_facs   *fac,                   /* OUT polymarker facilities */
    Pint           *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION

```
void pinq_pred_marker_rep(
    Pint             ws_type,    /* workstation type              */
    Pint             index,      /* predefined index              */
    Pint             *err_ind,   /* OUT error indicator           */
    Pmarker_bundle   *bundle     /* OUT predefined polymarker rep */
);
```

INQUIRE TEXT FACILITIES

```
void pinq_text_facs(
    Pint         ws_type,                /* workstation type          */
    Pint         num_elems_appl_list,    /* # elems in appl list      */
    Pint         start_ind,              /* starting index            */
    Pint         *err_ind,               /* OUT error indicator       */
    Ptext_facs   *fac,                   /* OUT text facilities       */
    Pint         *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

INQUIRE PREDEFINED TEXT REPRESENTATION

```
void pinq_pred_text_rep(
    Pint           ws_type,    /* workstation type          */
    Pint           index,      /* predefined index          */
    Pint           *err_ind,   /* OUT error indicator       */
    Ptext_bundle   *bundle     /* OUT predefined text rep   */
);
```

INQUIRE ANNOTATION FACILITIES

```
void pinq_anno_facs(
    Pint        ws_type,                /* workstation type                          */
    Pint        num_elems_appl_list,    /* # elems in appl list                      */
    Pint        start_ind,              /* starting index                            */
    Pint        *err_ind,               /* OUT error indicator                       */
    Pint_list   *styles,                /* OUT list annotation styles                */
    Pint        *num_elems_impl_list,   /* OUT # elems in impl list                  */
    Pint        *num_anno_char_hts,     /* OUT number of anno. text char. heights    */
    Pfloat      *min_anno_char_ht,      /* OUT minimum anno. text char. height       */
    Pfloat      *max_anno_char_ht       /* OUT maximum anno. text char. height       */
);
```

INQUIRE TEXT EXTENT

```
void pinq_text_extent(
    Pint                ws_type,          /* workstation type           */
    Pint                text_font,        /* text font                  */
    Pfloat              char_expan,       /* char expansion factor      */
    Pfloat              char_space,       /* char spacing               */
    Pfloat              char_ht,          /* char height                */
    Ptext_path          text_path,        /* text path                  */
    Phor_text_align     hor_text_align,   /* horizontal text alignment  */
    Pvert_text_align    ver_text_align,   /* vertical text alignment    */
    const char          *char_string,     /* character string           */
    Pint                *err_ind,         /* OUT error indicator        */
    Prect               *rect,            /* OUT extent rectangle       */
    Ppoint              *offset           /* OUT concatenation offset   */
);
```

INQUIRE INTERIOR FACILITIES

```
void pinq_int_facs(
    Pint        ws_type,                        /* workstation type                    */
    Pint        hatch_num_elems_appl_length,    /* # elems in appl hatch style list    */
    Pint        hatch_start_ind,                /* starting index                      */
    Pint        *err_ind,                       /* OUT error indicator                 */
    Pint_facs   *int_facs,                      /* OUT interior facilities             */
    Pint        *hatch_num_elems_impl_list      /* OUT # elems in impl list            */
);
```

INQUIRE PREDEFINED INTERIOR REPRESENTATION

```
void pinq_pred_int_rep(
    Pint          ws_type,    /* workstation type              */
    Pint          index,      /* predefined index              */
    Pint          *err_ind,   /* OUT error indicator           */
    Pint_bundle   *bundle     /* OUT predefined interior rep   */
);
```

INQUIRE EDGE FACILITIES

```
void pinq_edge_facs(
    Pint         ws_type,               /* workstation type          */
    Pint         num_elems_appl_list,   /* # elems in appl list      */
    Pint         start_ind,             /* starting index            */
    Pint         *err_ind,              /* OUT error indicator       */
    Pedge_facs   *fac,                  /* OUT edge facilities       */
    Pint         *num_elems_impl_list   /* OUT # elems in impl list  */
);
```

INQUIRE PREDEFINED EDGE REPRESENTATION

```
void pinq_pred_edge_rep(
    Pint           ws_type,    /* workstation type          */
    Pint           index,      /* predefined index          */
    Pint           *err_ind,   /* OUT error indicator       */
    Pedge_bundle   *bundle     /* OUT predefined edge rep   */
);
```

INQUIRE PATTERN FACILITIES

```
void pinq_pat_facs(
    Pint   ws_type,    /* workstation type                          */
    Pint   *err_ind,   /* OUT error indicator                       */
    Pint   *num_pred   /* OUT number of predefined pattern indices  */
);
```

INQUIRE PREDEFINED PATTERN REPRESENTATION

```
void pinq_pred_pat_rep(
    Pint       ws_type,    /* workstation type            */
    Pint       index,      /* predefined index            */
    Pstore     store,      /* handle to Store object      */
    Pint       *err_ind,   /* OUT error indicator         */
    Ppat_rep   **pat_rep   /* OUT predefined pattern rep  */
);
```

NOTE — The memory referenced by `*pat_rep` is managed by `store`.

INQUIRE COLOUR MODEL FACILITIES

```
void pinq_colr_model_facs(
    Pint        ws_type,                /* workstation type              */
    Pint        num_elems_appl_list,    /* # elems in appl list          */
    Pint        start_ind,              /* starting index                */
    Pint        *err_ind,               /* OUT error indicator           */
    Pint_list   *models,                /* OUT list of colour models     */
    Pint        *num_elems_impl_list,   /* OUT # elems in impl list      */
    Pint        *def                    /* OUT default colour model      */
);
```

INQUIRE COLOUR FACILITIES

```
void pinq_colr_facs(
    Pint         ws_type,    /* workstation type      */
    Pint         *err_ind,   /* OUT error indicator   */
    Pcolr_facs   *fac        /* OUT colour facilities */
);
```

INQUIRE PREDEFINED COLOUR REPRESENTATION

```
void pinq_pred_colr_rep(
    Pint        ws_type,    /* workstation type                        */
    Pint        colr_ind,   /* predefined index                        */
    Pint        *err_ind,   /* OUT error indicator                     */
    Pcolr_rep   *colr_rep   /* OUT predefined colour representation    */
);
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3

```
void pinq_list_avail_gdp3(
    Pint        ws_type,                /* workstation type          */
    Pint        num_elems_appl_list,    /* # elems in appl list      */
    Pint        start_ind,              /* starting index            */
    Pint        *err_ind,               /* OUT error indicator       */
    Pint_list   *gdps,                  /* OUT list of GDPs          */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES

```
void pinq_list_avail_gdp(
    Pint        ws_type,                /* workstation type           */
    Pint        num_elems_appl_list,    /* # elems in appl list       */
    Pint        start_ind,              /* starting index             */
    Pint        *err_ind,               /* OUT error indicator        */
    Pint_list   *gdps,                  /* OUT list of GDPs           */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list   */
);
```

INQUIRE GENERALIZED DRAWING PRIMITIVE 3

```
void pinq_gdp3(
    Pint    ws_type,    /* workstation type                 */
    Pint    gdp,        /* GDP function identifier          */
    Pint    *err_ind,   /* OUT error indicator              */
    Pint    *num_attr,  /* OUT number of attributes used    */
    Pattrs  attr[5]     /* OUT list of attributes used      */
);
```

INQUIRE GENERALIZED DRAWING PRIMITIVE

```
void pinq_gdp(
    Pint    ws_type,    /* workstation type                 */
    Pint    gdp,        /* GDP function identifier          */
    Pint    *err_ind,   /* OUT error indicator              */
    Pint    *num_attr,  /* OUT number of attributes used    */
    Pattrs  attr[5]     /* OUT list of attributes used      */
);
```

INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS

```
void pinq_list_avail_gse(
    Pint        ws_type,                /* workstation type           */
    Pint        num_elems_appl_list,    /* # elems in appl list       */
    Pint        start_ind,              /* starting index             */
    Pint        *err_ind,               /* OUT error indicator        */
    Pint_list   *gses,                  /* OUT list of GSEs           */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list   */
);
```

INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED

```
void pinq_num_disp_pris(
    Pint   ws_type,    /* workstation type                    */
    Pint   *err_ind,   /* OUT error indicator                 */
    Pint   *num_pri    /* OUT number of structure priorities  */
);
```

INQUIRE WORKSTATION STATE TABLE LENGTHS

```
void pinq_ws_st_table(
    Pint            ws_type,    /* workstation type                  */
    Pint            *err_ind,   /* OUT error indicator               */
    Pws_st_tables   *lengths    /* OUT lengths of workstation tables */
);
```

INQUIRE DYNAMICS OF STRUCTURES

```
void pinq_dyns_structs(
    Pint            ws_type,    /* workstation type        */
    Pint            *err_ind,   /* OUT error indicator     */
    Pdyns_structs   *dyns       /* OUT structure dynamics  */
);
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES

```
void pinq_num_avail_in(
    Pint      ws_type,    /* workstation type             */
    Pint      *err_ind,   /* OUT error indicator          */
    Pnum_in   *num_in     /* OUT number of input devices  */
);
```

INQUIRE DEFAULT LOCATOR DEVICE DATA 3

```
void pinq_def_loc_data3(
    Pint         ws_type,      /* workstation type                 */
    Pint         loc_num,      /* logical input device number      */
    Pstore       store,        /* handle to Store object           */
    Pint         *err_ind,     /* OUT error indicator              */
    Ppoint3      *loc_pos,     /* OUT default initial position     */
    Pint_list    **pet_list,   /* OUT list of prompt and echo types */
    Plimit3      *echo_vol,    /* OUT default echo volume          */
    Ploc_data3   **loc_data    /* OUT default data record          */
);
```

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

INQUIRE DEFAULT LOCATOR DEVICE DATA

```
void pinq_def_loc_data(
    Pint        ws_type,      /* workstation type                    */
    Pint        loc_num,      /* logical input device number         */
    Pstore      store,        /* handle to Store object              */
    Pint        *err_ind,     /* OUT error indicator                 */
    Ppoint      *loc_pos,     /* OUT default locator position        */
    Pint_list   **pet_list,   /* OUT list of prompt and echo types   */
    Plimit      *echo_area,   /* OUT default echo area               */
    Ploc_data   **loc_data    /* OUT default data record             */
);
```

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

INQUIRE DEFAULT STROKE DEVICE DATA 3

```
void pinq_def_stroke_data3(
    Pint            ws_type,        /* workstation type                 */
    Pint            stroke_num,     /* logical input device number      */
    Pstore          store,          /* handle to Store object           */
    Pint            *err_ind,       /* OUT error indicator              */
    Pint            *max_buf_size,  /* OUT max. input buffer size       */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types*/
    Plimit3         *echo_vol,      /* OUT default echo volume          */
    Pstroke_data3   **stroke_data   /* OUT default data record          */
);
```

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

INQUIRE DEFAULT STROKE DEVICE DATA

```
void pinq_def_stroke_data(
    Pint            ws_type,        /* workstation type                 */
    Pint            stroke_num,     /* logical input device number      */
    Pstore          store,          /* handle to Store object           */
    Pint            *err_ind,       /* OUT error indicator              */
    Pint            *max_buf_size,  /* OUT max. input buffer size       */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types*/
    Plimit          *echo_area,     /* OUT default echo area            */
    Pstroke_data    **stroke_data   /* OUT default data record          */
);
```

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

INQUIRE DEFAULT VALUATOR DEVICE DATA 3

```
void pinq_def_val_data3(
    Pint          ws_type,      /* workstation type                    */
    Pint          val_num,      /* logical input device number         */
    Pstore        store,        /* handle to Store object              */
    Pint          *err_ind,     /* OUT error indicator                 */
    Pfloat        *def_value,   /* OUT default initial value           */
    Pint_list     **pet_list,   /* OUT list of prompt and echo types   */
    Plimit3       *echo_vol,    /* OUT default echo volume             */
    Pval_data3    **val_data    /* OUT default data record             */
);
```

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

INQUIRE DEFAULT VALUATOR DEVICE DATA

```
void pinq_def_val_data(
    Pint          ws_type,      /* workstation type                    */
    Pint          val_num,      /* logical input device number         */
    Pstore        store,        /* handle to Store object              */
    Pint          *err_ind,     /* OUT error indicator                 */
    Pfloat        *def_value,   /* OUT default initial value           */
    Pint_list     **pet_list,   /* OUT list of prompt and echo types   */
    Plimit        *echo_area,   /* OUT default echo area               */
    Pval_data     **val_data    /* OUT default data record             */
);
```

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

INQUIRE DEFAULT CHOICE DEVICE DATA 3

```
void pinq_def_choice_data3(
    Pint              ws_type,        /* workstation type                    */
    Pint              choice_num,     /* logical input device number         */
    Pstore            store,          /* handle to Store object              */
    Pint              *err_ind,       /* OUT error indicator                 */
    Pint              *max_choices,   /* OUT max. number of choices          */
    Pint_list         **pet_list,     /* OUT list of prompt and echo types   */
    Plimit3           *echo_vol,      /* OUT default echo volume             */
    Pchoice_data3     **choice_data   /* OUT default data record             */
);
```

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

INQUIRE DEFAULT CHOICE DEVICE DATA

```
void pinq_def_choice_data(
    Pint          ws_type,       /* workstation type                    */
    Pint          choice_num,    /* logical input device number         */
    Pstore        store,         /* handle to Store object              */
    Pint          *err_ind,      /* OUT error indicator                 */
    Pint          *max_choices,  /* OUT max. number of choices          */
    Pint_list     **pet_list,    /* OUT list of prompt and echo types   */
    Plimit        *echo_area,    /* OUT default echo area               */
    Pchoice_data  **choice_data  /* OUT default data record             */
);
```

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

INQUIRE DEFAULT PICK DEVICE DATA 3

```
void pinq_def_pick_data3(
    Pint          ws_type,       /* workstation type                    */
    Pint          pick_num,      /* logical input device number         */
    Pstore        store,         /* handle to Store object              */
    Pint          *err_ind,      /* OUT error indicator                 */
    Pint_list     **pet_list,    /* OUT list of prompt and echo types   */
    Plimit3       *echo_vol,     /* OUT default echo volume             */
    Ppick_data3   **pick_data    /* OUT default data record             */
);
```

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

INQUIRE DEFAULT PICK DEVICE DATA

```
void pinq_def_pick_data(
    Pint          ws_type,       /* workstation type                    */
    Pint          pick_num,      /* logical input device number         */
    Pstore        store,         /* handle to Store object              */
    Pint          *err_ind,      /* OUT error indicator                 */
    Pint_list     **pet_list,    /* OUT list of prompt and echo types   */
    Plimit        *echo_area,    /* OUT default echo area               */
    Ppick_data    **pick_data    /* OUT default data record             */
);
```

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

INQUIRE DEFAULT STRING DEVICE DATA 3

```
void pinq_def_string_data3(
    Pint            ws_type,        /* workstation type                  */
    Pint            string_num,     /* logical input device number       */
    Pstore          store,          /* handle to Store object            */
    Pint            *err_ind,       /* OUT error indicator               */
    Pint            *max_buf_size,  /* OUT max. input buffer size        */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types */
    Plimit3         *echo_vol,      /* OUT default echo volume           */
    Pstring_data3   **string_data   /* OUT default data record           */
);
```

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

INQUIRE DEFAULT STRING DEVICE DATA

```
void pinq_def_string_data(
    Pint            ws_type,        /* workstation type                  */
    Pint            string_num,     /* logical input device number       */
    Pstore          store,          /* handle to Store object            */
    Pint            *err_ind,       /* OUT error indicator               */
    Pint            *max_buf_size,  /* OUT max. input buffer size        */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types */
    Plimit          *echo_area,     /* OUT default echo area             */
    Pstring_data    **string_data   /* OUT default data record           */
);
```

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

7.12.6 Inquiry function for structure state list

INQUIRE SET OF WORKSTATIONS TO WHICH POSTED

```
void pinq_wss_posted(
    Pint        struct_id,              /* structure identifier      */
    Pint        num_elems_appl_list,    /* # elems in appl list      */
    Pint        start_ind,              /* starting index            */
    Pint        *err_ind,               /* OUT error indicator       */
    Pint_list   *ws,                    /* OUT list of workstations  */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

7.12.7 Inquiry functions for structure content

INQUIRE OPEN STRUCTURE

```
void pinq_open_struct(
    Pint                  *err_ind,    /* OUT error indicator            */
    Popen_struct_status   *status,     /* OUT status of open structure   */
    Pint                  *struct_id   /* OUT structure identifier       */
);
```

INQUIRE ELEMENT POINTER

```
void pinq_elem_ptr(
    Pint   *err_ind,         /* OUT error indicator         */
    Pint   *elem_ptr_value   /* OUT element pointer value   */
);
```

INQUIRE CURRENT ELEMENT TYPE AND SIZE

```
void pinq_cur_elem_type_size(
    Pint         *err_ind,     /* OUT error indicator   */
    Pelem_type   *elem_type,   /* OUT element type      */
    size_t       *elem_size    /* OUT element size      */
);
```

INQUIRE CURRENT ELEMENT CONTENT

```
void pinq_cur_elem_content(
    Pstore       store,        /* handle to Store object   */
    Pint         *err_ind,     /* OUT error indicator      */
    Pelem_data   **elem_data   /* OUT element data         */
);
```

NOTE — The memory referenced by `*elem_data` is managed by `store`.

INQUIRE ELEMENT TYPE AND SIZE

```
void pinq_elem_type_size(
    Pint         struct_id,    /* structure identifier   */
    Pint         elem_num,     /* element number         */
    Pint         *err_ind,     /* OUT error indicator    */
    Pelem_type   *elem_type,   /* OUT element type       */
    size_t       *elem_size    /* OUT element size       */
);
```

INQUIRE ELEMENT CONTENT

```
void pinq_elem_content(
    Pint         struct_id,    /* structure identifier     */
    Pint         elem_num,     /* element number           */
    Pstore       store,        /* handle to Store object   */
    Pint         *err_ind,     /* OUT error indicator      */
    Pelem_data   **elem_data   /* OUT data record          */
);
```

NOTE — The memory referenced by `*elem_data` is managed by `store`.

INQUIRE STRUCTURE STATUS

```
void pinq_struct_status(
    Pint             struct_id,   /* structure identifier   */
    Pint             *err_ind,    /* OUT error indicator    */
    Pstruct_status   *status      /* OUT existence status   */
);
```

INQUIRE PATHS TO ANCESTORS

```
void pinq_paths_ances(
    Pint                 struct_id,   /* structure identifier     */
    Ppath_order          order,       /* path order               */
    Pint                 depth,       /* path depth               */
    Pstore               store,       /* handle to Store object   */
    Pint                 *err_ind,    /* OUT error indicator      */
    Pelem_ref_list_list  **paths      /* OUT structure path list  */
);
```

NOTE — The memory referenced by `*paths` is managed by `store`.

INQUIRE PATHS TO DESCENDANTS

```
void pinq_paths_descs(
    Pint                 struct_id,   /* structure identifier     */
    Ppath_order          order,       /* path order               */
    Pint                 depth,       /* path depth               */
    Pstore               store,       /* handle to Store object   */
    Pint                 *err_ind,    /* OUT error indicator      */
    Pelem_ref_list_list  **paths      /* OUT structure path list  */
);
```

NOTE — The memory referenced by `*paths` is managed by `store`.

ELEMENT SEARCH

```
void pelem_search(
    Pint                    struct_id,        /* structure identifier        */
    Pint                    start_elem,       /* starting element position   */
    Psearch_dir             dir,              /* search direction            */
    const Pelem_type_list   *incl,            /* element incl. list          */
    const Pelem_type_list   *excl,            /* element excl. list          */
    Pint                    *err_ind,         /* OUT error indicator         */
    Psearch_status          *status,          /* OUT search status           */
    Pint                    *found_elem_ptr   /* OUT found element pointer   */
);
```

INCREMENTAL SPATIAL SEARCH 3

```
void pincr_spa_search3(
    const Ppoint3           *ref_pt,              /* search reference point     */
    Pfloat                  dist,                 /* search distance            */
    const Pelem_ref_list    *sp,                  /* starting path list         */
    Pclip_ind               clip_ind,             /* modeling clip indicator    */
    Pint                    ceiling,              /* search ceiling index       */
    const Pfilter_list      *norm,                /* normal filter list         */
    const Pfilter_list      *inv,                 /* inverted filter list       */
    Pint                    num_elems_appl_list,  /* # elems in appl list       */
    Pint                    start_ind,            /* starting index             */
    Pint                    *err_ind,             /* OUT error indicator        */
    Pelem_ref_list          *fp,                  /* OUT found path             */
    Pint                    *num_elems_impl_list  /* OUT # elems in impl list   */
);
```

INCREMENTAL SPATIAL SEARCH

```
void pincr_spa_search(
    const Ppoint            *ref_pt,              /* search reference point     */
    Pfloat                  dist,                 /* search distance            */
    const Pelem_ref_list    *sp,                  /* starting path list         */
    Pclip_ind               clip_ind,             /* modeling clip indicator    */
    Pint                    ceiling,              /* search ceiling index       */
    const Pfilter_list      *norm,                /* normal filter list         */
    const Pfilter_list      *inv,                 /* inverted filter list       */
    Pint                    num_elems_appl_list,  /* # elems in appl list       */
    Pint                    start_ind,            /* starting index             */
    Pint                    *err_ind,             /* OUT error indicator        */
    Pelem_ref_list          *fp,                  /* OUT found path             */
    Pint                    *num_elems_impl_list  /* OUT # elems in impl list   */
);
```

7.12.8 Inquiry functions for error state list

INQUIRE INPUT QUEUE OVERFLOW

```
void pinq_in_overf(
    Pint        *err_ind,    /* OUT error indicator          */
    Pint        *ws_id,      /* OUT workstation identifier   */
    Pin_class   *in_class,   /* OUT input class              */
    Pint        *in_num      /* OUT input device number      */
);
```

INQUIRE ERROR HANDLING MODE

```
void pinq_err_hand_mode(
    Pint        *err_ind,   /* OUT error indicator   */
    Perr_mode   *err_mode   /* OUT error mode        */
);
```

7.13 Error control

EMERGENCY CLOSE PHIGS

```
void pemergency_close_phigs(
        void
);
```

ERROR HANDLING

```
void perr_hand(
    Pint            error_num,    /* error number                                  */
    Pint            func_num,     /* identifier of function that detected the error */
    const char      *error_file   /* name of error file                            */
);
```

ERROR LOGGING

```
void perr_log(
    Pint            error_num,    /* error number                                  */
    Pint            func_num,     /* identifier of function that detected the error */
    const char      *error_file   /* name of error file                            */
);
```

SET ERROR HANDLING MODE

```
void pset_err_hand_mode(
    Perr_mode   error_mode   /* error handling mode   */
);
```

7.14 Special interfaces

ESCAPE

```
void pescape(
    Pint                    func_id,     /* escape function identifier        */
    const Pescape_in_data   *in_data,    /* input data for the function       */
    Pstore                  store,       /* handle to Store object            */
    Pescape_out_data        **out_data   /* OUT output data of the function   */
);
```

If an error is detected by this function, `*out_data` shall be set to NULL.

NOTE — The memory referenced by `*out_data` is managed by `store`.

7.15 Binding defined utility functions

SET ERROR HANDLING

```
void pset_err_hand(
    void   (*new_err_hand)(),  /* application's error handling function   */
    void   (**old_err_hand)()  /* OUT old error handling function         */
);
```

CREATE STORE

```
void pcreate_store(
    Pint     *err_ind,   /* OUT error indicator   */
    Pstore   *store      /* OUT new Store         */
);
```

DELETE STORE

```
void pdel_store(
    Pint     *err_ind,   /* OUT error indicator          */
    Pstore   *store      /* IN/OUT Store to be deleted   */
);
```

Annex A

(informative)

Data types in compilation order and external functions

Annex A lists the data types of the binding in an order suitable for compilation and also lists an external declaration for each function.

```
#include <stddef.h>
/*
 * Note: the field "int impl_dep;" is a place-holder for
 * a implementation dependent field.
 */
```

A.1 Macro definitions

```
/*                              <0          Implementation Dependent Errors */

#define  PE_NO_ERROR            (0)      /* No Error */

/* State Errors */
#define  PE_NOT_PHCL            (1)      /* Ignoring function, function requires
                                            state (PHCL,WSCL,STCL,ARCL) */
#define  PE_NOT_PHOP            (2)      /* Ignoring function, function requires
                                            state (PHOP,*,*,*) */
#define  PE_NOT_WSOP            (3)      /* Ignoring function, function requires
                                            state (PHOP,WSOP,*,*) */
#define  PE_NOT_CL              (4)      /* Ignoring function, function requires
                                            state (PHOP,WSCL,STCL,ARCL) */
#define  PE_NOT_STOP            (5)      /* Ignoring function, function requires
                                            state (PHOP,*,STOP,*) */
#define  PE_NOT_STCL            (6)      /* Ignoring function, function requires
                                            state (PHOP,*,STCL,*) */
#define  PE_NOT_AROP            (7)      /* Ignoring function, function requires
                                            state (PHOP,*,*,AROP) */

/* Workstation Errors */
#define  PE_BAD_CONN_ID         (50)     /* Ignoring function, connection iden-
                                            tifier not recognized by the imple-
                                            mentation */
#define  PE_WS_TYPE             (51)     /* Ignoring function, this information
                                            is not yet available for this works-
                                            tation type; open a workstation of
                                            this type and use the specific works-
                                            tation type */
#define  PE_BAD_WS_TYPE         (52)     /* Ignoring function, workstation type
                                            not recognized by the implementation
                                            */
#define  PE_DUP_WS_ID           (53)     /* Ignoring function, workstation iden-
                                            tifier already is in use */
#define  PE_WS_NOT_OPEN         (54)     /* Ignoring function, the specified
                                            workstation is not open */
#define  PE_NO_OPEN_WS          (55)     /* Ignoring function, workstation cannot
                                            be opened for an implementation
```

```
                                        dependent reason */
#define  PE_WS_NOT_MO          (56)     /* Ignoring function, specified worksta-
                                        tion is not of category MO */
#define  PE_WS_MI              (57)     /* Ignoring function, specified worksta-
                                        tion is of category MI */
#define  PE_WS_NOT_MI          (58)     /* Ignoring function, specified worksta-
                                        tion is not of category MI */
#define  PE_WS_NO_OUTPUT       (59)     /* Ignoring function, the specified
                                        workstation does not have output
                                        capability (i.e., the workstation
                                        category is neither OUTPUT, OUTIN,
                                        nor MO) */
#define  PE_WS_NOT_OUTIN       (60)     /* Ignoring function, specified worksta-
                                        tion is not of category OUTIN */
#define  PE_WS_NO_INPUT        (61)     /* Ignoring function, specified worksta-
                                        tion is neither of category INPUT nor
                                        of category OUTIN */
#define  PE_WS_NOT_OUT         (62)     /* Ignoring function, specified worksta-
                                        tion is neither of category OUTPUT
                                        nor of category OUTIN */
#define  PE_MAX_WS             (63)     /* Ignoring function, opening this
                                        workstation would exceed the maximum
                                        number of simultaneously open works-
                                        tations */
#define  PE_NO_GDP             (64)     /* Ignoring function, the specified
                                        workstation type is not able to gen-
                                        erate the specified generalized draw-
                                        ing primitive */

/* Output Attribute Errors */
#define  PE_BUN_IND_LT_1       (100)    /* Ignoring function, the bundle index
                                        value is less than one */
#define  PE_REP_UNDEF          (101)    /* Ignoring function, the specified
                                        representation has not been defined
                                        */
#define  PE_REP_NOT_PREDEF     (102)    /* Ignoring function, the specified
                                        representation has not been prede-
                                        fined on this workstation */
#define  PE_MAX_BUN            (103)    /* Ignoring function, setting this bun-
                                        dle table entry would exceed the max-
                                        imum number of entries allowed in the
                                        workstation bundle table */
#define  PE_BAD_LINETYPE       (104)    /* Ignoring function, the specified
                                        linetype is not available on the
                                        specified workstation */
#define  PE_BAD_MARKER_TYPE    (105)    /* Ignoring function, the specified
                                        marker type is not available on the
                                        specified workstation */
#define  PE_BAD_FONT           (106)    /* Ignoring function, the specified font
                                        is not available for the requested
                                        text precision on the specified
                                        workstation */
#define  PE_BAD_EDGETYPE       (107)    /* Ignoring function, the specified
                                        edgetype is not available on the
```

```
                                        specified workstation */
#define  PE_BAD_INT_STYLE       (108)   /* Ignoring function, the specified
                                           interior style is not available on
                                           the workstation */
#define  PE_NO_PAT              (109)   /* Ignoring function, interior style
                                           PATTERN is not supported on the
                                           workstation */
#define  PE_BAD_COLR_MODEL      (110)   /* Ignoring function, the specified
                                           colour model is not available on the
                                           workstation */
#define  PE_BAD_HLHSR_MODE      (111)   /* Ignoring function, the specified
                                           HLHSR mode is not available on the
                                           specified workstation */
#define  PE_PAT_IND_LT_1        (112)   /* Ignoring function, the pattern index
                                           value is less than one */
#define  PE_COLR_IND_LT_0       (113)   /* Ignoring function, the colour index
                                           value is less than zero */
#define  PE_VIEW_IND_LT_0       (114)   /* Ignoring function, the view index
                                           value is less than zero */
#define  PE_VIEW_IND_LT_1       (115)   /* Ignoring function, the view index
                                           value is less than one */
#define  PE_BAD_PAT_DIM         (116)   /* Ignoring function, one of the dimen-
                                           sions of pattern colour array is less
                                           than one */
#define  PE_BAD_COLR_DIM        (117)   /* Ignoring function, one of the dimen-
                                           sions of the colour index array is
                                           less than zero */
#define  PE_BAD_COLR            (118)   /* Ignoring function, one of the com-
                                           ponents of the colour specification
                                           is out of range.  The valid range is
                                           dependent upon the current colour
                                           model */

/* Transformations and Viewing Errors */
#define  PE_MAX_VIEW            (150)   /* Ignoring function, setting this view
                                           table entry would exceed the maximum
                                           number of entries allowed in the
                                           workstations view table */
#define  PE_INVALID_WINDOW      (151)   /* Ignoring function, invalid window;
                                           XMIN >= XMAX, YMIN >= YMAX or ZMIN >
                                           ZMAX */
#define  PE_INVALID_VIEWPORT    (152)   /* Ignoring function, invalid viewport;
                                           XMIN >= XMAX, YMIN >= YMAX or ZMIN >
                                           ZMAX */
#define  PE_INVALID_CLIP        (153)   /* Ignoring function, invalid view clip-
                                           ping limits; XMIN >= XMAX, YMIN >=
                                           YMAX or ZMIN > ZMAX */
#define  PE_BAD_CLIP            (154)   /* Ignoring function, the view clipping
                                           limits are not within NPC range */
#define  PE_BAD_PROJ_VIEWPORT   (155)   /* Ignoring function, the projection
                                           viewport limits are not withing NPC
                                           range */
#define  PE_BAD_WS_WINDOW       (156)   /* Ignoring function, the workstation
                                           window limits are not within NPC
```

```
                                        range */
#define  PE_BAD_WS_VIEWPORT     (157)   /* Ignoring function, the workstation
                                           viewport is not within display space
                                           */
#define  PE_BAD_PLANES          (158)   /* Ignoring function, front plane and
                                           back plane distances are equal when
                                           z-extent of the projection viewport
                                           is zero */
#define  PE_BAD_VPN             (159)   /* Ignoring function, the view plane
                                           normal vector has length zero */
#define  PE_BAD_VUP             (160)   /* Ignoring function, the view up vector
                                           has length zero */
#define  PE_BAD_VUP_VPN         (161)   /* Ignoring function, the view up and
                                           view plane normal vectors are paral-
                                           lel thus the viewing coordinate sys-
                                           tem cannot be established */
#define  PE_BAD_PRP             (162)   /* Ignoring function, the projection
                                           reference point is between the front
                                           and back planes */
#define  PE_PRP_VIEW_PLANE      (163)   /* Ignoring function, the projection
                                           reference point cannot be positioned
                                           on the view plane */
#define  PE_FRONT_BACK          (164)   /* Ignoring function, the back plane is
                                           in front of the front plane */

/* Structure Errors */
#define  PE_IGNORE_STRUCTS      (200)   /* Warning, ignoring structures that do
                                           not exist */
#define  PE_BAD_STRUCT          (201)   /* Ignoring function, the specified
                                           structure does not exist */
#define  PE_BAD_ELEMENT         (202)   /* Ignoring function, the specified ele-
                                           ment does not exist */
#define  PE_BAD_PATH            (203)   /* Ignoring function, specified starting
                                           path not found in CSS  */
#define  PE_BAD_CEILING_IND     (204)   /* Ignoring function, specified search
                                           ceiling index out of range */
#define  PE_NO_LABEL            (205)   /* Ignoring function, the label does not
                                           exist in the open structure between
                                           the element pointer and the end of
                                           the structure */
#define  PE_NO_LABELS           (206)   /* Ignoring function, one or both of the
                                           labels does not exist in the open
                                           structure between the element pointer
                                           and the end of the structure */
#define  PE_BAD_PATH_DEPTH      (207)   /* Ignoring function, the specified
                                           path depth is less than zero (0) */
#define  PE_BAD_DISP_PRI        (208)   /* Ignoring function, the display prior-
                                           ity is out of range */

/* Input Errors */
#define  PE_NO_DEVICE           (250)   /* Ignoring function, the specified dev-
                                           ice is not available on the specified
                                           workstation */
#define  PE_NOT_REQUEST         (251)   /* Ignoring function, the function
```

```
                                        requires the input device to be in
                                        REQUEST mode */
#define  PE_NOT_SAMPLE          (252)   /* Ignoring function, the function
                                        requires the input device to be in
                                        SAMPLE mode */
#define  PE_BAD_PET             (253)   /* Warning, the specified prompt/echo
                                        type is not available on the speci-
                                        fied workstation.  Prompt/echo type
                                        one will be used in its place */
#define  PE_INVALID_ECHO        (254)   /* Ignoring function, invalid echo
                                        area/volume; XMIN >= XMAX, YMIN >=
                                        YMAX or ZMIN > ZMAX */
#define  PE_BAD_ECHO            (255)   /* Ignoring function, one of the echo
                                        area/volume boundary points is out-
                                        side the range of the device */
#define  PE_QUEUE_OFLOW         (256)   /* Warning, the input queue has over-
                                        flowed */
#define  PE_NO_QUEUE_OFLOW      (257)   /* Ignoring function, input queue has
                                        not overflowed */
#define  PE_OFLOW_NO_GO         (258)   /* Ignoring function, input queue has
                                        overflowed, but associated worksta-
                                        tion has been closed */
#define  PE_BAD_CLASS           (259)   /* Ignoring function, the input device
                                        class of the current input report
                                        does not match the class being
                                        requested */
#define  PE_BAD_DATA_REC        (260)   /* Ignoring function, one of the fields
                                        within the input device data record
                                        is in error */
#define  PE_INVALID_VALUE       (261)   /* Ignoring function, initial value is
                                        invalid */
#define  PE_STROKE_BUF_SIZE     (262)   /* Ignoring function, number of points
                                        in the initial stroke is greater than
                                        the buffer size */
#define  PE_STRING_BUF_SIZE     (263)   /* Ignoring function, length of the ini-
                                        tial string is greater than the
                                        buffer size */

/* Metafile Errors */
#define  PE_ILLEGAL_ITEM_TYPE   (300)   /* Ignoring function, item type is not
                                        allowed for user items */
#define  PE_INVALID_ITEM_LENGTH         (301)/*Ignoring function, item length is
                                        invalid */
#define  PE_METAFILE_EMPTY      (302)   /* Ignoring function, no item is left in
                                        metafile input */
#define  PE_INVALID_ITEM        (303)   /* Ignoring function, metafile item is
                                        invalid */
#define  PE_BAD_ITEM_TYPE       (304)   /* Ignoring function, item type is
                                        unknown */
#define  PE_BAD_ITEM_REC        (305)   /* Ignoring function, content of item
                                        data record is invalid for the speci-
                                        fied item type */
#define  PE_MAX_ITEM_LENGTH     (306)   /* Ignoring function, maximum item data
                                        record length is invalid */
```

```
#define  PE_USER_ITEM              (307)   /* Ignoring function, user item cannot
                                              be interpreted */

/* Escape Errors */
#define  PE_ESCAPE_NOT_AVAIL       (350)   /* Warning, the specified escape is not
                                              available on one or more workstations
                                              in this implementation.  The escape
                                              will be processed by those worksta-
                                              tions on which it is available */
#define  PE_BAD_ESCAPE_DATA        (351)   /* Ignoring function, one of the fields
                                              within the escape data record is in
                                              error */

/* Archival and Retrieval Errors */
#define  PE_AR_CANT_OPEN           (400)   /* Ignoring function, the archive file
                                              cannot be opened */
#define  PE_MAX_AR                 (401)   /* Ignoring function, opening this
                                              archive file would exceed the maximum
                                              number of simultaneously open archive
                                              files */
#define  PE_DUP_AR_ID              (402)   /* Ignoring function, archive file iden-
                                              tifier already in use */
#define  PE_BAD_AR                 (403)   /* Ignoring function, the archive file
                                              is not a PHIGS archive file */
#define  PE_AR_NOT_OPEN            (404)   /* Ignoring function, the specified
                                              archive file is not open */
#define  PE_NAME_CONFLICT          (405)   /* Ignoring function, name conflict
                                              occurred while conflict resolution
                                              flag has value ABANDON */
#define  PE_AR_FULL                (406)   /* Warning, the archive file is full.
                                              Any structures that were archived
                                              were archived in total */
#define  PE_AR_NO_STRUCT           (407)   /* Warning, some of the specified struc-
                                              tures do not exist on the archive
                                              file */
#define  PE_AR_NO_STRUCT_EMPTY     (408)   /* Warning, some of the specified struc-
                                              tures do not exist on the archive
                                              file.  PHIGS will create empty struc-
                                              ture in their places */

/* Miscellaneous Errors */
#define  PE_BAD_ERROR_FILE         (450)   /* Ignoring function, the specified
                                              error file is invalid */

/* System Errors */
#define  PE_OFLOW_PHIGS            (900)   /* Storage overflow has occurred in
                                              PHIGS */
#define  PE_OFLOW_CSS              (901)   /* Storage overflow has occurred in CSS
                                              */
#define  PE_IO_ERROR_READ          (902)   /* Input/Output error has occurred while
                                              reading */
#define  PE_IO_ERROR_WRITE         (903)   /* Input/Output error has occurred while
                                              writing */
#define  PE_IO_ERROR_TO_WS         (904)   /* Input/Output error has occurred while
```

```
                                        sending data to a workstation */
#define  PE_IO_ERROR_FROM_WS    (905)   /* Input/Output error has occurred
                                        receiving data from a workstation */
#define  PE_IO_ERROR_LIB        (906)   /* Input/Output error has occurred dur-
                                        ing program library management */
#define  PE_IO_ERROR_WDT        (907)   /* Input/Output error has occurred while
                                        reading workstation description table
                                        */
#define  PE_ARITHMETIC_ERROR    (908)   /* Arithmetic error has occurred */

/*       Binding Specific Errors */
#define  PE_START_IND_INVAL     (2200)  /* Ignoring function, start index is out
                                        of range */
#define  PE_LIST_LENGTH_LT_ZERO         (2201)/*Ignoring function, the length of
                                        the application's list is negative */
#define  PE_ENUM_TYPE_INVAL     (2202)  /* Ignoring function, enumeration type
                                        out of range */
#define  PE_ALLOC_STORE         (2203)  /* Ignoring function, error while allo-
                                        cating a Store */
#define  PE_ALLOC_STORE_MEM     (2204)  /* Ignoring function, error while allo-
                                        cating memory for a Store */



#define  Pfn_open_phigs                 (0)
#define  Pfn_close_phigs                (1)
#define  Pfn_open_ws                    (2)
#define  Pfn_close_ws                   (3)
#define  Pfn_redraw_all_structs         (4)
#define  Pfn_upd_ws                     (5)
#define  Pfn_set_disp_upd_st            (6)
#define  Pfn_message                    (7)
#define  Pfn_polyline3                  (8)
#define  Pfn_polyline                   (9)
#define  Pfn_polymarker3                (10)
#define  Pfn_polymarker                 (11)
#define  Pfn_text3                      (12)
#define  Pfn_text                       (13)
#define  Pfn_anno_text_rel3             (14)
#define  Pfn_anno_text_rel              (15)
#define  Pfn_fill_area3                 (16)
#define  Pfn_fill_area                  (17)
#define  Pfn_fill_area_set3             (18)
#define  Pfn_fill_area_set              (19)
#define  Pfn_cell_array3                (20)
#define  Pfn_cell_array                 (21)
#define  Pfn_gdp3                       (22)
#define  Pfn_gdp                        (23)
#define  Pfn_set_line_ind               (24)
#define  Pfn_set_marker_ind             (25)
#define  Pfn_set_text_ind               (26)
#define  Pfn_set_int_ind                (27)
#define  Pfn_set_edge_ind               (28)
#define  Pfn_set_linetype               (29)
```

```
#define  Pfn_set_linewidth                  (30)
#define  Pfn_set_line_colr_ind              (31)
#define  Pfn_set_marker_type                (32)
#define  Pfn_set_marker_size                (33)
#define  Pfn_set_marker_colr_ind            (34)
#define  Pfn_set_text_font                  (35)
#define  Pfn_set_text_prec                  (36)
#define  Pfn_set_char_expan                 (37)
#define  Pfn_set_char_space                 (38)
#define  Pfn_set_text_colr_ind              (39)
#define  Pfn_set_char_ht                    (40)
#define  Pfn_set_char_up_vec                (41)
#define  Pfn_set_text_path                  (42)
#define  Pfn_set_text_align                 (43)
#define  Pfn_set_anno_char_ht               (44)
#define  Pfn_set_anno_char_up_vec           (45)
#define  Pfn_set_anno_path                  (46)
#define  Pfn_set_anno_align                 (47)
#define  Pfn_set_anno_style                 (48)
#define  Pfn_set_int_style                  (49)
#define  Pfn_set_int_style_ind              (50)
#define  Pfn_set_int_colr_ind               (51)
#define  Pfn_set_edge_flag                  (52)
#define  Pfn_set_edgetype                   (53)
#define  Pfn_set_edgewidth                  (54)
#define  Pfn_set_edge_colr_ind              (55)
#define  Pfn_set_pat_size                   (56)
#define  Pfn_set_pat_ref_point_vecs         (57)
#define  Pfn_set_pat_ref_point              (58)
#define  Pfn_add_names_set                  (59)
#define  Pfn_remove_names_set               (60)
#define  Pfn_set_indiv_asf                  (61)
#define  Pfn_set_line_rep                   (62)
#define  Pfn_set_marker_rep                 (63)
#define  Pfn_set_text_rep                   (64)
#define  Pfn_set_int_rep                    (65)
#define  Pfn_set_edge_rep                   (66)
#define  Pfn_set_pat_rep                    (67)
#define  Pfn_set_colr_rep                   (68)
#define  Pfn_set_highl_filter               (69)
#define  Pfn_set_invis_filter               (70)
#define  Pfn_set_colr_model                 (71)
#define  Pfn_set_hlhsr_id                   (72)
#define  Pfn_set_hlhsr_mode                 (73)
#define  Pfn_set_local_tran3                (74)
#define  Pfn_set_local_tran                 (75)
#define  Pfn_set_global_tran3               (76)
#define  Pfn_set_global_tran                (77)
#define  Pfn_set_model_clip_vol3            (78)
#define  Pfn_set_model_clip_vol             (79)
#define  Pfn_set_model_clip_ind             (80)
#define  Pfn_restore_model_clip_vol         (81)
#define  Pfn_set_view_ind                   (82)
#define  Pfn_set_view_rep3                  (83)
```

```
#define  Pfn_set_view_rep                    (84)
#define  Pfn_set_view_tran_in_pri            (85)
#define  Pfn_set_ws_win3                     (86)
#define  Pfn_set_ws_win                      (87)
#define  Pfn_set_ws_vp3                      (88)
#define  Pfn_set_ws_vp                       (89)
#define  Pfn_open_struct                     (90)
#define  Pfn_close_struct                    (91)
#define  Pfn_exec_struct                     (92)
#define  Pfn_label                           (93)
#define  Pfn_appl_data                       (94)
#define  Pfn_gse                             (95)
#define  Pfn_set_edit_mode                   (96)
#define  Pfn_copy_all_elems_struct           (97)
#define  Pfn_set_elem_ptr                    (98)
#define  Pfn_offset_elem_ptr                 (99)
#define  Pfn_set_elem_ptr_label              (100)
#define  Pfn_del_elem                        (101)
#define  Pfn_del_elem_range                  (102)
#define  Pfn_del_elems_labels                (103)
#define  Pfn_empty_struct                    (104)
#define  Pfn_del_struct                      (105)
#define  Pfn_del_struct_net                  (106)
#define  Pfn_del_all_structs                 (107)
#define  Pfn_change_struct_id                (108)
#define  Pfn_change_struct_refs              (109)
#define  Pfn_change_struct_id_refs           (110)
#define  Pfn_post_struct                     (111)
#define  Pfn_unpost_struct                   (112)
#define  Pfn_unpost_all_structs              (113)
#define  Pfn_open_ar_file                    (114)
#define  Pfn_close_ar_file                   (115)
#define  Pfn_ar_structs                      (116)
#define  Pfn_ar_struct_nets                  (117)
#define  Pfn_ar_all_structs                  (118)
#define  Pfn_set_conf_res                    (119)
#define  Pfn_ret_struct_ids                  (120)
#define  Pfn_ret_structs                     (121)
#define  Pfn_ret_struct_nets                 (122)
#define  Pfn_ret_all_structs                 (123)
#define  Pfn_ret_paths_ances                 (124)
#define  Pfn_ret_paths_descs                 (125)
#define  Pfn_del_structs_ar                  (126)
#define  Pfn_del_struct_nets_ar              (127)
#define  Pfn_del_all_structs_ar              (128)
#define  Pfn_set_pick_id                     (129)
#define  Pfn_set_pick_filter                 (130)
#define  Pfn_init_loc3                       (131)
#define  Pfn_init_loc                        (132)
#define  Pfn_init_stroke3                    (133)
#define  Pfn_init_stroke                     (134)
#define  Pfn_init_val3                       (135)
#define  Pfn_init_val                        (136)
#define  Pfn_init_choice3                    (137)
```

```
#define  Pfn_init_choice                (138)
#define  Pfn_init_pick3                 (139)
#define  Pfn_init_pick                  (140)
#define  Pfn_init_string3               (141)
#define  Pfn_init_string                (142)
#define  Pfn_set_loc_mode               (143)
#define  Pfn_set_stroke_mode            (144)
#define  Pfn_set_val_mode               (145)
#define  Pfn_set_choice_mode            (146)
#define  Pfn_set_pick_mode              (147)
#define  Pfn_set_string_mode            (148)
#define  Pfn_req_loc3                   (149)
#define  Pfn_req_loc                    (150)
#define  Pfn_req_stroke3                (151)
#define  Pfn_req_stroke                 (152)
#define  Pfn_req_val                    (153)
#define  Pfn_req_choice                 (154)
#define  Pfn_req_pick                   (155)
#define  Pfn_req_string                 (156)
#define  Pfn_sample_loc3                (157)
#define  Pfn_sample_loc                 (158)
#define  Pfn_sample_stroke3             (159)
#define  Pfn_sample_stroke              (160)
#define  Pfn_sample_val                 (161)
#define  Pfn_sample_choice              (162)
#define  Pfn_sample_pick                (163)
#define  Pfn_sample_string              (164)
#define  Pfn_await_event                (165)
#define  Pfn_flush_events               (166)
#define  Pfn_get_loc3                   (167)
#define  Pfn_get_loc                    (168)
#define  Pfn_get_stroke3                (169)
#define  Pfn_get_stroke                 (170)
#define  Pfn_get_val                    (171)
#define  Pfn_get_choice                 (172)
#define  Pfn_get_pick                   (173)
#define  Pfn_get_string                 (174)
#define  Pfn_write_item                 (175)
#define  Pfn_get_item_type              (176)
#define  Pfn_read_item                  (177)
#define  Pfn_interpret_item             (178)
#define  Pfn_set_err_hand_mode          (179)
#define  Pfn_escape                     (180)
#define  Pfn_set_err_hand               (181)


#define  PLINE_SOLID                    (1)
#define  PLINE_DASH                     (2)
#define  PLINE_DOT                      (3)
#define  PLINE_DASH_DOT                 (4)



#define  PMARKER_DOT                    (1)
```

```
#define  PMARKER_PLUS                    (2)
#define  PMARKER_ASTERISK                (3)
#define  PMARKER_CIRCLE                  (4)
#define  PMARKER_CROSS                   (5)


#define PANNO_STYLE_UNCONNECTED          (1)
#define PANNO_STYLE_LEAD_LINE            (2)


#define  PMODEL_RGB                      (1)
#define  PMODEL_CIELUV                   (2)
#define  PMODEL_HSV                      (3)
#define  PMODEL_HLS                      (4)


#define  PLOC_DEF                        (1)
#define  PLOC_CROSS_HAIR                 (2)
#define  PLOC_TRACK_CROSS                (3)
#define  PLOC_RUB_BAND                   (4)
#define  PLOC_RECT                       (5)
#define  PLOC_DIGIT                      (6)

#define  PSTROKE_DEF                     (1)
#define  PSTROKE_DIGIT                   (2)
#define  PSTROKE_MARKER                  (3)
#define  PSTROKE_LINE                    (4)

#define  PVAL_DEF                        (1)
#define  PVAL_GRAPH                      (2)
#define  PVAL_DIGIT                      (3)

#define  PCHOICE_DEF                     (1)
#define  PCHOICE_PR_ECHO                 (2)
#define  PCHOICE_STRING_PR               (3)
#define  PCHOICE_STRING_IN               (4)
#define  PCHOICE_STRUCT                  (5)

#define  PPICK_DEF                       (1)
#define  PPICK_GROUP_HIGHL               (2)
#define  PPICK_STRUCT_NETWORK            (3)

#define PSTRING_DEF                      (1)


#define  PDEF_MEM_SIZE                   ((size_t) (-1))
#define  PDEF_ERR_FILE                   ((char *) (0))


#define  PFIRST_PHIGS_ELEM               (PELEM_POLYLINE3)
```

```
#define  PLAST_PHIGS_ELEM                    (PELEM_PICK_ID)
```

A.2 Types in compilation order

```
typedef float    Pfloat;
typedef int  Pint;


typedef Pfloat Pmatrix3[4][4];


typedef Pfloat Pmatrix[3][3];


typedef void *Pstore;


typedef enum {
    PWS_INDEP,
    PWS_DEP
} Pws_dep_ind;


typedef enum {
    PSYS_ST_PHCL,
    PSYS_ST_PHOP
} Psys_st;


typedef enum {
    PWS_ST_WSCL,
    PWS_ST_WSOP
} Pws_st;


typedef enum {
    PSTRUCT_ST_STCL,
    PSTRUCT_ST_STOP
} Pstruct_st;


typedef enum {
    PSTRUCT_STATUS_NON_EXISTENT,
    PSTRUCT_STATUS_EMPTY,
    PSTRUCT_STATUS_NOT_EMPTY
} Pstruct_status;
```

```
typedef enum {
    PST_ARCL,
    PST_AROP
} Par_st;


typedef enum {
    PCLASS_VEC,
    PCLASS_RASTER,
    PCLASS_OTHER
} Pws_class;


typedef enum {
    PCAT_OUT,
    PCAT_IN,
    PCAT_OUTIN,
    PCAT_MO,
    PCAT_MI
} Pws_cat;


typedef enum {
    PFLAG_COND,
    PFLAG_ALWAYS
} Pctrl_flag;


typedef enum {
    PFLAG_POSTPONE,
    PFLAG_PERFORM
} Pregen_flag;


typedef enum {
    PDEFER_ASAP,
    PDEFER_BNIG,
    PDEFER_BNIL,
    PDEFER_ASTI,
    PDEFER_WAIT
} Pdefer_mode;


typedef enum {
    PMODE_NIVE,
    PMODE_UWOR,
    PMODE_UQUM
} Pmod_mode;
```

```
typedef enum {
    PSIMULT_NO_MORE,
    PSIMULT_MORE
} Pmore_simult_events;


typedef enum {
    PNET_CSS,
    PNET_AR
} Pstruct_net_source;


typedef enum {
    PSURF_NOT_EMPTY,
    PSURF_EMPTY
} Pdisp_surf_empty;


typedef enum {
    PVISUAL_ST_CORRECT,
    PVISUAL_ST_DEFER,
    PVISUAL_ST_SIMULATED
} Pvisual_st;


typedef enum {
    PPREC_STRING,
    PPREC_CHAR,
    PPREC_STROKE
} Ptext_prec;


typedef enum {
    PPATH_RIGHT,
    PPATH_LEFT,
    PPATH_UP,
    PPATH_DOWN
} Ptext_path;


typedef enum {
    PHOR_NORM,
    PHOR_LEFT,
    PHOR_CTR,
    PHOR_RIGHT
} Phor_text_align;
```

```
typedef enum {
    PVERT_NORM,
    PVERT_TOP,
    PVERT_CAP,
    PVERT_HALF,
    PVERT_BASE,
    PVERT_BOTTOM
} Pvert_text_align;


typedef enum {
    PSTYLE_HOLLOW,
    PSTYLE_SOLID,
    PSTYLE_PAT,
    PSTYLE_HATCH,
    PSTYLE_EMPTY
} Pint_style;


typedef enum {
    PEDGE_OFF,
    PEDGE_ON
} Pedge_flag;


typedef enum {
    PASPECT_LINETYPE,
    PASPECT_LINEWIDTH,
    PASPECT_LINE_COLR_IND,
    PASPECT_MARKER_TYPE,
    PASPECT_MARKER_SIZE,
    PASPECT_MARKER_COLR_IND,
    PASPECT_TEXT_FONT,
    PASPECT_TEXT_PREC,
    PASPECT_CHAR_EXPAN,
    PASPECT_CHAR_SPACE,
    PASPECT_TEXT_COLR_IND,
    PASPECT_INT_STYLE,
    PASPECT_INT_STYLE_IND,
    PASPECT_INT_COLR_IND,
    PASPECT_EDGE_FLAG,
    PASPECT_EDGETYPE,
    PASPECT_EDGEWIDTH,
    PASPECT_EDGE_COLR_IND
} Paspect;


typedef enum {
    PASF_BUNDLED,
    PASF_INDIV
} Pasf;
```

```
typedef enum {
    PAVAIL_MONOCHR,
    PAVAIL_COLR
} Pcolr_avail;


typedef enum {
    PTYPE_PRECONCAT,
    PTYPE_POSTCONCAT,
    PTYPE_REPLACE
} Pcompose_type;


typedef enum {
    PTYPE_PARAL,
    PTYPE_PERSPECT
} Pproj_type;


typedef enum {
    PIND_NO_CLIP,
    PIND_CLIP
} Pclip_ind;


typedef enum {
    PPRI_HIGHER,
    PPRI_LOWER
} Prel_pri;


typedef enum {
    PRES_MAINTAIN,
    PRES_ABANDON,
    PRES_UPD
} Pconf_res;


typedef enum {
    PFLAG_LINE,
    PFLAG_FILL,
    PFLAG_FILL_SET
} Pline_fill_ctrl_flag;


typedef enum {
    PORDER_TOP_FIRST,
    PORDER_BOTTOM_FIRST
} Ppath_order;
```

```
typedef enum {
    POP_REQ,
    POP_SAMPLE,
    POP_EVENT
} Pop_mode;


typedef enum {
    PSWITCH_NO_ECHO,
    PSWITCH_ECHO
} Pecho_switch;


typedef enum {
    PIN_STATUS_NONE,
    PIN_STATUS_OK,
    PIN_STATUS_NO_IN
} Pin_status;


typedef enum {
    PSTRUCT_NONE,
    PSTRUCT_OPEN
} Popen_struct_status;


typedef enum {
    PIN_NONE,
    PIN_LOC,
    PIN_STROKE,
    PIN_VAL,
    PIN_CHOICE,
    PIN_PICK,
    PIN_STRING
} Pin_class;


typedef enum {
    PPR_OFF,
    PPR_ON
} Ppr_switch;


typedef enum {
    PINQ_SET,
    PINQ_REALIZED
} Pinq_type;
```

```
typedef enum {
    PUPD_NOT_PEND,
    PUPD_PEND
} Pupd_st;


typedef enum {
    PDC_METRES,
    PDC_OTHER
} Pdc_units;


typedef enum {
    PDYN_IRG,
    PDYN_IMM,
    PDYN_CBS
} Pdyn_mod;


typedef enum {
    PATTR_LINE,
    PATTR_MARKER,
    PATTR_TEXT,
    PATTR_INT,
    PATTR_EDGE
} Pattrs;
```

```
typedef enum {
    PELEM_ALL,
    PELEM_NIL,
    PELEM_POLYLINE3,
    PELEM_POLYLINE,
    PELEM_POLYMARKER3,
    PELEM_POLYMARKER,
    PELEM_TEXT3,
    PELEM_TEXT,
    PELEM_ANNO_TEXT_REL3,
    PELEM_ANNO_TEXT_REL,
    PELEM_FILL_AREA3,
    PELEM_FILL_AREA,
    PELEM_FILL_AREA_SET3,
    PELEM_FILL_AREA_SET,
    PELEM_CELL_ARRAY3,
    PELEM_CELL_ARRAY,
    PELEM_GDP3,
    PELEM_GDP,
    PELEM_LINE_IND,
    PELEM_MARKER_IND,
    PELEM_TEXT_IND,
    PELEM_INT_IND,
    PELEM_EDGE_IND,
    PELEM_LINETYPE,
    PELEM_LINEWIDTH,
    PELEM_LINE_COLR_IND,
    PELEM_MARKER_TYPE,
    PELEM_MARKER_SIZE,
    PELEM_MARKER_COLR_IND,
    PELEM_TEXT_FONT,
    PELEM_TEXT_PREC,
    PELEM_CHAR_EXPAN,
    PELEM_CHAR_SPACE,
    PELEM_TEXT_COLR_IND,
    PELEM_CHAR_HT,
    PELEM_CHAR_UP_VEC,
    PELEM_TEXT_PATH,
    PELEM_TEXT_ALIGN,
    PELEM_ANNO_CHAR_HT,
    PELEM_ANNO_CHAR_UP_VEC,
    PELEM_ANNO_PATH,
    PELEM_ANNO_ALIGN,
    PELEM_ANNO_STYLE,
    PELEM_INT_STYLE,
    PELEM_INT_STYLE_IND,
    PELEM_INT_COLR_IND,
    PELEM_EDGE_FLAG,
    PELEM_EDGETYPE,
    PELEM_EDGEWIDTH,
    PELEM_EDGE_COLR_IND,
    PELEM_PAT_SIZE,
    PELEM_PAT_REF_POINT_VECS,
```

```
    PELEM_PAT_REF_POINT,
    PELEM_ADD_NAMES_SET,
    PELEM_REMOVE_NAMES_SET,
    PELEM_INDIV_ASF,
    PELEM_HLHSR_ID,
    PELEM_LOCAL_MODEL_TRAN3,
    PELEM_LOCAL_MODEL_TRAN,
    PELEM_GLOBAL_MODEL_TRAN3,
    PELEM_GLOBAL_MODEL_TRAN,
    PELEM_MODEL_CLIP_VOL3,
    PELEM_MODEL_CLIP_VOL,
    PELEM_MODEL_CLIP_IND,
    PELEM_RESTORE_MODEL_CLIP_VOL,
    PELEM_VIEW_IND,
    PELEM_EXEC_STRUCT,
    PELEM_LABEL,
    PELEM_APPL_DATA,
    PELEM_GSE,
    PELEM_PICK_ID
} Pelem_type;


typedef enum {
    PEDIT_INSERT,
    PEDIT_REPLACE
} Pedit_mode;


typedef enum {
    PFLAG_DEL,
    PFLAG_KEEP
} Pref_flag;


typedef enum {
    PERR_OFF,
    PERR_ON
} Perr_mode;


typedef enum {
    PDIR_BACKWARD,
    PDIR_FORWARD
} Psearch_dir;


typedef enum {
    PSEARCH_STATUS_FAILURE,
    PSEARCH_STATUS_SUCCESS
} Psearch_status;
```

```
typedef struct {
  size_t  size;       /* sizeof data     */
  void    *data;      /* pointer to data */
} Pdata;


typedef struct {
  Pfloat  cieluv_x;          /* x coefficient */
  Pfloat  cieluv_y;          /* y coefficient */
  Pfloat  cieluv_y_lum;      /* y luminance   */
} Pcieluv;


typedef struct {
  Pfloat  hue;            /* hue        */
  Pfloat  lightness;      /* lightness  */
  Pfloat  satur;          /* saturation */
} Phls;


typedef struct {
  Pfloat  hue;        /* hue        */
  Pfloat  satur;      /* saturation */
  Pfloat  value;      /* value      */
} Phsv;


typedef struct {
  Pfloat  red;        /* red intensity   */
  Pfloat  green;      /* green intensity */
  Pfloat  blue;       /* blue intensity  */
} Prgb;


typedef union {
  Prgb          rgb;        /* Red Green Blue colour specification           */
  Pcieluv       cieluv;     /* CIE L*u*v* colour specification               */
  Phls          hls;        /* Hue Lightness Saturation colour specification */
  Phsv          hsv;        /* Hue Saturation Value colour specification     */
  Pdata         unsupp;     /* colour in an unsupported colour model         */
  int impl_dep;             /* implementation defined                        */
} Pcolr_rep;
```

```
typedef struct {
  Pint  num_ints;      /* number of Pints in list */
  Pint  *ints;         /* list of integers        */
} Pint_list;


typedef struct {
  Pint         id;       /* GSE identifier                                  */
  Pws_dep_ind  ind;      /* workstation independent/dependent indicator */
} Pgse_id_dep;


typedef struct {
  Pint         num_id_facs;      /* number of identifiers/dependency element */
  Pgse_id_dep  *id_facs;         /* list of GSE facilities                   */
} Pgse_id_dep_list;


typedef struct {
  Pint   id;         /* archive file identifier */
  char   *name;      /* archive file name       */
} Par_file;


typedef struct {
  Pint      num_ar_files;      /* number of archive files */
  Par_file  *ar_files;         /* list of archive files   */
} Par_file_list;


typedef struct {
  Pint  struct_id;      /* structure identifier */
  Pint  elem_pos;       /* element position     */
} Pelem_ref;


typedef struct {
  Pint       num_elem_refs;      /* number of element references */
  Pelem_ref  *elem_refs;         /* list of element references   */
} Pelem_ref_list;
```

```
typedef struct {
  Pint            num_elem_ref_lists;     /* number of element references lists */
  Pelem_ref_list  *elem_ref_lists;        /* list of element reference lists    */
} Pelem_ref_list_list;


typedef struct {
  Pint    id;           /* structure id     */
  Pfloat  disp_pri;     /* display priority */
} Pposted_struct;


typedef struct {
  Pint            num_postings;     /* number of structure postings */
  Pposted_struct  *postings;        /* list of postings             */
} Pposted_struct_list;


typedef struct {
  Pfloat  x;      /* x coordinate */
  Pfloat  y;      /* y coordinate */
} Ppoint;


typedef struct {
  Pfloat  x;      /* x coordinate */
  Pfloat  y;      /* y coordinate */
  Pfloat  z;      /* z coordinate */
} Ppoint3;


typedef struct {
  Pfloat  delta_x;      /* delta x value */
  Pfloat  delta_y;      /* delta y value */
} Pvec;


typedef struct {
  Pfloat  delta_x;      /* delta x value */
  Pfloat  delta_y;      /* delta y value */
  Pfloat  delta_z;      /* delta z value */
} Pvec3;
```

```
typedef struct {
  Pint  size_x;      /* x size */
  Pint  size_y;      /* y size */
} Pint_size;


typedef struct {
  Pint  size_x;      /* x size */
  Pint  size_y;      /* y size */
  Pint  size_z;      /* z size */
} Pint_size3;


typedef struct {
  Pfloat  size_x;      /* x size */
  Pfloat  size_y;      /* y size */
} Pfloat_size;


typedef struct {
  Pfloat  size_x;      /* x size */
  Pfloat  size_y;      /* y size */
  Pfloat  size_z;      /* z size */
} Pfloat_size3;


typedef struct {
  Ppoint  point;      /* point  */
  Pvec    norm;       /* normal */
} Phalf_space;


typedef struct {
  Ppoint3  point;      /* point  */
  Pvec3    norm;       /* normal */
} Phalf_space3;


typedef struct {
  Ppoint  p;      /* point p */
  Ppoint  q;      /* point q */
} Prect;
```

```
typedef struct {
  Ppoint3  p;      /* point p */
  Ppoint3  q;      /* point q */
  Ppoint3  r;      /* point r */
} Pparal;


typedef struct {
  Pfloat  x_min;      /* x min */
  Pfloat  x_max;      /* x max */
  Pfloat  y_min;      /* y min */
  Pfloat  y_max;      /* y max */
  Pfloat  z_min;      /* z min */
  Pfloat  z_max;      /* z max */
} Plimit3;


typedef struct {
  Pfloat  x_min;      /* x min */
  Pfloat  x_max;      /* x max */
  Pfloat  y_min;      /* y min */
  Pfloat  y_max;      /* y max */
} Plimit;


typedef struct {
  Pint    num_points;     /* number of Ppoints in the list */
  Ppoint  *points;        /* list of points                */
} Ppoint_list;


typedef struct {
  Pint     num_points;     /* number of Ppoint3s in the list */
  Ppoint3  *points;        /* list of points                 */
} Ppoint_list3;


typedef struct {
  Pint         num_point_lists;     /* number of point lists */
  Ppoint_list  *point_lists;        /* list of point lists   */
} Ppoint_list_list;
```

```
typedef struct {
  Pint          num_point_lists;      /* number of point lists */
  Ppoint_list3  *point_lists;         /* list of point lists   */
} Ppoint_list_list3;


typedef struct {
  Pint         num_half_spaces;      /* number of half spaces */
  Phalf_space  *half_spaces;         /* list of half spaces   */
} Phalf_space_list;


typedef struct {
  Pint          num_half_spaces;      /* number of half spaces */
  Phalf_space3  *half_spaces;         /* list of half spaces   */
} Phalf_space_list3;


typedef struct {
  Pint_list  incl_set;      /* inclusion set */
  Pint_list  excl_set;      /* exclusion set */
} Pfilter;


typedef struct {
  Pint     num_filters;      /* number of filters */
  Pfilter  *filters;         /* list of filters   */
} Pfilter_list;


typedef struct {
  Phor_text_align  hor;       /* horizontal component */
  Pvert_text_align vert;      /* vertical component   */
} Ptext_align;


typedef struct {
  Pint    type;          /* linetype                */
  Pfloat  width;         /* linewidth scale factor */
  Pint    colr_ind;      /* colour index            */
} Pline_bundle;
```

```
typedef struct {
  Pint    type;          /* marker type                */
  Pfloat  size;          /* marker size scale factor */
  Pint    colr_ind;      /* colour index               */
} Pmarker_bundle;


typedef struct {
  Pint        font;           /* text font                    */
  Ptext_prec  prec;           /* text precision               */
  Pfloat      char_expan;     /* character expansion factor */
  Pfloat      char_space;     /* character spacing            */
  Pint        colr_ind;       /* text colour index            */
} Ptext_bundle;


typedef struct {
  Pint_style  style;          /* interior style       */
  Pint        style_ind;      /* interior style index */
  Pint        colr_ind;       /* interior colour index */
} Pint_bundle;


typedef struct {
  Pedge_flag  flag;          /* edge flag              */
  Pint        type;          /* edgetype               */
  Pfloat      width;         /* edgewidth scale factor */
  Pint        colr_ind;      /* edge colour index      */
} Pedge_bundle;


typedef struct {
  Pint_size  dims;            /* pattern's dimensions */
  Pint       *colr_array;     /* colour index array   */
} Ppat_rep;


typedef struct {
  Pmatrix3   ori_matrix;      /* orientation matrix       */
  Pmatrix3   map_matrix;      /* mapping matrix           */
  Plimit3    clip_limit;      /* clipping limits          */
  Pclip_ind  xy_clip;         /* X-Y clipping indicator   */
  Pclip_ind  back_clip;       /* back clipping indicator  */
  Pclip_ind  front_clip;      /* front clipping indicator */
} Pview_rep3;
```

```
typedef struct {
  Pmatrix    ori_matrix;      /* orientation matrix     */
  Pmatrix    map_matrix;      /* mapping matrix         */
  Plimit     clip_limit;      /* clipping limits        */
  Pclip_ind  xy_clip;         /* X-Y clipping indicator */
} Pview_rep;
```

```
typedef struct {
  Plimit      win;                /* window limits              */
  Plimit3     proj_vp;            /* projection viewport limits */
  Pproj_type  proj_type;          /* projection type            */
  Ppoint3     proj_ref_point;     /* projection reference point */
  Pfloat      view_plane;         /* view plane distance        */
  Pfloat      back_plane;         /* back plane distance        */
  Pfloat      front_plane;        /* front plane distance       */
} Pview_map3;
```

```
typedef struct {
  Plimit  win;          /* window limits           */
  Plimit  proj_vp;      /* projection viewport limits */
} Pview_map;
```

```
typedef struct {
  Pasf          type_asf;         /* linetype asf          */
  Pasf          width_asf;        /* linewidth asf         */
  Pasf          colr_ind_asf;     /* line colour index asf */
  Pint          ind;              /* line index            */
  Pline_bundle  bundle;           /* line bundle           */
} Pline_attrs;
```

```
typedef struct {
  Pasf        style_asf;          /* interior style asf        */
  Pasf        style_ind_asf;      /* interior style index asf  */
  Pasf        colr_ind_asf;       /* interior colour index asf */
  Pint        ind;                /* interior index            */
  Pint_bundle bundle;             /* interior bundle           */
} Pint_attrs;
```

```
typedef struct {
  Pasf          flag_asf;         /* edge flag asf         */
  Pasf          type_asf;         /* edgetype asf          */
  Pasf          width_asf;        /* edgewidth asf         */
  Pasf          colr_ind_asf;     /* edge colour index asf */
  Pint          ind;              /* edge index            */
  Pedge_bundle  bundle;           /* edge bundle           */
} Pedge_attrs;


typedef struct {
  Pasf            type_asf;         /* marker type asf                */
  Pasf            size_asf;         /* marker size scale factor asf */
  Pasf            colr_ind_asf;     /* marker colour index asf      */
  Pint            ind;              /* marker index                   */
  Pmarker_bundle  bundle;           /* marker bundle                  */
} Pmarker_attrs;


typedef struct {
  Pint  struct_id;      /* structure identifier */
  Pint  pick_id;        /* pick identifier      */
  Pint  elem_pos;       /* element position     */
} Ppick_path_elem;


typedef struct {
  Pint             depth;          /* pick path depth */
  Ppick_path_elem  *path_list;     /* pick path list  */
} Ppick_path;


typedef struct {
  Pdc_units    dc_units;        /* device coordinate units         */
  Pfloat_size  size_dc;         /* device size in coordinate units */
  Pint_size    size_raster;     /* device size in raster units     */
} Pdisp_space_size;


typedef struct {
  Pdc_units     dc_units;         /* device coordinate units           */
  Pfloat_size3  size_dc;          /* device volume in coordinate units */
  Pint_size3    size_raster;      /* device volume in raster units     */
} Pdisp_space_size3;
```

```
typedef struct {

  Pdyn_mod  line_bundle;        /* polyline bundle representation   */
  Pdyn_mod  marker_bundle;      /* polymarker bundle representation */
  Pdyn_mod  text_bundle;        /* text bundle representation       */
  Pdyn_mod  int_bundle;         /* interior bundle representation   */
  Pdyn_mod  edge_bundle;        /* edge bundle representation       */
  Pdyn_mod  pat_rep;            /* pattern representation           */
  Pdyn_mod  colr_rep;           /* colour representation            */
  Pdyn_mod  view_rep;           /* view representation              */
  Pdyn_mod  ws_tran;            /* workstation transformation       */
  Pdyn_mod  highl_filter;       /* highlighting filter              */
  Pdyn_mod  invis_filter;       /* invisibility filter              */
  Pdyn_mod  hlhsr_mode;         /* HLHSR mode                       */

} Pdyns_ws_attrs;


typedef struct {

  Pint_list  types;             /* list of linetypes                    */
  Pint       num_widths;        /* number of available linewidths       */
  Pfloat     nom_width;         /* nominal linewidth                    */
  Pfloat     min_width;         /* min linewidth                        */
  Pfloat     max_width;         /* max linewidth                        */
  Pint       num_pred_inds;     /* number of predefined bundle indices */

} Pline_facs;


typedef struct {

  Pint_list  types;             /* list of marker types                 */
  Pint       num_sizes;         /* number of available marker sizes     */
  Pfloat     nom_size;          /* nominal marker size                  */
  Pfloat     min_size;          /* min marker size                      */
  Pfloat     max_size;          /* max marker size                      */
  Pint       num_pred_inds;     /* number of predefined bundle indices */

} Pmarker_facs;


typedef struct {

  Pint       font;      /* text font      */
  Ptext_prec prec;      /* text precision */

} Ptext_font_prec;
```

```
typedef struct {
  Pint             num_font_precs;      /* number of fonts and precisions        */
  Ptext_font_prec  *font_precs;         /* list of fonts and precisions          */
  Pint             num_char_hts;        /* number of character heights           */
  Pfloat           min_char_ht;         /* minimum height                        */
  Pfloat           max_char_ht;         /* maximum height                        */
  Pint             num_char_expans;     /* number of character expansion factors */
  Pfloat           min_char_expan;      /* minimum expansion factor              */
  Pfloat           max_char_expan;      /* maximum expansion factor              */
  Pint             num_pred_inds;       /* number of predefined bundle indices   */
} Ptext_facs;


typedef struct {
  Pint        num_int_styles;     /* number of interior styles           */
  Pint_style  int_styles[5];      /* list of available interior styles   */
  Pint_list   hatch_styles;       /* list of available hatch styles      */
  Pint        num_pred_inds;      /* number of predefined bundle indices */
} Pint_facs;


typedef struct {
  Pint_list  types;              /* list of edgetypes                     */
  Pint       num_widths;         /* number of available edgewidths        */
  Pfloat     nom_width;          /* nominal edgewidth                     */
  Pfloat     min_width;          /* min edgewidth                         */
  Pfloat     max_width;          /* max edgewidth                         */
  Pint       num_pred_inds;      /* number of predefined bundle indices */
} Pedge_facs;


typedef struct {
  Pint         num_colrs;          /* number of colours                    */
  Pcolr_avail  colr_avail;         /* colour availability                  */
  Pint         num_pred_inds;      /* number of predefined colour indices  */
  Pcieluv      prim_colrs[3];      /* primary colours                      */
} Pcolr_facs;
```

```
typedef struct {

  Pint  line_bundles;      /* max. # of polyline table entries   */
  Pint  mark_bundles;      /* max. # of polymarker table entries */
  Pint  text_bundles;      /* max. # of text table entries       */
  Pint  int_bundles;       /* max. # of interior table entries   */
  Pint  edge_bundles;      /* max. # of edge table entries       */
  Pint  pat_reps;          /* max. # of pattern table entries    */
  Pint  colr_reps;         /* max. # of colour table entries     */
  Pint  view_reps;         /* max. # of view table entries       */

} Pws_st_tables;


typedef struct {

  Pdyn_mod  content;      /* structure content    */
  Pdyn_mod  post;         /* post structure       */
  Pdyn_mod  unpost;       /* unpost structure     */
  Pdyn_mod  del;          /* delete structure     */
  Pdyn_mod  ref;          /* structure reference  */

} Pdyns_structs;


typedef struct {

  Pint  loc;          /* locators  */
  Pint  stroke;       /* strokes   */
  Pint  val;          /* valuators */
  Pint  choice;       /* choices   */
  Pint  pick;         /* picks     */
  Pint  string;       /* strings   */

} Pnum_in;


typedef struct {

  Pint        num_elem_types;      /* number of element types */
  Pelem_type  *elem_types;         /* list of element types   */

} Pelem_type_list;


typedef union {

  int impl_dep;              /* Metafile Records                */
  Pdata          unsupp;     /* unsupported Metafile item data */

} Pitem_data;
```

```
typedef struct {
  union Ploc_pets {
    struct Ploc_pet_r1 {
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r1;
    struct Ploc_pet_r2 {
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r2;
    struct Ploc_pet_r3 {
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r3;
    struct Ploc_pet_r4 {
                  Pline_attrs          line_attrs;               /* polyline attributes   */
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r4;
    struct Ploc_pet_r5 {
                  Pline_fill_ctrl_flag line_fill_ctrl_flag;      /* control flag          */
                  union Ploc_attrs {
                                       Pline_attrs               line_attrs;
                                                                 /* polyline attributes   */
                                       Pint_attrs                int_attrs;
                                                                 /* interior attributes   */
                                       struct Ploc_fill_set {
                                                                 Pint_attrs  int_attrs;
                                                                 /* interior attributes   */
                                                                 Pedge_attrs  edge_attrs;
                                                                 /* edge attributes       */
                                       } fill_set;
                  } attrs;
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r5;
    struct Ploc_pet_r6 {
                  int impl_dep;                                  /* impl. dependent       */
    } pet_r6;
    int impl_dep;                                                /* impl. defined PET's   */
  } pets;
} Ploc_data;
```

```
typedef struct {
  Pint    in_buf_size;                                  /* input buffer size            */
  Pint    init_pos;                                     /* initial editing position     */
  Pfloat  x_interval;                                   /* x interval                   */
  Pfloat  y_interval;                                   /* y interval                   */
  Pfloat  time_interval;                                /* time interval                */
  union Pstroke_pets {
          struct Pstroke_pet_r1 {
                          int impl_dep;                 /* implementation dependent     */
          } pet_r1;
          struct Pstroke_pet_r2 {
                          int impl_dep;                 /* implementation dependent     */
          } pet_r2;
          struct Pstroke_pet_r3 {
                          Pmarker_attrs  marker_attrs;  /* marker attributes            */
                          int impl_dep;                 /* implementation dependent     */
          } pet_r3;
          struct Pstroke_pet_r4 {
                          Pline_attrs    line_attrs;    /* line attributes              */
                          int impl_dep;                 /* implementation dependent     */
          } pet_r4;
          int impl_dep;                                 /* implementation defined PET's */
  } pets;
} Pstroke_data;


typedef struct {
  Pfloat  low_value;                    /* low value of valuator range  */
  Pfloat  high_value;                   /* high value of valuator range */
  union Pval_pets {
          struct Pval_pet_r1 {
                          int impl_dep; /* implementation dependent     */
          } pet_r1;
          int impl_dep;                 /* implementation defined PET's */
  } pets;
} Pval_data;
```

```
typedef struct {
  union Pchoice_pets {
    struct Pchoice_pet_r1 {
                int impl_dep;                   /* implementation dependent     */
    } pet_r1;
    struct Pchoice_pet_r2 {
                Pint            num_prompts;    /* number of prompts            */
                Ppr_switch      *prompts;       /* array of prompts             */
                int impl_dep;                   /* implementation dependent     */
    } pet_r2;
    struct Pchoice_pet_r3 {
                Pint            num_strings;    /* number of choice strings     */
                char            **strings;      /* array of choice strings      */
                int impl_dep;                   /* implementation dependent     */
    } pet_r3;
    struct Pchoice_pet_r4 {
                Pint            num_strings;    /* number of choice strings     */
                char            **strings;      /* array of choice strings      */
                int impl_dep;                   /* implementation dependent     */
    } pet_r4;
    struct Pchoice_pet_r5 {
                Pint            struct_id;      /* struct identifier            */
                Pint            num_pick_ids;   /* number of pick identifiers   */
                Pint            *pick_ids;      /* array of pick identifiers    */
                int impl_dep;                   /* implementation dependent     */
    } pet_r5;
    int impl_dep;                               /* implementation defined PET's */
  } pets;
} Pchoice_data;


typedef struct {
  union Ppick_pets {
    struct Ppick_pet_r1 {
                int impl_dep;    /* implementation dependent     */
    } pet_r1;
    int impl_dep;                /* implementation defined PET's */
  } pets;
} Ppick_data;
```

```
typedef struct {
  Pint  in_buf_size;                    /* input buffer size            */
  Pint  init_pos;                       /* initial editing position     */
  union Pstring_pets {
        struct Pstring_pet_r1 {
                    int impl_dep;       /* implementation dependent     */
        } pet_r1;
        int impl_dep;                   /* implementation defined PET's */
  } pets;
} Pstring_data;


typedef struct {
  union Ploc3_pets {
    struct Ploc3_pet_r1 {
                int impl_dep;                                   /* impl. dependent      */
    } pet_r1;
    struct Ploc3_pet_r2 {
                int impl_dep;                                   /* impl. dependent      */
    } pet_r2;
    struct Ploc3_pet_r3 {
                int impl_dep;                                   /* impl. dependent      */
    } pet_r3;
    struct Ploc3_pet_r4 {
                Pline_attrs     line_attrs;                     /* polyline attributes  */
                int impl_dep;                                   /* impl. dependent      */
    } pet_r4;
    struct Ploc3_pet_r5 {
                Pline_fill_ctrl_flag line_fill_ctrl_flag;       /* control flag         */
                union Ploc3_attrs {
                                Pline_attrs     line_attrs;
                                                /* polyline attributes  */
                                Pint_attrs      int_attrs;
                                                /* interior attributes  */
                                struct Ploc3_fill_set {
                                                Pint_attrs  int_attrs;
                                                /* interior attributes  */
                                                Pedge_attrs  edge_attrs;
                                                /* edge attributes      */
                                } fill_set;
                } attrs;
                int impl_dep;                                   /* impl. dependent      */
    } pet_r5;
    struct Ploc3_pet_r6 {
                int impl_dep;                                   /* impl. dependent      */
    } pet_r6;
    int impl_dep;                                               /* impl. defined PET's  */
  } pets;
} Ploc_data3;
```

```
typedef struct {
  Pint    in_buf_size;                                  /* input buffer size            */
  Pint    init_pos;                                     /* initial editing position     */
  Pfloat  x_interval;                                   /* x interval                   */
  Pfloat  y_interval;                                   /* y interval                   */
  Pfloat  z_interval;                                   /* z interval                   */
  Pfloat  time_interval;                                /* time interval                */
  union Pstroke3_pets {
          struct Pstroke3_pet_r1 {
                          int impl_dep;                 /* implementation dependent     */
          } pet_r1;
          struct Pstroke3_pet_r2 {
                          int impl_dep;                 /* implementation dependent     */
          } pet_r2;
          struct Pstroke3_pet_r3 {
                          Pmarker_attrs  marker_attrs;  /* marker attributes            */
                          int impl_dep;                 /* implementation dependent     */
          } pet_r3;
          struct Pstroke3_pet_r4 {
                          Pline_attrs    line_attrs;    /* line attributes              */
                          int impl_dep;                 /* implementation dependent     */
          } pet_r4;
          int impl_dep;                                 /* implementation defined PET's */
  } pets;
} Pstroke_data3;


typedef struct {
  Pfloat  low_value;                    /* low value of valuator range  */
  Pfloat  high_value;                   /* high value of valuator range */
  union Pval3_pets {
          struct Pval3_pet_r1 {
                          int impl_dep;    /* implementation dependent     */
          } pet_r1;
          int impl_dep;                 /* implementation defined PET's */
  } pets;
} Pval_data3;
```

```
typedef struct {
  union Pchoice3_pets {
    struct Pchoice3_pet_r1 {
                int impl_dep;                   /* implementation dependent     */
    } pet_r1;
    struct Pchoice3_pet_r2 {
                Pint            num_prompts;    /* number of prompts            */
                Ppr_switch      *prompts;       /* array of prompts             */
                int impl_dep;                   /* implementation dependent     */
    } pet_r2;
    struct Pchoice3_pet_r3 {
                Pint            num_strings;    /* number of choice strings     */
                char            **strings;      /* array of choice strings      */
                int impl_dep;                   /* implementation dependent     */
    } pet_r3;
    struct Pchoice3_pet_r4 {
                Pint            num_strings;    /* number of choice strings     */
                char            **strings;      /* array of choice strings      */
                int impl_dep;                   /* implementation dependent     */
    } pet_r4;
    struct Pchoice3_pet_r5 {
                Pint            struct_id;      /* struct identifier            */
                Pint            num_pick_ids;   /* number of pick identifiers   */
                Pint            *pick_ids;      /* array of pick identifiers    */
                int impl_dep;                   /* implementation dependent     */
    } pet_r5;
    int impl_dep;                               /* implementation defined PET's */
  } pets;
} Pchoice_data3;


typedef struct {
  union Ppick3_pets {
    struct Ppick3_pet_r1 {
                int impl_dep;    /* implementation dependent     */
    } pet_r1;
    int impl_dep;                /* implementation defined PET's */
  } pets;
} Ppick_data3;
```

```
typedef struct {
  Pint  in_buf_size;                    /* input buffer size            */
  Pint  init_pos;                       /* initial editing position     */
  union Pstring3_pets {
        struct Pstring3_pet_r1 {
                        int impl_dep;   /* implementation dependent     */
        } pet_r1;
        int impl_dep;                   /* implementation defined PET's */
  } pets;
} Pstring_data3;


typedef union {
  struct Pgdp_r1 {
                int impl_dep;       /* registration dependent       */
  } gdp_r1;
  Pdata         unsupp;             /* unsupported GDP data record */
  int impl_dep;                     /* implementation defined       */
} Pgdp_data;


typedef union {
  struct Pgdp3_r1 {
                int impl_dep;       /* registration dependent        */
  } gdp3_r1;
  Pdata         unsupp;             /* unsupported GDP3 data record */
  int impl_dep;                     /* implementation defined        */
} Pgdp_data3;


typedef union {
  struct Pgse_r1 {
                int impl_dep;       /* registration dependent       */
  } gse_r1;
  Pdata         unsupp;             /* unsupported GSE data record */
  int impl_dep;                     /* implementation defined       */
} Pgse_data;


typedef union {
  struct Pescape_out_r1 {
                int impl_dep;       /* registration dependent */
  } escape_out_r1;
  int impl_dep;                     /* implementation defined */
} Pescape_out_data;
```

```
typedef union {
  struct Pescape_in_r1 {
                int impl_dep;      /* registration dependent */
  } escape_in_r1;
  int impl_dep;                    /* implementation defined */
} Pescape_in_data;
```

```
typedef union {
  Pint              int_data;                    /* integer valued data     */
  Pfloat            float_data;                  /* float valued data       */
  Ppoint_list3      point_list3;                 /* list of 3d points       */
  Ppoint_list       point_list;                  /* list of 2d points       */
  Ppoint_list_list3 point_list_list3;            /* list of 3d point lists */
  Ppoint_list_list  point_list_list;             /* list of 2d point lists */
  struct Pelem_text3 {
                    Ppoint3         pos;         /* text position           */
                    Pvec3           dir[2];      /* direction vectors       */
                    char            *char_string; /* character string       */
  } text3;
  struct Pelem_text {
                    Ppoint          pos;         /* text position           */
                    char            *char_string; /* character string       */
  } text;
  struct Pelem_anno_text_rel3 {
                    Ppoint3         ref_point;   /* reference point         */
                    Pvec3           offset;      /* anno. offset            */
                    char            *char_string; /* character string       */
  } anno_text_rel3;
  struct Pelem_anno_text_rel {
                    Ppoint          ref_point;   /* reference point         */
                    Pvec            offset;      /* anno. offset            */
                    char            *char_string; /* character string       */
  } anno_text_rel;
  struct Pelem_cell_array3 {
                    Pparal          paral;       /* parallelogram           */
                    Ppat_rep        colr_array;  /* colour array            */
  } cell_array3;
  struct Pelem_cell_array {
                    Prect           rect;        /* rectangle               */
                    Ppat_rep        colr_array;  /* colour array            */
  } cell_array;
  struct Pelem_gdp3 {
                    Pint            id;          /* GDP3 id                 */
                    Ppoint_list3    point_list;  /* point list              */
                    Pgdp_data3      data;        /* data record             */
  } gdp3;
  struct Pelem_gdp {
                    Pint            id;          /* GDP id                  */
                    Ppoint_list     point_list;  /* point list              */
                    Pgdp_data       data;        /* data record             */
  } gdp;
  Ptext_prec        text_prec;                   /* text precision          */
  Pvec              char_up_vec;                 /* char up vector          */
  Ptext_path        text_path;                   /* text path               */
  Ptext_align       text_align;                  /* text alignment          */
  Pint_style        int_style;                   /* interior style          */
  Pedge_flag        edge_flag;                   /* edge flag               */
  Ppoint            pat_ref_point;               /* pattern ref. point      */
  Pfloat_size       pat_size;                    /* pattern size            */
  struct Pelem_pat_ref_point_vecs {
```

```
                    Ppoint3             ref_point;      /* pattern ref. point     */
                    Pvec3               ref_vec[2];     /* vectors                */
  } pat_ref_point_vecs;
  Pint_list         names;                              /* name sets              */
  struct Pelem_asf {
                    Paspect             id;             /* attribute id           */
                    Pasf                source;         /* attribute source       */
  } asf;
  struct Pelem_local_tran3 {
                    Pcompose_type       compose_type;   /* composition type       */
                    Pmatrix3            matrix;         /* tran. matrix           */
  } local_tran3;
  struct Pelem_local_tran {
                    Pcompose_type       compose_type;   /* composition type       */
                    Pmatrix             matrix;         /* tran. matrix           */
  } local_tran;
  Pmatrix3          global_tran3;                       /* global transform3      */
  Pmatrix           global_tran;                        /* global transform       */
  struct Pelem_model_clip3 {
                    Pint                op;             /* operator               */
                    Phalf_space_list3   half_spaces;    /* half space list        */
  } model_clip3;
  struct Pelem_model_clip {
                    Pint                op;             /* operator               */
                    Phalf_space_list    half_spaces;    /* half space list        */
  } model_clip;
  Pclip_ind         clip_ind;                           /* clipping indicator     */
  Pdata             appl_data;                          /* application data       */
  struct Pelem_gse {
                    Pint                id;             /* GSE id                 */
                    Pgse_data           data;           /* GSE data record        */
  } gse;
} Pelem_data;
```

A.3 External functions

```
/* OPEN PHIGS */
extern void popen_phigs(
    const char    *err_file,   /* name of error file                                  */
    size_t        mem_units    /* size_t units of memory available for buffer space   */
);

/* CLOSE PHIGS */
extern void pclose_phigs(
      void
);
```

```
/* OPEN WORKSTATION */
extern void popen_ws(
    Pint         ws_id,      /* workstation identifier   */
    const void   *conn_id,   /* connection identifier    */
    Pint         ws_type     /* workstation type         */
);

/* CLOSE WORKSTATION */
extern void pclose_ws(
    Pint   ws_id   /* workstation identifier   */
);

/* REDRAW ALL STRUCTURES */
extern void predraw_all_structs(
    Pint         ws_id,      /* workstation identifier   */
    Pctrl_flag   ctrl_flag   /* control flag             */
);

/* UPDATE WORKSTATION */
extern void pupd_ws(
    Pint          ws_id,       /* workstation identifier     */
    Pregen_flag   regen_flag   /* update regeneration flag   */
);

/* SET DISPLAY UPDATE STATE */
extern void pset_disp_upd_st(
    Pint          ws_id,      /* workstation identifier   */
    Pdefer_mode   def_mode,   /* deferral mode            */
    Pmod_mode     mod_mode    /* modification mode        */
);

/* MESSAGE */
extern void pmessage(
    Pint          ws_id,     /* workstation identifier   */
    const char    *message   /* message string           */
);

/* POLYLINE 3 */
extern void ppolyline3(
    const Ppoint_list3   *point_list   /* list of points   */
);

/* POLYLINE */
extern void ppolyline(
    const Ppoint_list   *point_list   /* list of points   */
);

/* POLYMARKER 3 */
extern void ppolymarker3(
    const Ppoint_list3   *point_list   /* list of points   */
);
```

```
/* POLYMARKER */
extern void ppolymarker(
    const Ppoint_list   *point_list   /* list of points   */
);

/* TEXT 3 */
extern void ptext3(
    const Ppoint3   *text_pos,     /* text position           */
    const Pvec3     text_dir[2],   /* text direction vectors  */
    const char      *char_string   /* character string        */
);

/* TEXT */
extern void ptext(
    const Ppoint   *text_pos,     /* text position      */
    const char     *char_string   /* character string   */
);

/* ANNOTATION TEXT RELATIVE 3 */
extern void panno_text_rel3(
    const Ppoint3   *ref_point,    /* reference point              */
    const Pvec3     *offset,       /* annotation offset            */
    const char      *char_string   /* annotation character string  */
);

/* ANNOTATION TEXT RELATIVE */
extern void panno_text_rel(
    const Ppoint   *ref_point,    /* reference point              */
    const Pvec     *offset,       /* annotation offset            */
    const char     *char_string   /* annotation character string  */
);

/* FILL AREA 3 */
extern void pfill_area3(
    const Ppoint_list3   *point_list   /* list of points   */
);

/* FILL AREA */
extern void pfill_area(
    const Ppoint_list   *point_list   /* list of points   */
);

/* FILL AREA SET 3 */
extern void pfill_area_set3(
    const Ppoint_list_list3   *point_list_list   /* list of point lists   */
);

/* FILL AREA SET */
extern void pfill_area_set(
    const Ppoint_list_list   *point_list_list   /* list of point lists   */
);
```

```
/* CELL ARRAY 3 */
extern void pcell_array3(
    const Pparal     *paral,       /* cell parallelogram   */
    const Ppat_rep   *colr_array   /* colour array         */
);

/* CELL ARRAY */
extern void pcell_array(
    const Prect      *rect,        /* cell rectangle   */
    const Ppat_rep   *colr_array   /* colour array     */
);

/* GENERALIZED DRAWING PRIMITIVE 3 */
extern void pgdp3(
    const Ppoint_list3   *point_list,   /* list of points              */
    Pint                 gdp3_id,       /* gdp function identifier     */
    const Pgdp_data3     *gdp_data      /* gdp data record             */
);

/* GENERALIZED DRAWING PRIMITIVE */
extern void pgdp(
    const Ppoint_list   *point_list,   /* list of points            */
    Pint                gdp_id,        /* gdp function identifier   */
    const Pgdp_data     *gdp_data      /* gdp data record           */
);

/* SET POLYLINE INDEX */
extern void pset_line_ind(
    Pint   line_ind   /* polyline index   */
);

/* SET POLYMARKER INDEX */
extern void pset_marker_ind(
    Pint   marker_ind /* polymarker index   */
);

/* SET TEXT INDEX */
extern void pset_text_ind(
    Pint   text_ind   /* text index   */
);

/* SET INTERIOR INDEX */
extern void pset_int_ind(
    Pint   int_ind   /* interior index   */
);

/* SET EDGE INDEX */
extern void pset_edge_ind(
    Pint   edge_ind   /* edge index   */
);
```

```
/* SET LINETYPE */
extern void pset_linetype(
    Pint   linetype   /* linetype   */
);

/* SET LINEWIDTH SCALE FACTOR */
extern void pset_linewidth(
    Pfloat   linewidth   /* linewidth scale factor   */
);

/* SET POLYLINE COLOUR INDEX */
extern void pset_line_colr_ind(
    Pint   line_colr_ind   /* polyline colour index   */
);

/* SET MARKER TYPE */
extern void pset_marker_type(
    Pint   marker_type   /* marker type   */
);

/* SET MARKER SIZE SCALE FACTOR */
extern void pset_marker_size(
    Pfloat   marker_size   /* marker size scale factor   */
);

/* SET POLYMARKER COLOUR INDEX */
extern void pset_marker_colr_ind(
    Pint   marker_colr_ind   /* polymarker colour index   */
);

/* SET TEXT FONT */
extern void pset_text_font(
    Pint   font   /* text font   */
);

/* SET TEXT PRECISION */
extern void pset_text_prec(
    Ptext_prec   prec   /* text precision   */
);

/* SET CHARACTER EXPANSION FACTOR */
extern void pset_char_expan(
    Pfloat   char_expan   /* character expansion factor   */
);

/* SET CHARACTER SPACING */
extern void pset_char_space(
    Pfloat   char_space   /* character spacing   */
);
```

```
/* SET TEXT COLOUR INDEX */
extern void pset_text_colr_ind(
    Pint   text_colr_ind   /* text colour index   */
);

/* SET CHARACTER HEIGHT */
extern void pset_char_ht(
    Pfloat   char_ht   /* character height   */
);

/* SET CHARACTER UP VECTOR */
extern void pset_char_up_vec(
    const Pvec   *char_up_vec   /* character up vector   */
);

/* SET TEXT PATH */
extern void pset_text_path(
    Ptext_path   text_path   /* text path   */
);

/* SET TEXT ALIGNMENT */
extern void pset_text_align(
    const Ptext_align   *text_align   /* text alignment   */
);

/* SET ANNOTATION TEXT CHARACTER HEIGHT */
extern void pset_anno_char_ht(
    Pfloat   char_ht   /* character height   */
);

/* SET ANNOTATION TEXT CHARACTER UP VECTOR */
extern void pset_anno_char_up_vec(
    const Pvec   *char_up_vec   /* character up vector   */
);

/* SET ANNOTATION TEXT PATH */
extern void pset_anno_path(
    Ptext_path   text_path   /* text path   */
);

/* SET ANNOTATION TEXT ALIGNMENT */
extern void pset_anno_align(
    const Ptext_align   *text_align   /* text alignment   */
);

/* SET ANNOTATION STYLE */
extern void pset_anno_style(
    Pint   anno_style   /* annotation style   */
);
```

```
/* SET INTERIOR STYLE */
extern void pset_int_style(
    Pint_style   int_style   /* interior style   */
);

/* SET INTERIOR STYLE INDEX */
extern void pset_int_style_ind(
    Pint   int_style_ind   /* interior style index   */
);

/* SET INTERIOR COLOUR INDEX */
extern void pset_int_colr_ind(
    Pint   int_colr_ind   /* interior colour index   */
);

/* SET EDGE FLAG */
extern void pset_edge_flag(
    Pedge_flag   edge_flag   /* edge flag   */
);

/* SET EDGETYPE */
extern void pset_edgetype(
    Pint   edgetype   /* edgetype   */
);

/* SET EDGEWIDTH SCALE FACTOR */
extern void pset_edgewidth(
    Pfloat   edgewidth   /* edgewidth scale factor   */
);

/* SET EDGE COLOUR INDEX */
extern void pset_edge_colr_ind(
    Pint   edge_colr_ind   /* edge colour index   */
);

/* SET PATTERN SIZE */
extern void pset_pat_size(
    const Pfloat_size   *pat_size   /* pattern size   */
);

/* SET PATTERN REFERENCE POINT AND VECTORS */
extern void pset_pat_ref_point_vecs(
    const Ppoint3   *pat_ref_point,   /* pattern reference point                    */
    const Pvec3     pat_ref_vec[2]    /* reference vectors: 0 = X axis; 1 = Y axis  */
);

/* SET PATTERN REFERENCE POINT */
extern void pset_pat_ref_point(
    const Ppoint   *pat_ref_point   /* pattern reference point   */
);
```

```
/* ADD NAMES TO SET */
extern void padd_names_set(
    const Pint_list   *names   /* name set to be added   */
);

/* REMOVE NAMES FROM SET */
extern void premove_names_set(
    const Pint_list   *names   /* name set to be removed   */
);

/* SET INDIVIDUAL ASF */
extern void pset_indiv_asf(
    Paspect   asf_id,      /* attribute source identifier   */
    Pasf      asf_source   /* attribute source              */
);

/* SET POLYLINE REPRESENTATION */
extern void pset_line_rep(
    Pint                ws_id,         /* workstation identifier    */
    Pint                line_ind,      /* polyline bundle index     */
    const Pline_bundle  *line_bundle   /* polyline representation   */
);

/* SET POLYMARKER REPRESENTATION */
extern void pset_marker_rep(
    Pint                   ws_id,          /* workstation identifier      */
    Pint                   marker_ind,     /* polymarker bundle index     */
    const Pmarker_bundle   *marker_bundle  /* polymarker representation   */
);

/* SET TEXT REPRESENTATION */
extern void pset_text_rep(
    Pint                ws_id,         /* workstation identifier   */
    Pint                text_ind,      /* text bundle index        */
    const Ptext_bundle  *text_bundle   /* text representation      */
);

/* SET INTERIOR REPRESENTATION */
extern void pset_int_rep(
    Pint                ws_id,        /* workstation identifier    */
    Pint                int_ind,      /* interior bundle index     */
    const Pint_bundle   *int_bundle   /* interior representation   */
);

/* SET EDGE REPRESENTATION */
extern void pset_edge_rep(
    Pint                ws_id,         /* workstation identifier   */
    Pint                edge_ind,      /* edge bundle index        */
    const Pedge_bundle  *edge_bundle   /* edge representation      */
);
```

```
/* SET PATTERN REPRESENTATION */
extern void pset_pat_rep(
    Pint             ws_id,        /* workstation identifier   */
    Pint             pat_ind,      /* pattern bundle index     */
    const Ppat_rep   *pat_bundle   /* pattern representation   */
);

/* SET COLOUR REPRESENTATION */
extern void pset_colr_rep(
    Pint              ws_id,      /* workstation identifier   */
    Pint              colr_ind,   /* colour bundle index      */
    const Pcolr_rep   *colr_rep   /* colour representation    */
);

/* SET HIGHLIGHTING FILTER */
extern void pset_highl_filter(
    Pint            ws_id,    /* workstation identifier   */
    const Pfilter   *filter   /* highlighting filter      */
);

/* SET INVISIBILITY FILTER */
extern void pset_invis_filter(
    Pint            ws_id,    /* workstation identifier   */
    const Pfilter   *filter   /* invisibility filter      */
);

/* SET COLOUR MODEL */
extern void pset_colr_model(
    Pint   ws_id,       /* workstation identifier   */
    Pint   colr_model   /* colour model             */
);

/* SET HLHSR IDENTIFIER */
extern void pset_hlhsr_id(
    Pint   hlhsr_id   /* HLHSR identifier   */
);

/* SET HLHSR MODE */
extern void pset_hlhsr_mode(
    Pint   ws_id,       /* workstation identifier   */
    Pint   hlhsr_mode   /* HLHSR mode               */
);

/* SET LOCAL TRANSFORMATION 3 */
extern void pset_local_tran3(
    Pmatrix3        local_tran,    /* local transformation matrix   */
    Pcompose_type   compose_type   /* composition type              */
);
```

```
/* SET LOCAL TRANSFORMATION */
extern void pset_local_tran(
    Pmatrix         local_tran,    /* local transformation matrix   */
    Pcompose_type   compose_type   /* composition type              */
);

/* SET GLOBAL TRANSFORMATION 3 */
extern void pset_global_tran3(
    Pmatrix3   global_tran   /* global transformation matrix   */
);

/* SET GLOBAL TRANSFORMATION */
extern void pset_global_tran(
    Pmatrix   global_tran   /* global transformation matrix */
);

/* SET MODELLING CLIPPING VOLUME 3 */
extern void pset_model_clip_vol3(
    Pint                       op,           /* operator              */
    const Phalf_space_list3   *half_spaces   /* list of half spaces   */
);

/* SET MODELLING CLIPPING VOLUME */
extern void pset_model_clip_vol(
    Pint                     op,           /* operator              */
    const Phalf_space_list   *half_spaces   /* list of half spaces   */
);

/* SET MODELLING CLIPPING INDICATOR */
extern void pset_model_clip_ind(
    Pclip_ind   clip_ind   /* clipping indicator   */
);

/* RESTORE MODELLING CLIPPING VOLUME */
extern void prestore_model_clip_vol(
          void
);

/* SET VIEW INDEX */
extern void pset_view_ind(
    Pint   view_ind   /* view index   */
);

/* SET VIEW REPRESENTATION 3 */
extern void pset_view_rep3(
    Pint               ws_id,      /* workstation identifier   */
    Pint               view_ind,   /* view index               */
    const Pview_rep3   *view_rep   /* view representation      */
);
```

```
/* SET VIEW REPRESENTATION */
extern void pset_view_rep(
    Pint              ws_id,      /* workstation identifier   */
    Pint              view_ind,   /* view index               */
    const Pview_rep   *view_rep   /* view representation      */
);

/* SET VIEW TRANSFORMATION INPUT PRIORITY */
extern void pset_view_tran_in_pri(
    Pint       ws_id,          /* workstation identifier   */
    Pint       view_ind,       /* view index               */
    Pint       ref_view_ind,   /* reference view index     */
    Prel_pri   rel_pri         /* relative priority        */
);

/* SET WORKSTATION WINDOW 3 */
extern void pset_ws_win3(
    Pint           ws_id,           /* workstation identifier      */
    const Plimit3  *ws_win_limits   /* workstation window limits   */
);

/* SET WORKSTATION WINDOW */
extern void pset_ws_win(
    Pint           ws_id,           /* workstation identifier      */
    const Plimit   *ws_win_limits   /* workstation window limits   */
);

/* SET WORKSTATION VIEWPORT 3 */
extern void pset_ws_vp3(
    Pint           ws_id,           /* workstation identifier       */
    const Plimit3  *ws_vp_limits    /* workstation viewport limits  */
);

/* SET WORKSTATION VIEWPORT */
extern void pset_ws_vp(
    Pint           ws_id,           /* workstation identifier       */
    const Plimit   *ws_vp_limits    /* workstation viewport limits  */
);

/* TRANSLATE 3 */
extern void ptranslate3(
    const Pvec3    *trans_vec,   /* translation vector          */
    Pint           *err_ind,     /* OUT error indicator         */
    Pmatrix3       result_tran   /* OUT transformation matrix   */
);

/* TRANSLATE */
extern void ptranslate(
    const Pvec    *trans_vec,   /* translation vector          */
    Pint          *err_ind,     /* OUT error indicator         */
    Pmatrix       result_tran   /* OUT transformation matrix   */
);
```

```
/* SCALE 3 */
extern void pscale3(
     const Pvec3   *scale_vec,   /* scale factor vector          */
     Pint          *err_ind,     /* OUT error indicator          */
     Pmatrix3      result_tran   /* OUT transformation matrix    */
);

/* SCALE */
extern void pscale(
     const Pvec    *scale_vec,   /* scale factor vector          */
     Pint          *err_ind,     /* OUT error indicator          */
     Pmatrix       result_tran   /* OUT transformation matrix    */
);

/* ROTATE X */
extern void protate_x(
     Pfloat     angle,         /* rotation angle               */
     Pint       *err_ind,      /* OUT error indicator          */
     Pmatrix3   result_tran    /* OUT transformation matrix    */
);

/* ROTATE Y */
extern void protate_y(
     Pfloat     angle,         /* rotation angle               */
     Pint       *err_ind,      /* OUT error indicator          */
     Pmatrix3   result_tran    /* OUT transformation matrix    */
);

/* ROTATE Z */
extern void protate_z(
     Pfloat     angle,         /* rotation angle               */
     Pint       *err_ind,      /* OUT error indicator          */
     Pmatrix3   result_tran    /* OUT transformation matrix    */
);

/* ROTATE */
extern void protate(
     Pfloat     angle,         /* rotation angle               */
     Pint       *err_ind,      /* OUT error indicator          */
     Pmatrix    result_tran    /* OUT transformation matrix    */
);

/* COMPOSE MATRIX 3 */
extern void pcompose_matrix3(
     Pmatrix3   tran_a,        /* transformation matrix a      */
     Pmatrix3   tran_b,        /* transformation matrix b      */
     Pint       *err_ind,      /* OUT error indicator          */
     Pmatrix3   result_tran    /* OUT result matrix            */
);
```

```
/* COMPOSE MATRIX */
extern void pcompose_matrix(
     Pmatrix   tran_a,       /* transformation matrix a   */
     Pmatrix   tran_b,       /* transformation matrix b   */
     Pint      *err_ind,     /* OUT error indicator       */
     Pmatrix   result_tran   /* OUT result matrix         */
);

/* TRANSFORM POINT 3 */
extern void ptran_point3(
     const Ppoint3   *point,     /* point                  */
     Pmatrix3        tran,       /* transformation matrix  */
     Pint            *err_ind,   /* OUT error indicator    */
     Ppoint3         *result     /* OUT transformed point  */
);

/* TRANSFORM POINT */
extern void ptran_point(
     const Ppoint   *point,     /* point                   */
     Pmatrix        tran,       /* transformation matrix   */
     Pint           *err_ind,   /* OUT error indicator     */
     Ppoint         *result     /* OUT transformed point   */
);

/* BUILD TRANSFORMATION MATRIX 3 */
extern void pbuild_tran_matrix3(
     const Ppoint3   *point,       /* fixed point                */
     const Pvec3     *shift_vec,   /* shift vector               */
     Pfloat          x_angle,      /* rotation angle X           */
     Pfloat          y_angle,      /* rotation angle Y           */
     Pfloat          z_angle,      /* rotation angle Z           */
     const Pvec3     *scale_vec,   /* scale vector               */
     Pint            *err_ind,     /* OUT error indicator        */
     Pmatrix3        result_tran   /* OUT transformation matrix  */
);

/* BUILD TRANSFORMATION MATRIX */
extern void pbuild_tran_matrix(
     const Ppoint    *point,       /* fixed point                */
     const Pvec      *shift_vec,   /* shift vector               */
     Pfloat          angle,        /* rotation angle             */
     const Pvec      *scale_vec,   /* scale vector               */
     Pint            *err_ind,     /* OUT error indicator        */
     Pmatrix         result_tran   /* OUT transformation matrix  */
);
```

```
/* COMPOSE TRANSFORMATION MATRIX 3 */
extern void pcompose_tran_matrix3(
     Pmatrix3        tran,         /* transformation matrix       */
     const Ppoint3   *point,       /* fixed point                 */
     const Pvec3     *shift_vec,   /* shift vector                */
     Pfloat          x_angle,      /* rotation angle X            */
     Pfloat          y_angle,      /* rotation angle Y            */
     Pfloat          z_angle,      /* rotation angle Z            */
     const Pvec3     *scale_vec,   /* scale vector                */
     Pint            *err_ind,     /* OUT error indicator         */
     Pmatrix3        result_tran   /* OUT transformation matrix   */
);

/* COMPOSE TRANSFORMATION MATRIX */
extern void pcompose_tran_matrix(
     Pmatrix         tran,         /* transformation matrix       */
     const Ppoint    *point,       /* fixed point                 */
     const Pvec      *shift_vec,   /* shift vector                */
     Pfloat          angle,        /* rotation angle              */
     const Pvec      *scale_vec,   /* scale vector                */
     Pint            *err_ind,     /* OUT error indicator         */
     Pmatrix         result_tran   /* OUT transformation matrix   */
);

/* EVALUATE VIEW ORIENTATION MATRIX 3 */
extern void peval_view_ori_matrix3(
     const Ppoint3   *view_ref_point,   /* view reference point          */
     const Pvec3     *view_norm_vec,    /* view plane normal             */
     const Pvec3     *view_up_vec,      /* view up vector                */
     Pint            *err_ind,          /* OUT error indicator           */
     Pmatrix3        result_tran        /* OUT view orientation matrix   */
);

/* EVALUATE VIEW ORIENTATION MATRIX */
extern void peval_view_ori_matrix(
     const Ppoint    *view_ref_point,   /* view reference point          */
     const Pvec      *view_up_vec,      /* view up vector                */
     Pint            *err_ind,          /* OUT error indicator           */
     Pmatrix         result_tran        /* OUT view orientation matrix   */
);

/* EVALUATE VIEW MAPPING MATRIX 3 */
extern void peval_view_map_matrix3(
     const Pview_map3   *mapping,     /* view mapping                */
     Pint               *err_ind,     /* OUT error indicator         */
     Pmatrix3           result_tran   /* OUT view mapping matrix     */
);
```

```
/* EVALUATE VIEW MAPPING MATRIX */
extern void peval_view_map_matrix(
    const Pview_map   *mapping,     /* view mapping              */
    Pint              *err_ind,     /* OUT error indicator       */
    Pmatrix           result_tran   /* OUT view mapping matrix   */
);

/* OPEN STRUCTURE */
extern void popen_struct(
    Pint   struct_id   /* structure identifier   */
);

/* CLOSE STRUCTURE */
extern void pclose_struct(
      void
);

/* EXECUTE STRUCTURE */
extern void pexec_struct(
    Pint   struct_id   /* structure identifier   */
);

/* LABEL */
extern void plabel(
    Pint   label_id   /* label identifier   */
);

/* APPLICATION DATA */
extern void pappl_data(
    const Pdata   *data   /* application data   */
);

/* GENERALIZED STRUCTURE ELEMENT */
extern void pgse(
    Pint                id,         /* gse identifier    */
    const Pgse_data     *gse_data   /* gse data record   */
);

/* SET EDIT MODE */
extern void pset_edit_mode(
    Pedit_mode   edit_mode   /* edit mode   */
);

/* COPY ALL ELEMENTS FROM STRUCTURE */
extern void pcopy_all_elems_struct(
    Pint   struct_id   /* structure identifier   */
);

/* SET ELEMENT POINTER */
extern void pset_elem_ptr(
    Pint   elem_ptr_value   /* element pointer value   */
);
```

```
/* OFFSET ELEMENT POINTER */
extern void poffset_elem_ptr(
    Pint   elem_ptr_offset   /* element pointer offset   */
);

/* SET ELEMENT POINTER AT LABEL */
extern void pset_elem_ptr_label(
    Pint   label_id   /* label identifier   */
);

/* DELETE ELEMENT */
extern void pdel_elem(
     void
);

/* DELETE ELEMENT RANGE */
extern void pdel_elem_range(
    Pint   elem_ptr1_value,   /* element pointer 1 value   */
    Pint   elem_ptr2_value    /* element pointer 2 value   */
);

/* DELETE ELEMENTS BETWEEN LABELS */
extern void pdel_elems_labels(
    Pint   label1_id,   /* label 1 identifier   */
    Pint   label2_id    /* label 2 identifier   */
);

/* EMPTY STRUCTURE */
extern void pempty_struct(
    Pint   struct_id   /* structure id   */
);

/* DELETE STRUCTURE */
extern void pdel_struct(
    Pint   struct_id   /* structure identifier   */
);

/* DELETE STRUCTURE NETWORK */
extern void pdel_struct_net(
    Pint        struct_id,   /* structure identifier      */
    Pref_flag   ref_flag     /* reference handling flag   */
);

/* DELETE ALL STRUCTURES */
extern void pdel_all_structs(
       void
);

/* CHANGE STRUCTURE IDENTIFIER */
extern void pchange_struct_id(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);
```

```
/* CHANGE STRUCTURE REFERENCES */
extern void pchange_struct_refs(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);

/* CHANGE STRUCTURE IDENTIFIER AND REFERENCES */
extern void pchange_struct_id_refs(
    Pint   orig_struct_id,    /* original structure id   */
    Pint   result_struct_id   /* result structure id     */
);

/* POST STRUCTURE */
extern void ppost_struct(
    Pint     ws_id,       /* workstation identifier   */
    Pint     struct_id,   /* structure identifier     */
    Pfloat   pri          /* priority                 */
);

/* UNPOST STRUCTURE */
extern void punpost_struct(
    Pint   ws_id,      /* workstation identifier   */
    Pint   struct_id   /* structure identifier     */
);

/* UNPOST ALL STRUCTURES */
extern void punpost_all_structs(
    Pint   ws_id   /* workstation identifier   */
);

/* OPEN ARCHIVE FILE */
extern void popen_ar_file(
    Pint          archive_id,     /* archive identifier   */
    const char    *archive_file   /* archive file name    */
);

/* CLOSE ARCHIVE FILE */
extern void pclose_ar_file(
    Pint   archive_id   /* archive identifier   */
);

/* ARCHIVE STRUCTURES */
extern void par_structs(
    Pint              archive_id,   /* archive identifier             */
    const Pint_list   *struct_ids   /* list of structure identifiers  */
);

/* ARCHIVE STRUCTURE NETWORKS */
extern void par_struct_nets(
    Pint              archive_id,   /* archive identifier             */
    const Pint_list   *struct_ids   /* list of structure identifiers  */
);
```

```
/* ARCHIVE ALL STRUCTURES */
extern void par_all_structs(
    Pint   archive_id   /* archive identifier   */
);

/* SET CONFLICT RESOLUTION */
extern void pset_conf_res(
    Pconf_res   archival_res,   /* archival conflict resolution    */
    Pconf_res   retrieval_res   /* retrieval conflict resolution   */
);

/* RETRIEVE STRUCTURE IDENTIFIERS */
extern void pret_struct_ids(
    Pint        archive_id,             /* archive identifier          */
    Pint        num_elems_appl_list,    /* # elems in appl list        */
    Pint        start_ind,              /* starting index              */
    Pint_list   *ids,                   /* OUT list of structure ids   */
    Pint        *num_elems_impl_list    /* OUT # elems in impl. list   */
);

/* RETRIEVE STRUCTURES */
extern void pret_structs(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);

/* RETRIEVE STRUCTURE NETWORKS */
extern void pret_struct_nets(
    Pint              archive_id,   /* archive identifier              */
    const Pint_list   *struct_ids   /* list of structure identifiers   */
);

/* RETRIEVE ALL STRUCTURES */
extern void pret_all_structs(
    Pint   archive_id   /* archive identifier   */
);

/* RETRIEVE PATHS TO ANCESTORS */
extern void pret_paths_ances(
    Pint                ar_id,       /* archive identifier        */
    Pint                struct_id,   /* structure identifier      */
    Ppath_order         order,       /* path order                */
    Pint                depth,       /* path depth                */
    Pstore              store,       /* handle to Store object    */
    Pelem_ref_list_list **paths      /* OUT structure path list   */
);
```

```
/* RETRIEVE PATHS TO DESCENDANTS */
extern void pret_paths_descs(
    Pint                ar_id,       /* archive identifier        */
    Pint                struct_id,   /* structure identifier      */
    Ppath_order         order,       /* path order                */
    Pint                depth,       /* path depth                */
    Pstore              store,       /* handle to Store object    */
    Pelem_ref_list_list **paths      /* OUT structure path list   */
);

/* DELETE STRUCTURES FROM ARCHIVE */
extern void pdel_structs_ar(
    Pint             archive_id,   /* archive identifier              */
    const Pint_list  *struct_ids   /* list of structure identifiers   */
);

/* DELETE STRUCTURE NETWORKS FROM ARCHIVE */
extern void pdel_struct_nets_ar(
    Pint             archive_id,   /* archive identifier              */
    const Pint_list  *struct_ids   /* list of structure identifiers   */
);

/* DELETE ALL STRUCTURES FROM ARCHIVE */
extern void pdel_all_structs_ar(
    Pint  archive_id   /* archive identifier   */
);

/* SET PICK IDENTIFIER */
extern void pset_pick_id(
    Pint  pick_id   /* pick identifier   */
);

/* SET PICK FILTER */
extern void pset_pick_filter(
    Pint            ws_id,      /* workstation identifier   */
    Pint            pick_num,   /* pick device number       */
    const Pfilter   *filter     /* pick filter              */
);

/* INITIALIZE LOCATOR 3 */
extern void pinit_loc3(
    Pint              ws_id,          /* workstation identifier     */
    Pint              loc_num,        /* locator device number      */
    Pint              init_view_ind,  /* initial view index         */
    const Ppoint3     *init_loc_pos,  /* initial locator position   */
    Pint              pet,            /* prompt and echo type       */
    const Plimit3     *echo_vol,      /* echo volume                */
    const Ploc_data3  *loc_data       /* data record                */
);
```

```
/* INITIALIZE LOCATOR */
extern void pinit_loc(
    Pint              ws_id,          /* workstation identifier     */
    Pint              loc_num,        /* locator device number      */
    Pint              init_view_ind,  /* initial view index         */
    const Ppoint      *init_loc_pos,  /* initial locator position   */
    Pint              pet,            /* prompt and echo type       */
    const Plimit      *echo_area,     /* echo area                  */
    const Ploc_data   *loc_data       /* data record                */
);

/* INITIALIZE STROKE 3 */
extern void pinit_stroke3(
    Pint                  ws_id,          /* workstation identifier   */
    Pint                  stroke_num,     /* stroke device number     */
    Pint                  init_view_ind,  /* initial view index       */
    const Ppoint_list3    *init_stroke,   /* initial stroke           */
    Pint                  pet,            /* prompt and echo type     */
    const Plimit3         *echo_vol,      /* echo volume              */
    const Pstroke_data3   *stroke_data    /* data record              */
);

/* INITIALIZE STROKE */
extern void pinit_stroke(
    Pint                ws_id,          /* workstation identifier   */
    Pint                stroke_num,     /* stroke device number     */
    Pint                init_view_ind,  /* initial view index       */
    const Ppoint_list   *init_stroke,   /* initial stroke           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Pstroke_data  *stroke_data    /* data record              */
);

/* INITIALIZE VALUATOR 3 */
extern void pinit_val3(
    Pint                ws_id,        /* workstation identifier   */
    Pint                val_num,      /* valuator device number   */
    Pfloat              init_value,   /* initial value            */
    Pint                pet,          /* prompt and echo type     */
    const Plimit3       *echo_vol,    /* echo volume              */
    const Pval_data3    *val_data     /* data record              */
);

/* INITIALIZE VALUATOR */
extern void pinit_val(
    Pint              ws_id,        /* workstation identifier   */
    Pint              val_num,      /* valuator device number   */
    Pfloat            init_value,   /* initial value            */
    Pint              pet,          /* prompt and echo type     */
    const Plimit      *echo_area,   /* echo area                */
    const Pval_data   *val_data     /* data record              */
);
```

```
/* INITIALIZE CHOICE 3 */
extern void pinit_choice3(
    Pint                ws_id,          /* workstation identifier   */
    Pint                choice_num,     /* choice device number     */
    Pin_status          init_status,    /* initial choice status    */
    Pint                init_choice,    /* initial choice           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit3       *echo_vol,      /* echo volume              */
    const Pchoice_data3 *choice_data    /* data record              */
);

/* INITIALIZE CHOICE */
extern void pinit_choice(
    Pint                ws_id,          /* workstation identifier   */
    Pint                choice_num,     /* choice device number     */
    Pin_status          init_status,    /* initial choice status    */
    Pint                init_choice,    /* initial choice           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Pchoice_data  *choice_data    /* data record              */
);

/* INITIALIZE PICK 3 */
extern void pinit_pick3(
    Pint                ws_id,          /* workstation identifier   */
    Pint                pick_num,       /* pick device number       */
    Pin_status          init_status,    /* initial pick status      */
    const Ppick_path    *init_pick,     /* initial pick path        */
    Pint                pet,            /* prompt and echo type     */
    const Plimit3       *echo_vol,      /* echo volume              */
    const Ppick_data3   *pick_data,     /* data record              */
    Ppath_order         order           /* pick path order          */
);

/* INITIALIZE PICK */
extern void pinit_pick(
    Pint                ws_id,          /* workstation identifier   */
    Pint                pick_num,       /* pick device number       */
    Pin_status          init_status,    /* initial pick status      */
    const Ppick_path    *init_pick,     /* initial pick path        */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Ppick_data    *pick_data,     /* data record              */
    Ppath_order         order           /* pick path order          */
);
```

```
/* INITIALIZE STRING 3 */
extern void pinit_string3(
    Pint                ws_id,          /* workstation identifier   */
    Pint                string_num,     /* string device number     */
    const char          *init_string,   /* initial string           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit3       *echo_vol,      /* echo volume              */
    const Pstring_data3 *string_data    /* data record              */
);

/* INITIALIZE STRING */
extern void pinit_string(
    Pint                ws_id,          /* workstation identifier   */
    Pint                string_num,     /* string device number     */
    const char          *init_string,   /* initial string           */
    Pint                pet,            /* prompt and echo type     */
    const Plimit        *echo_area,     /* echo area                */
    const Pstring_data  *string_data    /* data record              */
);

/* SET LOCATOR MODE */
extern void pset_loc_mode(
    Pint          ws_id,        /* workstation identifier   */
    Pint          loc_num,      /* locator device number    */
    Pop_mode      op_mode,      /* operating mode           */
    Pecho_switch  echo_switch   /* echo switch              */
);

/* SET STROKE MODE */
extern void pset_stroke_mode(
    Pint          ws_id,        /* workstation identifier   */
    Pint          stroke_num,   /* stroke device number     */
    Pop_mode      op_mode,      /* operating mode           */
    Pecho_switch  echo_switch   /* echo switch              */
);

/* SET VALUATOR MODE */
extern void pset_val_mode(
    Pint          ws_id,        /* workstation identifier   */
    Pint          val_num,      /* valuator device number   */
    Pop_mode      op_mode,      /* operating mode           */
    Pecho_switch  echo_switch   /* echo switch              */
);

/* SET CHOICE MODE */
extern void pset_choice_mode(
    Pint          ws_id,        /* workstation identifier   */
    Pint          choice_num,   /* choice device number     */
    Pop_mode      op_mode,      /* operating mode           */
    Pecho_switch  echo_switch   /* echo switch              */
);
```

```
/* SET PICK MODE */
extern void pset_pick_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           pick_num,     /* pick device number       */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);

/* SET STRING MODE */
extern void pset_string_mode(
     Pint           ws_id,        /* workstation identifier   */
     Pint           string_num,   /* string device number     */
     Pop_mode       op_mode,      /* operating mode           */
     Pecho_switch   echo_switch   /* echo switch              */
);

/* REQUEST LOCATOR 3 */
extern void preq_loc3(
     Pint         ws_id,         /* workstation identifier   */
     Pint         loc_num,       /* locator device number    */
     Pin_status   *in_status,    /* OUT input status         */
     Pint         *view_ind,     /* OUT view index           */
     Ppoint3      *loc_pos       /* OUT locator position     */
);

/* REQUEST LOCATOR */
extern void preq_loc(
     Pint         ws_id,         /* workstation identifier   */
     Pint         loc_num,       /* locator device number    */
     Pin_status   *in_status,    /* OUT input status         */
     Pint         *view_ind,     /* OUT view index           */
     Ppoint       *loc_pos       /* OUT locator position     */
);

/* REQUEST STROKE 3 */
extern void preq_stroke3(
     Pint           ws_id,        /* workstation identifier   */
     Pint           stroke_num,   /* stroke device number     */
     Pin_status     *in_status,   /* OUT input status         */
     Pint           *view_ind,    /* OUT view index           */
     Ppoint_list3   *stroke       /* OUT stroke               */
);

/* REQUEST STROKE */
extern void preq_stroke(
     Pint           ws_id,        /* workstation identifier   */
     Pint           stroke_num,   /* stroke device number     */
     Pin_status     *in_status,   /* OUT input status         */
     Pint           *view_ind,    /* OUT view index           */
     Ppoint_list    *stroke       /* OUT stroke               */
);
```

```
/* REQUEST VALUATOR */
extern void preq_val(
     Pint         ws_id,        /* workstation identifier   */
     Pint         val_num,      /* valuator device number   */
     Pin_status   *in_status,   /* OUT input status         */
     Pfloat       *value        /* OUT valuator value       */
);

/* REQUEST CHOICE */
extern void preq_choice(
     Pint         ws_id,        /* workstation identifier   */
     Pint         choice_num,   /* choice device number     */
     Pin_status   *in_status,   /* OUT choice status        */
     Pint         *choice       /* OUT requested choice     */
);

/* REQUEST PICK */
extern void preq_pick(
     Pint         ws_id,        /* workstation identifier        */
     Pint         pick_num,     /* pick device number            */
     Pint         depth,        /* max. depth of returned path   */
     Pin_status   *in_status,   /* OUT pick status               */
     Ppick_path   *pick         /* OUT requested pick path       */
);

/* REQUEST STRING */
extern void preq_string(
     Pint         ws_id,        /* workstation identifier   */
     Pint         string_num,   /* string device number     */
     Pin_status   *in_status,   /* OUT input status         */
     char         *string       /* OUT requested string     */
);

/* SAMPLE LOCATOR 3 */
extern void psample_loc3(
     Pint      ws_id,       /* workstation identifier   */
     Pint      loc_num,     /* locator device number    */
     Pint      *view_ind,   /* OUT view index           */
     Ppoint3   *loc_pos     /* OUT locator position     */
);

/* SAMPLE LOCATOR */
extern void psample_loc(
     Pint      ws_id,       /* workstation identifier   */
     Pint      loc_num,     /* locator device number    */
     Pint      *view_ind,   /* OUT view index           */
     Ppoint    *loc_pos     /* OUT locator position     */
);
```

```
/* SAMPLE STROKE 3 */
extern void psample_stroke3(
    Pint           ws_id,        /* workstation identifier   */
    Pint           stroke_num,   /* stroke device number     */
    Pint           *view_ind,    /* OUT view index           */
    Ppoint_list3   *stroke       /* OUT stroke               */
);

/* SAMPLE STROKE */
extern void psample_stroke(
    Pint           ws_id,        /* workstation identifier   */
    Pint           stroke_num,   /* stroke device number     */
    Pint           *view_ind,    /* OUT view index           */
    Ppoint_list    *stroke       /* OUT stroke               */
);

/* SAMPLE VALUATOR */
extern void psample_val(
    Pint     ws_id,     /* workstation identifier   */
    Pint     val_num,   /* valuator device number   */
    Pfloat   *value     /* OUT valuator value       */
);

/* SAMPLE CHOICE */
extern void psample_choice(
    Pint         ws_id,              /* workstation identifier    */
    Pint         choice_num,         /* choice device number      */
    Pin_status   *choice_in_status,  /* OUT choice input status   */
    Pint         *choice             /* OUT choice                */
);

/* SAMPLE PICK */
extern void psample_pick(
    Pint         ws_id,              /* workstation identifier        */
    Pint         pick_num,           /* pick device number            */
    Pint         depth,              /* max. depth of returned path   */
    Pin_status   *pick_in_status,    /* OUT pick input status         */
    Ppick_path   *pick               /* OUT pick path                 */
);

/* SAMPLE STRING */
extern void psample_string(
    Pint    ws_id,        /* workstation identifier   */
    Pint    string_num,   /* string device number     */
    char    *string       /* OUT string               */
);
```

```
/* AWAIT EVENT */
extern void pawait_event(
    Pfloat      timeout,      /* timeout (seconds)                 */
    Pint        *ws_id,       /* OUT workstation identifier        */
    Pin_class   *dev_class,   /* OUT device class                  */
    Pint        *in_num       /* OUT logical input device number   */
);

/* FLUSH DEVICE EVENTS */
extern void pflush_events(
    Pint        ws_id,        /* workstation identifier            */
    Pin_class   dev_class,    /* device class                      */
    Pint        in_num        /* logical input device number       */
);

/* GET LOCATOR 3 */
extern void pget_loc3(
    Pint      *view_ind,    /* OUT view index          */
    Ppoint3   *loc_pos      /* OUT locator position    */
);

/* GET LOCATOR */
extern void pget_loc(
    Pint      *view_ind,    /* OUT view index          */
    Ppoint    *loc_pos      /* OUT locator position    */
);

/* GET STROKE 3 */
extern void pget_stroke3(
    Pint           *view_ind,   /* OUT view index   */
    Ppoint_list3   *stroke      /* OUT stroke       */
);

/* GET STROKE */
extern void pget_stroke(
    Pint           *view_ind,   /* OUT view index   */
    Ppoint_list    *stroke      /* OUT stroke       */
);

/* GET VALUATOR */
extern void pget_val(
    Pfloat   *value   /* OUT valuator value   */
);

/* GET CHOICE */
extern void pget_choice(
    Pin_status   *in_status,   /* OUT choice status   */
    Pint         *choice       /* OUT choice          */
);
```

```
/* GET PICK */
extern void pget_pick(
    Pint         depth,        /* max. depth of returned path   */
    Pin_status   *in_status,   /* OUT pick status               */
    Ppick_path   *pick         /* OUT pick path                 */
);

/* GET STRING */
extern void pget_string(
    char    *string   /* OUT string   */
);

/* WRITE ITEM TO METAFILE */
extern void pwrite_item(
    Pint               ws_id,              /* workstation identifier    */
    Pint               item_type,          /* item type                 */
    Pint               item_data_length,   /* item data record length   */
    const Pitem_data   *item_data          /* item data record          */
);

/* GET ITEM TYPE FROM METAFILE */
extern void pget_item_type(
    Pint   ws_id,              /* workstation identifier        */
    Pint   *item_type,         /* OUT item type                 */
    Pint   *item_data_length   /* OUT item data record length   */
);

/* READ ITEM FROM METAFILE */
extern void pread_item(
    Pint         ws_id,                /* workstation identifier        */
    Pint         max_item_data_length, /* max item data record length   */
    Pitem_data   *item_data            /* OUT item data record          */
);

/* INTERPRET ITEM */
extern void pinterpret_item(
    Pint               type,               /* item type                 */
    Pint               item_data_length,   /* item data record length   */
    const Pitem_data   *item_data          /* item data record          */
);

/* INQUIRE SYSTEM STATE VALUE */
extern void pinq_sys_st(
    Psys_st   *sys_st   /* OUT the system state   */
);

/* INQUIRE WORKSTATION STATE VALUE */
extern void pinq_ws_st(
    Pws_st   *ws_st   /* OUT workstation state   */
);
```

```
/* INQUIRE STRUCTURE STATE VALUE */
extern void pinq_struct_st(
    Pstruct_st   *struct_st   /* OUT structure state   */
);

/* INQUIRE ARCHIVE STATE VALUE */
extern void pinq_ar_st(
    Par_st   *ar_st   /* OUT archive state   */
);

/* INQUIRE LIST OF AVAILABLE WORKSTATION TYPES */
extern void pinq_list_avail_ws_types(
    Pint        num_elems_appl_list,   /* # elems in appl list       */
    Pint        start_ind,             /* starting index             */
    Pint        *err_ind,              /* OUT error indicator        */
    Pint_list   *types,                /* OUT list of ws types       */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list   */
);

/* INQUIRE PHIGS FACILITIES */
extern void pinq_phigs_facs(
    Pint        num_elems_appl_list,    /* # elems in appl list                    */
    Pint        start_ind,              /* starting index                          */
    Pint        *err_ind,               /* OUT error indicator                     */
    Pint        *max_open_ws,           /* OUT max. # simult. open ws              */
    Pint        *max_open_ar,           /* OUT max. # simult. open archive files   */
    Pint        *num_avail_names,       /* OUT # available names for name sets     */
    Pint_list   *char_sets,             /* OUT list of character sets              */
    Pint        *num_elems_impl_list,   /* OUT # elems in impl list                */
    Pint        *iss_norm_max,          /* OUT ISS max. length of normal filter list   */
    Pint        *iss_inv_max            /* OUT ISS max. length of inverted filter list */
);

/* INQUIRE GENERALIZED STRUCTURE ELEMENT FACILITIES */
extern void pinq_gse_facs(
    Pint                num_elems_appl_list,   /* # elems in appl list                  */
    Pint                start_ind,             /* starting index                        */
    Pint                *err_ind,              /* OUT error indicator                   */
    Pgse_id_dep_list    *gse,                  /* OUT list of GSE ids and dependencies  */
    Pint                *num_elems_impl_list   /* OUT # elems in impl list              */
);

/* INQUIRE MODELLING CLIPPING FACILITIES */
extern void pinq_model_clip_facs(
    Pint        num_elems_appl_list,   /* # elems in appl list          */
    Pint        start_ind,             /* starting index                */
    Pint        *err_ind,              /* OUT error indicator           */
    Pint        *num_planes,           /* OUT number of distinct planes */
    Pint_list   *ops,                  /* OUT list of operators         */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list      */
);
```

```
/* INQUIRE EDIT MODE */
extern void pinq_edit_mode(
    Pint         *err_ind,    /* OUT error indicator   */
    Pedit_mode   *edit_mode   /* OUT edit mode         */
);

/* INQUIRE SET OF OPEN WORKSTATIONS */
extern void pinq_open_wss(
    Pint        num_elems_appl_list,   /* # elems in appl list       */
    Pint        start_ind,             /* starting index             */
    Pint        *err_ind,              /* OUT error indicator        */
    Pint_list   *open_ws_ids,          /* OUT list of ws ids         */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list   */
);

/* INQUIRE STRUCTURE IDENTIFIERS */
extern void pinq_struct_ids(
    Pint        num_elems_appl_list,   /* # elems in appl list       */
    Pint        start_ind,             /* starting index             */
    Pint        *err_ind,              /* OUT error indicator        */
    Pint_list   *struct_ids,           /* OUT list of structure ids  */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list   */
);

/* INQUIRE ARCHIVE FILES */
extern void pinq_ar_files(
    Pstore          store,       /* handle to Store object                  */
    Pint            *err_ind,    /* OUT error indicator                     */
    Par_file_list   **ar_files   /* OUT list of archive file names and ids  */
);

/* INQUIRE CONFLICT RESOLUTION */
extern void pinq_conf_res(
    Pint        *err_ind,     /* OUT error indicator       */
    Pconf_res   *archive_res, /* OUT archival resolution   */
    Pconf_res   *retrieve_res /* OUT retrieval resolution  */
);

/* INQUIRE ALL CONFLICTING STRUCTURES */
extern void pinq_all_conf_structs(
    Pint        ar_id,                 /* archive identifier                  */
    Pint        num_elems_appl_list,   /* # elems in appl list                */
    Pint        start_ind,             /* starting index                      */
    Pint        *err_ind,              /* OUT error indicator                 */
    Pint_list   *ids,                  /* OUT list of conflicting structure ids */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list            */
);
```

```
/* INQUIRE CONFLICTING STRUCTURES IN NETWORK */
extern void pinq_conf_structs_net(
    Pint                ar_id,                /* archive identifier              */
    Pint                struct_id,            /* structure identifier            */
    Pstruct_net_source  source,               /* structure network source        */
    Pint                num_elems_appl_list,  /* # elems in appl list            */
    Pint                start_ind,            /* starting index                  */
    Pint                *err_ind,             /* OUT error indicator             */
    Pint_list           *ids,                 /* OUT conflicting struct id list  */
    Pint                *num_elems_impl_list  /* OUT # elems in impl list        */
);

/* INQUIRE MORE SIMULTANEOUS EVENTS */
extern void pinq_more_simult_events(
    Pint                  *err_ind,        /* OUT error indicator       */
    Pmore_simult_events   *simult_events   /* OUT simultaneous events   */
);

/* INQUIRE WORKSTATION CONNECTION AND TYPE */
extern void pinq_ws_conn_type(
    Pint     ws_id,       /* workstation identifier     */
    Pstore   store,       /* handle to Store object     */
    Pint     *err_ind,    /* OUT error indicator        */
    void     **conn_id,   /* OUT connection identifier  */
    Pint     *ws_type     /* OUT workstation type       */
);

/* INQUIRE LIST OF VIEW INDICES */
extern void pinq_list_view_inds(
    Pint        ws_id,                /* workstation identifier     */
    Pint        num_elems_appl_list,  /* # elems in appl list       */
    Pint        start_ind,            /* starting index             */
    Pint        *err_ind,             /* OUT error indicator        */
    Pint_list   *view_inds,           /* OUT list of view indices   */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list   */
);

/* INQUIRE VIEW REPRESENTATION */
extern void pinq_view_rep(
    Pint        ws_id,       /* workstation identifier             */
    Pint        view_ind,    /* view index                         */
    Pint        *err_ind,    /* OUT error indicator                */
    Pupd_st     *upd_st,     /* OUT transformation update state    */
    Pview_rep3  *cur_view,   /* OUT current view representation    */
    Pview_rep3  *req_view    /* OUT requested view representation  */
);
```

```
/* INQUIRE HLHSR MODE */
extern void pinq_hlhsr_mode(
    Pint      ws_id,       /* workstation identifier     */
    Pint      *err_ind,    /* OUT error indicator        */
    Pupd_st   *upd_st,     /* OUT HLHSR update state     */
    Pint      *cur_mode,   /* OUT current HLHSR mode     */
    Pint      *req_mode    /* OUT requested HLHSR mode   */
);

/* INQUIRE POSTED STRUCTURES */
extern void pinq_posted_structs(
    Pint                  ws_id,                /* workstation identifier          */
    Pint                  num_elems_appl_list,  /* # elems in appl list            */
    Pint                  start_ind,            /* starting index                  */
    Pint                  *err_ind,             /* OUT error indicator             */
    Pposted_struct_list   *struct_ids,          /* OUT list of posted structures   */
    Pint                  *num_elems_impl_list  /* OUT # elems in impl list        */
);

/* INQUIRE DISPLAY UPDATE STATE */
extern void pinq_disp_upd_st(
    Pint              ws_id,              /* workstation identifier              */
    Pint              *err_ind,           /* OUT error indicator                 */
    Pdefer_mode       *def_mode,          /* OUT deferral mode                   */
    Pmod_mode         *mod_mode,          /* OUT modification mode               */
    Pdisp_surf_empty  *disp_surf_empty,   /* OUT display surface empty           */
    Pvisual_st        *vis_st             /* OUT state of visual representation  */
);

/* INQUIRE LIST OF POLYLINE INDICES */
extern void pinq_list_line_inds(
    Pint        ws_id,                /* workstation identifier                */
    Pint        num_elems_appl_list,  /* # elems in appl list                  */
    Pint        start_ind,            /* starting index                        */
    Pint        *err_ind,             /* OUT error indicator                   */
    Pint_list   *def_line_ind,        /* OUT list of defined polyline indices  */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list              */
);

/* INQUIRE POLYLINE REPRESENTATION */
extern void pinq_line_rep(
    Pint           ws_id,      /* workstation identifier       */
    Pint           index,      /* polyline index               */
    Pinq_type      type,       /* type of returned value       */
    Pint           *err_ind,   /* OUT error indicator          */
    Pline_bundle   *line_rep   /* OUT polyline representation  */
);
```

```
/* INQUIRE LIST OF POLYMARKER INDICES */
extern void pinq_list_marker_inds(
    Pint        ws_id,                  /* workstation identifier                  */
    Pint        num_elems_appl_list,    /* # elems in appl list                    */
    Pint        start_ind,              /* starting index                          */
    Pint        *err_ind,               /* OUT error indicator                     */
    Pint_list   *def_marker_ind,        /* OUT list of defined polymarker indices  */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list                */
);

/* INQUIRE POLYMARKER REPRESENTATION */
extern void pinq_marker_rep(
    Pint             ws_id,         /* workstation identifier          */
    Pint             index,         /* polymarker index                */
    Pinq_type        type,          /* type of returned value          */
    Pint             *err_ind,      /* OUT error indicator             */
    Pmarker_bundle   *marker_rep    /* OUT polymarker representation   */
);

/* INQUIRE LIST OF TEXT INDICES */
extern void pinq_list_text_inds(
    Pint        ws_id,                  /* workstation identifier          */
    Pint        num_elems_appl_list,    /* # elems in appl list            */
    Pint        start_ind,              /* starting index                  */
    Pint        *err_ind,               /* OUT error indicator             */
    Pint_list   *def_text_ind,          /* OUT list of defined text indices */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list        */
);

/* INQUIRE TEXT REPRESENTATION */
extern void pinq_text_rep(
    Pint           ws_id,       /* workstation identifier    */
    Pint           index,       /* text index                */
    Pinq_type      type,        /* type of returned value    */
    Pint           *err_ind,    /* OUT error indicator       */
    Ptext_bundle   *text_rep    /* OUT text representation   */
);

/* INQUIRE LIST OF INTERIOR INDICES */
extern void pinq_list_int_inds(
    Pint        ws_id,                  /* workstation identifier                  */
    Pint        num_elems_appl_list,    /* # elems in appl list                    */
    Pint        start_ind,              /* starting index                          */
    Pint        *err_ind,               /* OUT error indicator                     */
    Pint_list   *def_int_ind,           /* OUT list of defined interior indices    */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list                */
);
```

```
/* INQUIRE INTERIOR REPRESENTATION */
extern void pinq_int_rep(
    Pint          ws_id,      /* workstation identifier        */
    Pint          index,      /* interior index                */
    Pinq_type     type,       /* type of returned value        */
    Pint          *err_ind,   /* OUT error indicator           */
    Pint_bundle   *int_rep    /* OUT interior representation   */
);

/* INQUIRE LIST OF EDGE INDICES */
extern void pinq_list_edge_inds(
    Pint        ws_id,                /* workstation identifier              */
    Pint        num_elems_appl_list,  /* # elems in appl list                */
    Pint        start_ind,            /* starting index                      */
    Pint        *err_ind,             /* OUT error indicator                 */
    Pint_list   *def_edge_ind,        /* OUT list of defined edge indices    */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list            */
);

/* INQUIRE EDGE REPRESENTATION */
extern void pinq_edge_rep(
    Pint          ws_id,      /* workstation identifier    */
    Pint          index,      /* edge index                */
    Pinq_type     type,       /* type of returned value    */
    Pint          *err_ind,   /* OUT error indicator       */
    Pedge_bundle  *edge_rep   /* OUT edge representation   */
);

/* INQUIRE LIST OF PATTERN INDICES */
extern void pinq_list_pat_inds(
    Pint        ws_id,                /* workstation identifier              */
    Pint        num_elems_appl_list,  /* # elems in appl list                */
    Pint        start_ind,            /* starting index                      */
    Pint        *err_ind,             /* OUT error indicator                 */
    Pint_list   *def_pat_ind,         /* OUT list of defined pattern indices */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list            */
);

/* INQUIRE PATTERN REPRESENTATION */
extern void pinq_pat_rep(
    Pint          ws_id,      /* workstation identifier      */
    Pint          index,      /* pattern index               */
    Pinq_type     type,       /* type of returned value      */
    Pstore        store,      /* handle to Store object      */
    Pint          *err_ind,   /* OUT error indicator         */
    Ppat_rep      **pat_rep   /* OUT pattern representation  */
);
```

```
/* INQUIRE COLOUR MODEL */
extern void pinq_colr_model(
    Pint   ws_id,      /* workstation identifier     */
    Pint   *err_ind,   /* OUT error indicator        */
    Pint   *model      /* OUT current colour model   */
);

/* INQUIRE LIST OF COLOUR INDICES */
extern void pinq_list_colr_inds(
    Pint        ws_id,                /* workstation identifier      */
    Pint        num_elems_appl_list,  /* # elems in appl list        */
    Pint        start_ind,            /* starting index              */
    Pint        *err_ind,             /* OUT error indicator         */
    Pint_list   *colr_ind,            /* OUT list of colour indices  */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list    */
);

/* INQUIRE COLOUR REPRESENTATION */
extern void pinq_colr_rep(
    Pint        ws_id,      /* workstation identifier      */
    Pint        colr_ind,   /* colour index                */
    Pinq_type   type,       /* type of returned value      */
    Pint        *err_ind,   /* OUT error indicator         */
    Pcolr_rep   *colr_rep   /* OUT colour representation   */
);

/* INQUIRE HIGHLIGHTING FILTER */
extern void pinq_highl_filter(
    Pint      ws_id,          /* workstation identifier    */
    Pstore    store,          /* handle to Store object    */
    Pint      *err_ind,       /* OUT error indicator       */
    Pfilter   **highl_filter  /* OUT highlighting filter   */
);

/* INQUIRE INVISIBILITY FILTER */
extern void pinq_invis_filter(
    Pint      ws_id,          /* workstation identifier    */
    Pstore    store,          /* handle to Store object    */
    Pint      *err_ind,       /* OUT error indicator       */
    Pfilter   **invis_filter  /* OUT invisibility filter   */
);

/* INQUIRE WORKSTATION TRANSFORMATION 3 */
extern void pinq_ws_tran3(
    Pint      ws_id,          /* workstation identifier              */
    Pint      *err_ind,       /* OUT error indicator                 */
    Pupd_st   *upd_st,        /* OUT update state                    */
    Plimit3   *req_win_lim,   /* OUT requested workstation window    */
    Plimit3   *cur_win_lim,   /* OUT current workstation window      */
    Plimit3   *req_vp_lim,    /* OUT requested workstation viewport  */
    Plimit3   *cur_vp_lim     /* OUT current workstation viewport    */
);
```

```
/* INQUIRE WORKSTATION TRANSFORMATION */
extern void pinq_ws_tran(
    Pint      ws_id,          /* workstation identifier              */
    Pint      *err_ind,       /* OUT error indicator                 */
    Pupd_st   *upd_st,        /* OUT update state                    */
    Plimit    *req_win_lim,   /* OUT requested workstation window    */
    Plimit    *cur_win_lim,   /* OUT current workstation window      */
    Plimit    *req_vp_lim,    /* OUT requested workstation viewport  */
    Plimit    *cur_vp_lim     /* OUT current workstation viewport    */
);

/* INQUIRE LOCATOR DEVICE STATE 3 */
extern void pinq_loc_st3(
    Pint           ws_id,           /* workstation identifier        */
    Pint           loc_num,         /* locator device number         */
    Pinq_type      type,            /* type of returned value        */
    Pstore         store,           /* handle to Store object        */
    Pint           *err_ind,        /* OUT error indicator           */
    Pop_mode       *op_mode,        /* OUT operating mode            */
    Pecho_switch   *echo_switch,    /* OUT echo switch               */
    Pint           *init_view_ind,  /* OUT initial view index        */
    Ppoint3        *init_loc_pos,   /* OUT initial locator position  */
    Pint           *prompt_echo,    /* OUT prompt/echo type          */
    Plimit3        *echo_vol,       /* OUT echo volume               */
    Ploc_data3     **loc_data       /* OUT data record               */
);

/* INQUIRE LOCATOR DEVICE STATE */
extern void pinq_loc_st(
    Pint           ws_id,           /* workstation identifier        */
    Pint           loc_num,         /* locator device number         */
    Pinq_type      type,            /* type of returned value        */
    Pstore         store,           /* handle to Store object        */
    Pint           *err_ind,        /* OUT error indicator           */
    Pop_mode       *op_mode,        /* OUT operating mode            */
    Pecho_switch   *echo_switch,    /* OUT echo switch               */
    Pint           *init_view_ind,  /* OUT initial view index        */
    Ppoint         *init_loc_pos,   /* OUT initial locator position  */
    Pint           *prompt_echo,    /* OUT prompt/echo type          */
    Plimit         *echo_area,      /* OUT echo area                 */
    Ploc_data      **loc_data       /* OUT data record               */
);
```

```
/* INQUIRE STROKE DEVICE STATE 3 */
extern void pinq_stroke_st3(
    Pint            ws_id,          /* workstation identifier   */
    Pint            stroke_num,     /* stroke device number     */
    Pinq_type       type,           /* type of returned value   */
    Pstore          store,          /* handle to Store object   */
    Pint            *err_ind,       /* OUT error indicator      */
    Pop_mode        *op_mode,       /* OUT operating mode       */
    Pecho_switch    *echo_switch,   /* OUT echo switch          */
    Pint            *init_view_ind, /* OUT initial view index   */
    Ppoint_list3    **init_stroke,  /* OUT initial stroke       */
    Pint            *prompt_echo,   /* OUT prompt/echo type     */
    Plimit3         *echo_vol,      /* OUT echo volume          */
    Pstroke_data3   **stroke_data   /* OUT data record          */
);

/* INQUIRE STROKE DEVICE STATE */
extern void pinq_stroke_st(
    Pint            ws_id,          /* workstation identifier   */
    Pint            stroke_num,     /* stoke device number      */
    Pinq_type       type,           /* type of returned value   */
    Pstore          store,          /* handle to Store object   */
    Pint            *err_ind,       /* OUT error indicator      */
    Pop_mode        *op_mode,       /* OUT operating mode       */
    Pecho_switch    *echo_switch,   /* OUT echo switch          */
    Pint            *init_view_ind, /* OUT initial view index   */
    Ppoint_list     **init_stroke,  /* OUT initial stroke       */
    Pint            *prompt_echo,   /* OUT prompt/echo type     */
    Plimit          *echo_area,     /* OUT echo area            */
    Pstroke_data    **stroke_data   /* OUT data record          */
);

/* INQUIRE VALUATOR DEVICE STATE 3 */
extern void pinq_val_st3(
    Pint            ws_id,          /* workstation identifier   */
    Pint            val_num,        /* valuator device number   */
    Pstore          store,          /* handle to Store object   */
    Pint            *err_ind,       /* OUT error indicator      */
    Pop_mode        *op_mode,       /* OUT operating mode       */
    Pecho_switch    *echo_switch,   /* OUT echo switch          */
    Pfloat          *init_value,    /* OUT initial value        */
    Pint            *prompt_echo,   /* OUT prompt/echo type     */
    Plimit3         *echo_vol,      /* OUT echo volume          */
    Pval_data3      **val_data      /* OUT data record          */
);
```

```
/* INQUIRE VALUATOR DEVICE STATE */
extern void pinq_val_st(
    Pint          ws_id,          /* workstation identifier   */
    Pint          val_num,        /* valuator device number   */
    Pstore        store,          /* handle to Store object   */
    Pint          *err_ind,       /* OUT error indicator      */
    Pop_mode      *op_mode,       /* OUT operating mode       */
    Pecho_switch  *echo_switch,   /* OUT echo switch          */
    Pfloat        *init_value,    /* OUT initial value        */
    Pint          *prompt_echo,   /* OUT prompt/echo type     */
    Plimit        *echo_area,     /* OUT echo area            */
    Pval_data     **val_data      /* OUT data record          */
);

/* INQUIRE CHOICE DEVICE STATE 3 */
extern void pinq_choice_st3(
    Pint           ws_id,          /* workstation identifier      */
    Pint           choice_num,     /* choice device number        */
    Pstore         store,          /* handle to Store object      */
    Pint           *err_ind,       /* OUT error indicator         */
    Pop_mode       *op_mode,       /* OUT operating mode          */
    Pecho_switch   *echo_switch,   /* OUT echo switch             */
    Pin_status     *init_status,   /* OUT initial choice status   */
    Pint           *init_choice,   /* OUT initial choice          */
    Pint           *prompt_echo,   /* OUT prompt/echo type        */
    Plimit3        *echo_vol,      /* OUT echo volume             */
    Pchoice_data3  **choice_data   /* OUT data record             */
);

/* INQUIRE CHOICE DEVICE STATE */
extern void pinq_choice_st(
    Pint          ws_id,          /* workstation identifier      */
    Pint          choice_num,     /* choice device number        */
    Pstore        store,          /* handle to Store object      */
    Pint          *err_ind,       /* OUT error indicator         */
    Pop_mode      *op_mode,       /* OUT operating mode          */
    Pecho_switch  *echo_switch,   /* OUT echo switch             */
    Pin_status    *init_status,   /* OUT initial choice status   */
    Pint          *init_choice,   /* OUT initial choice          */
    Pint          *prompt_echo,   /* OUT prompt/echo type        */
    Plimit        *echo_area,     /* OUT echo area               */
    Pchoice_data  **choice_data   /* OUT data record             */
);
```

```
/* INQUIRE PICK DEVICE STATE 3 */
extern void pinq_pick_st3(
    Pint           ws_id,          /* workstation identifier    */
    Pint           pick_num,       /* pick device number        */
    Pinq_type      type,           /* type of returned value    */
    Pstore         store,          /* handle to Store object    */
    Pint           *err_ind,       /* OUT error indicator       */
    Pop_mode       *op_mode,       /* OUT operating mode        */
    Pecho_switch   *echo_switch,   /* OUT echo switch           */
    Pfilter        **pick_filter,  /* OUT pick filter           */
    Pin_status     *init_status,   /* OUT initial pick status   */
    Ppick_path     **init_pick,    /* OUT initial pick path     */
    Pint           *prompt_echo,   /* OUT prompt/echo type      */
    Plimit3        *echo_vol,      /* OUT echo volume           */
    Ppick_data3    **pick_data,    /* OUT data record           */
    Ppath_order    *order          /* OUT pick path order       */
);

/* INQUIRE PICK DEVICE STATE */
extern void pinq_pick_st(
    Pint           ws_id,          /* workstation identifier    */
    Pint           pick_num,       /* pick device number        */
    Pinq_type      type,           /* type of returned value    */
    Pstore         store,          /* handle to Store object    */
    Pint           *err_ind,       /* OUT error indicator       */
    Pop_mode       *op_mode,       /* OUT operating mode        */
    Pecho_switch   *echo_switch,   /* OUT echo switch           */
    Pfilter        **pick_filter,  /* OUT pick filter           */
    Pin_status     *init_status,   /* OUT initial pick status   */
    Ppick_path     **init_pick,    /* OUT initial pick path     */
    Pint           *prompt_echo,   /* OUT prompt/echo type      */
    Plimit         *echo_area,     /* OUT echo area             */
    Ppick_data     **pick_data,    /* OUT data record           */
    Ppath_order    *order          /* OUT pick path order       */
);

/* INQUIRE STRING DEVICE STATE 3 */
extern void pinq_string_st3(
    Pint           ws_id,          /* workstation identifier    */
    Pint           string_num,     /* string device number      */
    Pstore         store,          /* handle to Store object    */
    Pint           *err_ind,       /* OUT error indicator       */
    Pop_mode       *op_mode,       /* OUT operating mode        */
    Pecho_switch   *echo_switch,   /* OUT echo switch           */
    char           **init_string,  /* OUT initial string        */
    Pint           *prompt_echo,   /* OUT prompt/echo type      */
    Plimit3        *echo_vol,      /* OUT echo volume           */
    Pstring_data3  **string_data   /* OUT data record           */
);
```

```
/* INQUIRE STRING DEVICE STATE */
extern void pinq_string_st(
    Pint          ws_id,          /* workstation identifier   */
    Pint          string_num,     /* string device number     */
    Pstore        store,          /* handle to Store object   */
    Pint          *err_ind,       /* OUT error indicator      */
    Pop_mode      *op_mode,       /* OUT operating mode       */
    Pecho_switch  *echo_switch,   /* OUT echo switch          */
    char          **init_string,  /* OUT initial string       */
    Pint          *prompt_echo,   /* OUT prompt/echo type     */
    Plimit        *echo_area,     /* OUT echo area            */
    Pstring_data  **string_data   /* OUT data record          */
);

/* INQUIRE WORKSTATION CATEGORY */
extern void pinq_ws_cat(
    Pint      ws_type,    /* workstation type          */
    Pint      *err_ind,   /* OUT error indicator       */
    Pws_cat   *cat        /* OUT workstation category  */
);

/* INQUIRE DISPLAY SPACE SIZE 3 */
extern void pinq_disp_space_size3(
    Pint                ws_type,    /* workstation type      */
    Pint                *err_ind,   /* OUT error indicator   */
    Pdisp_space_size3   *size       /* OUT display size      */
);

/* INQUIRE DISPLAY SPACE SIZE */
extern void pinq_disp_space_size(
    Pint                ws_type,    /* workstation type      */
    Pint                *err_ind,   /* OUT error indicator   */
    Pdisp_space_size    *size       /* OUT display size      */
);

/* INQUIRE HLHSR IDENTIFIER FACILITIES */
extern void pinq_hlhsr_id_facs(
    Pint        ws_type,                /* workstation type          */
    Pint        num_elems_appl_list,    /* # elems in appl list      */
    Pint        start_ind,              /* starting index            */
    Pint        *err_ind,               /* OUT error indicator       */
    Pint_list   *hlhsr_ids,             /* OUT list of HLHSR ids     */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list  */
);
```

```
/* INQUIRE HLHSR MODE FACILITIES */
extern void pinq_hlhsr_mode_facs(
    Pint        ws_type,              /* workstation type          */
    Pint        num_elems_appl_list,  /* # elems in appl list      */
    Pint        start_ind,            /* starting index            */
    Pint        *err_ind,             /* OUT error indicator       */
    Pint_list   *hlhsr_modes,         /* OUT list of HLHSR modes   */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list  */
);

/* INQUIRE VIEW FACILITIES */
extern void pinq_view_facs(
    Pint   ws_type,        /* workstation type                     */
    Pint   *err_ind,       /* OUT error indicator                  */
    Pint   *num_view_ind   /* OUT number of predefined view indices */
);

/* INQUIRE PREDEFINED VIEW REPRESENTATION */
extern void pinq_pred_view_rep(
    Pint         ws_type,    /* workstation type          */
    Pint         index,      /* predefined view index     */
    Pint         *err_ind,   /* OUT error indicator       */
    Pview_rep3   *view       /* OUT view representation   */
);

/* INQUIRE WORKSTATION CLASSIFICATION */
extern void pinq_ws_class(
    Pint        ws_type,    /* workstation type        */
    Pint        *err_ind,   /* OUT error indicator     */
    Pws_class   *ws_class   /* OUT workstation class   */
);

/* INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES */
extern void pinq_dyns_ws_attrs(
    Pint            ws_type,    /* workstation type          */
    Pint            *err_ind,   /* OUT error indicator       */
    Pdyns_ws_attrs  *attr       /* OUT attributes dynamics   */
);

/* INQUIRE DEFAULT DISPLAY UPDATE STATE */
extern void pinq_def_disp_upd_st(
    Pint          ws_type,     /* workstation type        */
    Pint          *err_ind,    /* OUT error indicator     */
    Pdefer_mode   *def_mode,   /* OUT deferral mode       */
    Pmod_mode     *mod_mode    /* OUT modification mode   */
);
```

```
/* INQUIRE POLYLINE FACILITIES */
extern void pinq_line_facs(
    Pint         ws_type,               /* workstation type          */
    Pint         num_elems_appl_list,   /* # elems in appl list      */
    Pint         start_ind,             /* starting index            */
    Pint         *err_ind,              /* OUT error indicator       */
    Pline_facs   *fac,                  /* OUT polyline facilities   */
    Pint         *num_elems_impl_list   /* OUT # elems in impl list  */
);

/* INQUIRE PREDEFINED POLYLINE REPRESENTATION */
extern void pinq_pred_line_rep(
    Pint           ws_type,    /* workstation type              */
    Pint           index,      /* predefined index              */
    Pint           *err_ind,   /* OUT error indicator           */
    Pline_bundle   *bundle     /* OUT predefined polyline rep   */
);

/* INQUIRE POLYMARKER FACILITIES */
extern void pinq_marker_facs(
    Pint           ws_type,               /* workstation type          */
    Pint           num_elems_appl_list,   /* # elems in appl list      */
    Pint           start_ind,             /* starting index            */
    Pint           *err_ind,              /* OUT error indicator       */
    Pmarker_facs   *fac,                  /* OUT polymarker facilities */
    Pint           *num_elems_impl_list   /* OUT # elems in impl list  */
);

/* INQUIRE PREDEFINED POLYMARKER REPRESENTATION */
extern void pinq_pred_marker_rep(
    Pint             ws_type,    /* workstation type              */
    Pint             index,      /* predefined index              */
    Pint             *err_ind,   /* OUT error indicator           */
    Pmarker_bundle   *bundle     /* OUT predefined polymarker rep */
);

/* INQUIRE TEXT FACILITIES */
extern void pinq_text_facs(
    Pint         ws_type,               /* workstation type          */
    Pint         num_elems_appl_list,   /* # elems in appl list      */
    Pint         start_ind,             /* starting index            */
    Pint         *err_ind,              /* OUT error indicator       */
    Ptext_facs   *fac,                  /* OUT text facilities       */
    Pint         *num_elems_impl_list   /* OUT # elems in impl list  */
);

/* INQUIRE PREDEFINED TEXT REPRESENTATION */
extern void pinq_pred_text_rep(
    Pint           ws_type,    /* workstation type          */
    Pint           index,      /* predefined index          */
    Pint           *err_ind,   /* OUT error indicator       */
    Ptext_bundle   *bundle     /* OUT predefined text rep   */
);
```

```
/* INQUIRE ANNOTATION FACILITIES */
extern void pinq_anno_facs(
    Pint        ws_type,                /* workstation type                          */
    Pint        num_elems_appl_list,    /* # elems in appl list                      */
    Pint        start_ind,              /* starting index                            */
    Pint        *err_ind,               /* OUT error indicator                       */
    Pint_list   *styles,                /* OUT list annotation styles                */
    Pint        *num_elems_impl_list,   /* OUT # elems in impl list                  */
    Pint        *num_anno_char_hts,     /* OUT number of anno. text char. heights    */
    Pfloat      *min_anno_char_ht,      /* OUT minimum anno. text char. height       */
    Pfloat      *max_anno_char_ht       /* OUT maximum anno. text char. height       */
);

/* INQUIRE TEXT EXTENT */
extern void pinq_text_extent(
    Pint              ws_type,          /* workstation type          */
    Pint              text_font,        /* text font                 */
    Pfloat            char_expan,       /* char expansion factor     */
    Pfloat            char_space,       /* char spacing              */
    Pfloat            char_ht,          /* char height               */
    Ptext_path        text_path,        /* text path                 */
    Phor_text_align   hor_text_align,   /* horizontal text alignment */
    Pvert_text_align  ver_text_align,   /* vertical text alignment   */
    const char        *char_string,     /* character string          */
    Pint              *err_ind,         /* OUT error indicator       */
    Prect             *rect,            /* OUT extent rectangle      */
    Ppoint            *offset           /* OUT concatenation offset  */
);

/* INQUIRE INTERIOR FACILITIES */
extern void pinq_int_facs(
    Pint        ws_type,                        /* workstation type                    */
    Pint        hatch_num_elems_appl_length,    /* # elems in appl hatch style list    */
    Pint        hatch_start_ind,                /* starting index                      */
    Pint        *err_ind,                       /* OUT error indicator                 */
    Pint_facs   *int_facs,                      /* OUT interior facilities             */
    Pint        *hatch_num_elems_impl_list      /* OUT # elems in impl list            */
);

/* INQUIRE PREDEFINED INTERIOR REPRESENTATION */
extern void pinq_pred_int_rep(
    Pint          ws_type,    /* workstation type              */
    Pint          index,      /* predefined index              */
    Pint          *err_ind,   /* OUT error indicator           */
    Pint_bundle   *bundle     /* OUT predefined interior rep   */
);
```

```
/* INQUIRE EDGE FACILITIES */
extern void pinq_edge_facs(
    Pint         ws_type,              /* workstation type          */
    Pint         num_elems_appl_list,  /* # elems in appl list      */
    Pint         start_ind,            /* starting index            */
    Pint         *err_ind,             /* OUT error indicator       */
    Pedge_facs   *fac,                 /* OUT edge facilities       */
    Pint         *num_elems_impl_list  /* OUT # elems in impl list  */
);

/* INQUIRE PREDEFINED EDGE REPRESENTATION */
extern void pinq_pred_edge_rep(
    Pint           ws_type,    /* workstation type          */
    Pint           index,      /* predefined index          */
    Pint           *err_ind,   /* OUT error indicator       */
    Pedge_bundle   *bundle     /* OUT predefined edge rep   */
);

/* INQUIRE PATTERN FACILITIES */
extern void pinq_pat_facs(
    Pint   ws_type,    /* workstation type                          */
    Pint   *err_ind,   /* OUT error indicator                       */
    Pint   *num_pred   /* OUT number of predefined pattern indices  */
);

/* INQUIRE PREDEFINED PATTERN REPRESENTATION */
extern void pinq_pred_pat_rep(
    Pint       ws_type,    /* workstation type            */
    Pint       index,      /* predefined index            */
    Pstore     store,      /* handle to Store object      */
    Pint       *err_ind,   /* OUT error indicator         */
    Ppat_rep   **pat_rep   /* OUT predefined pattern rep  */
);

/* INQUIRE COLOUR MODEL FACILITIES */
extern void pinq_colr_model_facs(
    Pint        ws_type,               /* workstation type          */
    Pint        num_elems_appl_list,   /* # elems in appl list      */
    Pint        start_ind,             /* starting index            */
    Pint        *err_ind,              /* OUT error indicator       */
    Pint_list   *models,               /* OUT list of colour models */
    Pint        *num_elems_impl_list,  /* OUT # elems in impl list  */
    Pint        *def                   /* OUT default colour model  */
);

/* INQUIRE COLOUR FACILITIES */
extern void pinq_colr_facs(
    Pint         ws_type,    /* workstation type       */
    Pint         *err_ind,   /* OUT error indicator    */
    Pcolr_facs   *fac        /* OUT colour facilities  */
);
```

```
/* INQUIRE PREDEFINED COLOUR REPRESENTATION */
extern void pinq_pred_colr_rep(
    Pint        ws_type,    /* workstation type                        */
    Pint        colr_ind,   /* predefined index                        */
    Pint        *err_ind,   /* OUT error indicator                     */
    Pcolr_rep   *colr_rep   /* OUT predefined colour representation    */
);

/* INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES 3 */
extern void pinq_list_avail_gdp3(
    Pint        ws_type,                /* workstation type            */
    Pint        num_elems_appl_list,    /* # elems in appl list        */
    Pint        start_ind,              /* starting index              */
    Pint        *err_ind,               /* OUT error indicator         */
    Pint_list   *gdps,                  /* OUT list of GDPs            */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list    */
);

/* INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES */
extern void pinq_list_avail_gdp(
    Pint        ws_type,                /* workstation type            */
    Pint        num_elems_appl_list,    /* # elems in appl list        */
    Pint        start_ind,              /* starting index              */
    Pint        *err_ind,               /* OUT error indicator         */
    Pint_list   *gdps,                  /* OUT list of GDPs            */
    Pint        *num_elems_impl_list    /* OUT # elems in impl list    */
);

/* INQUIRE GENERALIZED DRAWING PRIMITIVE 3 */
extern void pinq_gdp3(
    Pint     ws_type,       /* workstation type                */
    Pint     gdp,           /* GDP function identifier         */
    Pint     *err_ind,      /* OUT error indicator             */
    Pint     *num_attr,     /* OUT number of attributes used   */
    Pattrs   attr[5]        /* OUT list of attributes used     */
);

/* INQUIRE GENERALIZED DRAWING PRIMITIVE */
extern void pinq_gdp(
    Pint     ws_type,       /* workstation type                */
    Pint     gdp,           /* GDP function identifier         */
    Pint     *err_ind,      /* OUT error indicator             */
    Pint     *num_attr,     /* OUT number of attributes used   */
    Pattrs   attr[5]        /* OUT list of attributes used     */
);
```

```
/* INQUIRE LIST OF AVAILABLE GENERALIZED STRUCTURE ELEMENTS */
extern void pinq_list_avail_gse(
    Pint        ws_type,              /* workstation type          */
    Pint        num_elems_appl_list,  /* # elems in appl list      */
    Pint        start_ind,            /* starting index            */
    Pint        *err_ind,             /* OUT error indicator       */
    Pint_list   *gses,                /* OUT list of GSEs          */
    Pint        *num_elems_impl_list  /* OUT # elems in impl list  */
);

/* INQUIRE NUMBER OF DISPLAY PRIORITIES SUPPORTED */
extern void pinq_num_disp_pris(
    Pint   ws_type,    /* workstation type                    */
    Pint   *err_ind,   /* OUT error indicator                 */
    Pint   *num_pri    /* OUT number of structure priorities  */
);

/* INQUIRE WORKSTATION STATE TABLE LENGTHS */
extern void pinq_ws_st_table(
    Pint            ws_type,    /* workstation type                  */
    Pint            *err_ind,   /* OUT error indicator               */
    Pws_st_tables   *lengths    /* OUT lengths of workstation tables */
);

/* INQUIRE DYNAMICS OF STRUCTURES */
extern void pinq_dyns_structs(
    Pint            ws_type,    /* workstation type          */
    Pint            *err_ind,   /* OUT error indicator       */
    Pdyns_structs   *dyns       /* OUT structure dynamics    */
);

/* INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES */
extern void pinq_num_avail_in(
    Pint      ws_type,    /* workstation type            */
    Pint      *err_ind,   /* OUT error indicator         */
    Pnum_in   *num_in     /* OUT number of input devices */
);

/* INQUIRE DEFAULT LOCATOR DEVICE DATA 3 */
extern void pinq_def_loc_data3(
    Pint         ws_type,      /* workstation type                  */
    Pint         loc_num,      /* logical input device number       */
    Pstore       store,        /* handle to Store object            */
    Pint         *err_ind,     /* OUT error indicator               */
    Ppoint3      *loc_pos,     /* OUT default initial position      */
    Pint_list    **pet_list,   /* OUT list of prompt and echo types */
    Plimit3      *echo_vol,    /* OUT default echo volume           */
    Ploc_data3   **loc_data    /* OUT default data record           */
);
```

```
/* INQUIRE DEFAULT LOCATOR DEVICE DATA */
extern void pinq_def_loc_data(
    Pint        ws_type,      /* workstation type                  */
    Pint        loc_num,      /* logical input device number       */
    Pstore      store,        /* handle to Store object            */
    Pint        *err_ind,     /* OUT error indicator               */
    Ppoint      *loc_pos,     /* OUT default locator position      */
    Pint_list   **pet_list,   /* OUT list of prompt and echo types */
    Plimit      *echo_area,   /* OUT default echo area             */
    Ploc_data   **loc_data    /* OUT default data record           */
);

/* INQUIRE DEFAULT STROKE DEVICE DATA 3 */
extern void pinq_def_stroke_data3(
    Pint            ws_type,        /* workstation type                  */
    Pint            stroke_num,     /* logical input device number       */
    Pstore          store,          /* handle to Store object            */
    Pint            *err_ind,       /* OUT error indicator               */
    Pint            *max_buf_size,  /* OUT max. input buffer size        */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types */
    Plimit3         *echo_vol,      /* OUT default echo volume           */
    Pstroke_data3   **stroke_data   /* OUT default data record           */
);

/* INQUIRE DEFAULT STROKE DEVICE DATA */
extern void pinq_def_stroke_data(
    Pint            ws_type,        /* workstation type                  */
    Pint            stroke_num,     /* logical input device number       */
    Pstore          store,          /* handle to Store object            */
    Pint            *err_ind,       /* OUT error indicator               */
    Pint            *max_buf_size,  /* OUT max. input buffer size        */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types */
    Plimit          *echo_area,     /* OUT default echo area             */
    Pstroke_data    **stroke_data   /* OUT default data record           */
);

/* INQUIRE DEFAULT VALUATOR DEVICE DATA 3 */
extern void pinq_def_val_data3(
    Pint        ws_type,      /* workstation type                  */
    Pint        val_num,      /* logical input device number       */
    Pstore      store,        /* handle to Store object            */
    Pint        *err_ind,     /* OUT error indicator               */
    Pfloat      *def_value,   /* OUT default initial value         */
    Pint_list   **pet_list,   /* OUT list of prompt and echo types */
    Plimit3     *echo_vol,    /* OUT default echo volume           */
    Pval_data3  **val_data    /* OUT default data record           */
);
```

```
/* INQUIRE DEFAULT VALUATOR DEVICE DATA */
extern void pinq_def_val_data(
    Pint        ws_type,      /* workstation type                    */
    Pint        val_num,      /* logical input device number         */
    Pstore      store,        /* handle to Store object              */
    Pint        *err_ind,     /* OUT error indicator                 */
    Pfloat      *def_value,   /* OUT default initial value           */
    Pint_list   **pet_list,   /* OUT list of prompt and echo types   */
    Plimit      *echo_area,   /* OUT default echo area               */
    Pval_data   **val_data    /* OUT default data record             */
);

/* INQUIRE DEFAULT CHOICE DEVICE DATA 3 */
extern void pinq_def_choice_data3(
    Pint            ws_type,        /* workstation type                    */
    Pint            choice_num,     /* logical input device number         */
    Pstore          store,          /* handle to Store object              */
    Pint            *err_ind,       /* OUT error indicator                 */
    Pint            *max_choices,   /* OUT max. number of choices          */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types   */
    Plimit3         *echo_vol,      /* OUT default echo volume             */
    Pchoice_data3   **choice_data   /* OUT default data record             */
);

/* INQUIRE DEFAULT CHOICE DEVICE DATA */
extern void pinq_def_choice_data(
    Pint            ws_type,        /* workstation type                    */
    Pint            choice_num,     /* logical input device number         */
    Pstore          store,          /* handle to Store object              */
    Pint            *err_ind,       /* OUT error indicator                 */
    Pint            *max_choices,   /* OUT max. number of choices          */
    Pint_list       **pet_list,     /* OUT list of prompt and echo types   */
    Plimit          *echo_area,     /* OUT default echo area               */
    Pchoice_data    **choice_data   /* OUT default data record             */
);

/* INQUIRE DEFAULT PICK DEVICE DATA 3 */
extern void pinq_def_pick_data3(
    Pint        ws_type,      /* workstation type                    */
    Pint        pick_num,     /* logical input device number         */
    Pstore      store,        /* handle to Store object              */
    Pint        *err_ind,     /* OUT error indicator                 */
    Pint_list   **pet_list,   /* OUT list of prompt and echo types   */
    Plimit3     *echo_vol,    /* OUT default echo volume             */
    Ppick_data3 **pick_data   /* OUT default data record             */
);
```

```
/* INQUIRE DEFAULT PICK DEVICE DATA */
extern void pinq_def_pick_data(
    Pint          ws_type,      /* workstation type                    */
    Pint          pick_num,     /* logical input device number         */
    Pstore        store,        /* handle to Store object              */
    Pint          *err_ind,     /* OUT error indicator                 */
    Pint_list     **pet_list,   /* OUT list of prompt and echo types   */
    Plimit        *echo_area,   /* OUT default echo area               */
    Ppick_data    **pick_data   /* OUT default data record             */
);

/* INQUIRE DEFAULT STRING DEVICE DATA 3 */
extern void pinq_def_string_data3(
    Pint             ws_type,         /* workstation type                    */
    Pint             string_num,      /* logical input device number         */
    Pstore           store,           /* handle to Store object              */
    Pint             *err_ind,        /* OUT error indicator                 */
    Pint             *max_buf_size,   /* OUT max. input buffer size          */
    Pint_list        **pet_list,      /* OUT list of prompt and echo types   */
    Plimit3          *echo_vol,       /* OUT default echo volume             */
    Pstring_data3    **string_data    /* OUT default data record             */
);

/* INQUIRE DEFAULT STRING DEVICE DATA */
extern void pinq_def_string_data(
    Pint           ws_type,         /* workstation type                    */
    Pint           string_num,      /* logical input device number         */
    Pstore         store,           /* handle to Store object              */
    Pint           *err_ind,        /* OUT error indicator                 */
    Pint           *max_buf_size,   /* OUT max. input buffer size          */
    Pint_list      **pet_list,      /* OUT list of prompt and echo types   */
    Plimit         *echo_area,      /* OUT default echo area               */
    Pstring_data   **string_data    /* OUT default data record             */
);

/* INQUIRE SET OF WORKSTATIONS TO WHICH POSTED */
extern void pinq_wss_posted(
    Pint        struct_id,             /* structure identifier       */
    Pint        num_elems_appl_list,   /* # elems in appl list       */
    Pint        start_ind,             /* starting index             */
    Pint        *err_ind,              /* OUT error indicator        */
    Pint_list   *ws,                   /* OUT list of workstations   */
    Pint        *num_elems_impl_list   /* OUT # elems in impl list   */
);

/* INQUIRE OPEN STRUCTURE */
extern void pinq_open_struct(
    Pint                  *err_ind,    /* OUT error indicator            */
    Popen_struct_status   *status,     /* OUT status of open structure   */
    Pint                  *struct_id   /* OUT structure identifier       */
);
```

```
/* INQUIRE ELEMENT POINTER */
extern void pinq_elem_ptr(
    Pint   *err_ind,         /* OUT error indicator         */
    Pint   *elem_ptr_value   /* OUT element pointer value   */
);

/* INQUIRE CURRENT ELEMENT TYPE AND SIZE */
extern void pinq_cur_elem_type_size(
    Pint         *err_ind,     /* OUT error indicator   */
    Pelem_type   *elem_type,   /* OUT element type      */
    size_t       *elem_size    /* OUT element size      */
);

/* INQUIRE CURRENT ELEMENT CONTENT */
extern void pinq_cur_elem_content(
    Pstore       store,        /* handle to Store object   */
    Pint         *err_ind,     /* OUT error indicator      */
    Pelem_data   **elem_data   /* OUT element data         */
);

/* INQUIRE ELEMENT TYPE AND SIZE */
extern void pinq_elem_type_size(
    Pint         struct_id,    /* structure identifier   */
    Pint         elem_num,     /* element number         */
    Pint         *err_ind,     /* OUT error indicator    */
    Pelem_type   *elem_type,   /* OUT element type       */
    size_t       *elem_size    /* OUT element size       */
);

/* INQUIRE ELEMENT CONTENT */
extern void pinq_elem_content(
    Pint         struct_id,    /* structure identifier     */
    Pint         elem_num,     /* element number           */
    Pstore       store,        /* handle to Store object   */
    Pint         *err_ind,     /* OUT error indicator      */
    Pelem_data   **elem_data   /* OUT data record          */
);

/* INQUIRE STRUCTURE STATUS */
extern void pinq_struct_status(
    Pint             struct_id,   /* structure identifier   */
    Pint             *err_ind,    /* OUT error indicator    */
    Pstruct_status   *status      /* OUT existence status   */
);
```

```
/* INQUIRE PATHS TO ANCESTORS */
extern void pinq_paths_ances(
    Pint                  struct_id,   /* structure identifier      */
    Ppath_order           order,       /* path order                */
    Pint                  depth,       /* path depth                */
    Pstore                store,       /* handle to Store object    */
    Pint                  *err_ind,    /* OUT error indicator       */
    Pelem_ref_list_list   **paths      /* OUT structure path list   */
);

/* INQUIRE PATHS TO DESCENDANTS */
extern void pinq_paths_descs(
    Pint                  struct_id,   /* structure identifier      */
    Ppath_order           order,       /* path order                */
    Pint                  depth,       /* path depth                */
    Pstore                store,       /* handle to Store object    */
    Pint                  *err_ind,    /* OUT error indicator       */
    Pelem_ref_list_list   **paths      /* OUT structure path list   */
);

/* ELEMENT SEARCH */
extern void pelem_search(
    Pint                    struct_id,      /* structure identifier        */
    Pint                    start_elem,     /* starting element position   */
    Psearch_dir             dir,            /* search direction            */
    const Pelem_type_list   *incl,          /* element incl. list          */
    const Pelem_type_list   *excl,          /* element excl. list          */
    Pint                    *err_ind,       /* OUT error indicator         */
    Psearch_status          *status,        /* OUT search status           */
    Pint                    *found_elem_ptr /* OUT found element pointer   */
);

/* INCREMENTAL SPATIAL SEARCH 3 */
extern void pincr_spa_search3(
    const Ppoint3           *ref_pt,            /* search reference point     */
    Pfloat                  dist,               /* search distance            */
    const Pelem_ref_list    *sp,                /* starting path list         */
    Pclip_ind               clip_ind,           /* modeling clip indicator    */
    Pint                    ceiling,            /* search ceiling index       */
    const Pfilter_list      *norm,              /* normal filter list         */
    const Pfilter_list      *inv,               /* inverted filter list       */
    Pint                    num_elems_appl_list,/* # elems in appl list       */
    Pint                    start_ind,          /* starting index             */
    Pint                    *err_ind,           /* OUT error indicator        */
    Pelem_ref_list          *fp,                /* OUT found path             */
    Pint                    *num_elems_impl_list/* OUT # elems in impl list   */
);
```

```
/* INCREMENTAL SPATIAL SEARCH */
extern void pincr_spa_search(
    const Ppoint          *ref_pt,              /* search reference point     */
    Pfloat                dist,                 /* search distance            */
    const Pelem_ref_list  *sp,                  /* starting path list         */
    Pclip_ind             clip_ind,             /* modeling clip indicator    */
    Pint                  ceiling,              /* search ceiling index       */
    const Pfilter_list    *norm,                /* normal filter list         */
    const Pfilter_list    *inv,                 /* inverted filter list       */
    Pint                  num_elems_appl_list,  /* # elems in appl list       */
    Pint                  start_ind,            /* starting index             */
    Pint                  *err_ind,             /* OUT error indicator        */
    Pelem_ref_list        *fp,                  /* OUT found path             */
    Pint                  *num_elems_impl_list  /* OUT # elems in impl list   */
);

/* INQUIRE INPUT QUEUE OVERFLOW */
extern void pinq_in_overf(
    Pint        *err_ind,    /* OUT error indicator          */
    Pint        *ws_id,      /* OUT workstation identifier   */
    Pin_class   *in_class,   /* OUT input class              */
    Pint        *in_num      /* OUT input device number      */
);

/* INQUIRE ERROR HANDLING MODE */
extern void pinq_err_hand_mode(
    Pint        *err_ind,    /* OUT error indicator   */
    Perr_mode   *err_mode    /* OUT error mode        */
);

/* EMERGENCY CLOSE PHIGS */
extern void pemergency_close_phigs(
        void
);

/* ERROR HANDLING */
extern void perr_hand(
    Pint          error_num,    /* error number                                   */
    Pint          func_num,     /* identifier of function that detected the error */
    const char    *error_file   /* name of error file                             */
);

/* ERROR LOGGING */
extern void perr_log(
    Pint          error_num,    /* error number                                   */
    Pint          func_num,     /* identifier of function that detected the error */
    const char    *error_file   /* name of error file                             */
);

/* SET ERROR HANDLING MODE */
extern void pset_err_hand_mode(
    Perr_mode   error_mode   /* error handling mode   */
);
```